32-BIT ARM-BASED MICROPROCESSORS

SAM9N12/SAM9CN11/SAM9CN12

Description

The SAM9N and SAM9CN Arm926EJ-S™-based embedded MPUs offer the frequently-requested combination of user interface functionality and high data rate connectivity, with LCD controller, resistive touchscreen, multiple UARTs, SPI, I2C, full-speed USB Host and Device and SDIO.

These eMPUs support the latest generation of LPDDR/DDR2 and NAND Flash memory interfaces for program and data storage. An internal 133 MHz multi-layer bus architecture associated with eight DMA channels and distributed memory—including a 32-Kbyte SRAM—sustains the high bandwidth required by the processor and the high-speed peripherals.

The SAM9CN devices offer on-chip hardware accelerators with DMA support that enable high-speed data encryption and authentication of transferred data or applications. Supported standards are up to 256-bit AES, and FIPS Publication 180-2 compliant SHA1 and SHA256. A True Random Number Generator is embedded for key generation and exchange protocols. The devices also feature fuse bits for crypto key (SAM9CN12), user configuration (SAM9N12 and SAM9CN11) and device configuration (all). The SAM9CN12 includes a secure Boot ROM; the SAM9N12 and SAM9CN11 include a standard Boot ROM.

The I/Os support 1.8V or 3.3V operation and are independently configurable for the memory interface and peripheral I/Os. This feature eliminates the need for any external level shifters, while 0.8mm ball pitch packages lower PCB cost and complexity.

The SAM9N and SAM9CN power management controllers feature efficient clock gating and a battery backup section that minimizes power consumption in active and standby modes. The following table presents the embedded features of each device.

- Device Configuration

Feature	SAM9N12	SAM9CN11 (for evaluation only)	SAM9CN12
Standard Boot with BSC	✓	✓	–
Secure Boot	–	–	✓
TRNG	✓	✓	✓
AES	–	✓	✓
SHA	–	✓	✓
JTAG Access	✓	✓	–

Features

- Core
  - Arm926EJ-S Arm® Thumb® Processor running up to 400 MHz
  - 16 Kbytes Data Cache, 16 Kbytes Instruction Cache, Memory Management Unit
- Memories
  - One 128-Kbyte internal ROM embedding standard or secure bootstrap routine
  - One 32-Kbyte internal SRAM, single-cycle access at system speed
  - 32-bit External Bus Interface supporting 8-bank DDR2/LPDDR, SDR/LPSDR, Static Memories
  - MLC/SLC NAND Controller, with up to 24-bit Programmable Multibit Error Correction Code (PMECC)

© 2017 Microchip Technology Inc.

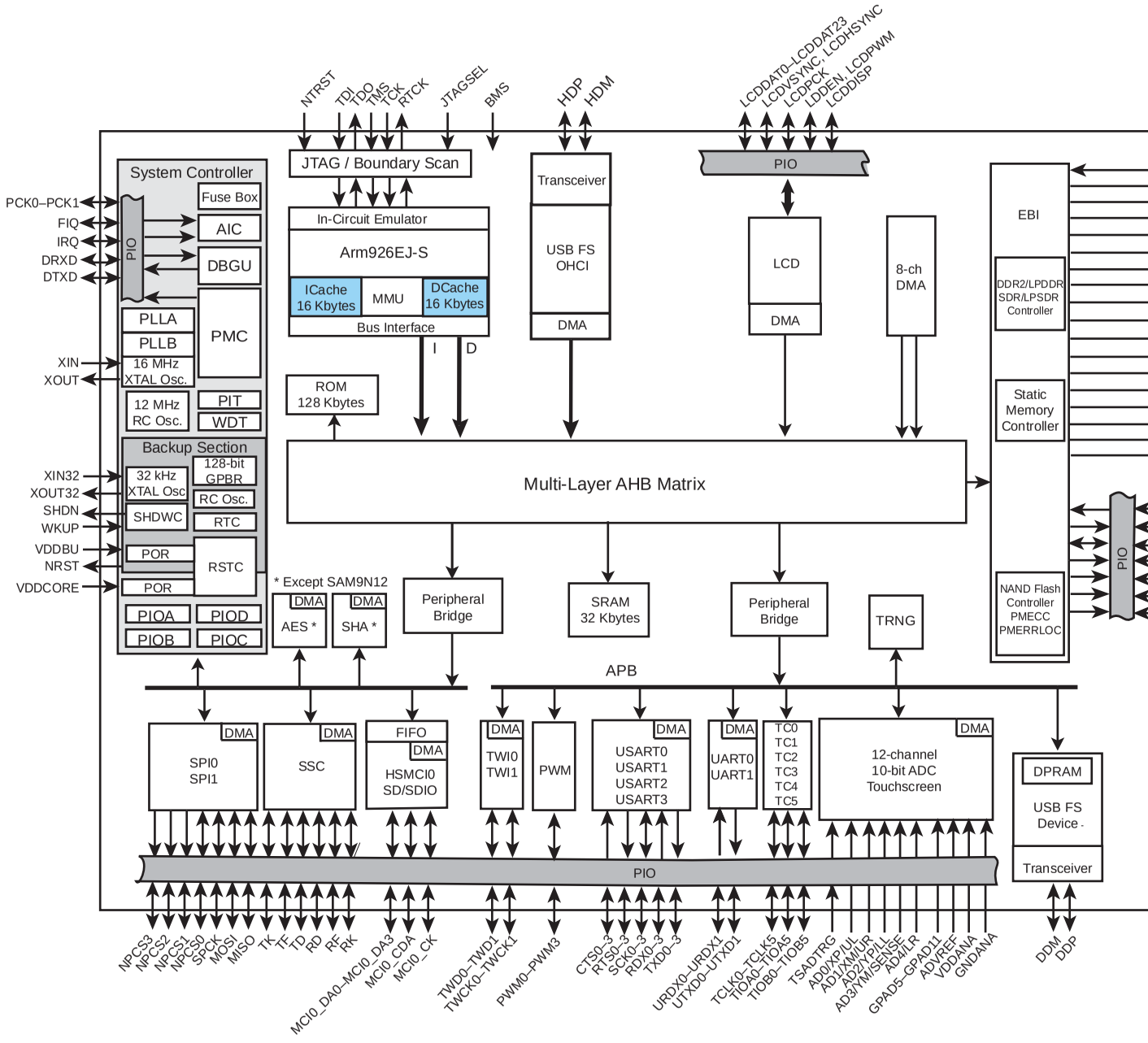
- System running up to 133 MHz
  - Power-on Reset, Reset Controller, Shutdown Controller, Periodic Interval Timer, Watchdog Timer and Real Time Clock
  - Boot Mode Select Option, Remap Command
  - Internal Low Power 32 kHz RC and Fast 12 MHz RC Oscillators
  - Selectable 32768 Hz Low-power Oscillator, 16 MHz Oscillator, one PLL for the system and one PLL optimized for USB
  - Six 32-bit-layer AHB Bus Matrix
  - Dual Peripheral Bridge with dedicated programmable clock
  - One dual port 8-channel DMA Controller
  - Advanced Interrupt Controller (AIC)
  - Two Programmable External Clock Signals
- Low-power Mode
  - Shutdown Controller with four 32-bit General-purpose Backup Registers
  - Clock Generator and Power Management Controller
  - Very Slow Clock Operating Mode, Software Programmable Power Optimization Capabilities
- Peripherals
  - LCD Controller
  - USB Device Full Speed with dedicated On-chip Transceiver
  - USB Host Full Speed with dedicated On-chip Transceiver
  - One High speed SD card and SDIO Host Controller
  - Two Master/Slave Serial Peripheral Interfaces (SPI)
  - Two 3-channel 32-bit Timer/Counters (TC)
  - One Synchronous Serial Controller (SSC)
  - One 4-channel 16-bit PWM Controller
  - Two 2-wire Interfaces (TWI)
  - Four Universal Synchronous Asynchronous Receiver Transmitters (USART)
  - Two Universal Asynchronous Receiver Transmitters (UART)
  - One Debug Unit (DBGU)
  - One 12-channel 10-bit Analog-to-Digital Converter with up to 5-wire resistive Touchscreen support
- Safety
  - Crystal Failure Detection
  - Independent Watchdog
  - Power-on Reset Cells
  - Register Write Protection
  - SHA (SHA1 and SHA256) Compliant with FIPS Publication 180-2 (SAM9CN11/SAM9CN12 devices)
- Cryptography
  - True Random Number Generator (TRNG) compliant with NIST Special Publication 800-22
  - AES 256-, 192-, 128-bit Key Algorithm compliant with FIPS Publication 197 (SAM9CN11/SAM9CN12 devices)
  - 256 Fuse bits for crypto key and 64 Fuse bits for device configuration, including JTAG disable and forced boot from the on-chip ROM
- I/O
  - Four 32-bit Parallel Input/Output Controllers
  - 105 Programmable I/O Lines Multiplexed with up to Three Peripheral I/Os
  - Input Change Interrupt Capability on Each I/O Line, optional Schmitt Trigger input
  - Individually Programmable Open-drain, Pull-up and Pull-down Resistor, Synchronous Output
- Packages
  - 217-ball BGA, pitch 0.8 mm
  - 247-ball BGA, pitch 0.5 mm

© 2017 Microchip Technology Inc.

© 2017 Microchip Technology Inc.

1. Block Diagram

Figure 1-1: SAM9N12/CN11/CN12 Block Diagram



2. Signal Description

Table 2-1 gives details on the signal names classified by peripheral.

Table 2-1: Signal Description List

Signal Name	Function	Type	Active Level
Clocks, Oscillators and PLLs			
XIN	Main Oscillator Input	Input	–
XOUT	Main Oscillator Output	Output	–
XIN32	Slow Clock Oscillator Input	Input	–
XOUT32	Slow Clock Oscillator Output	Output	–
PCK0–PCK1	Programmable Clock Output	Output	–
Shutdown, Wakeup Logic			
SHDN	Shut-Down Control	Output	–
WKUP	Wake-Up Input	Input	–
ICE and JTAG			
TCK	Test Clock	Input	–
TDI	Test Data In	Input	–
TDO	Test Data Out	Output	–
TMS	Test Mode Select	Input	–
JTAGSEL	JTAG Selection	Input	–
RTCK	Return Test Clock	Output	–
Reset/Test			
NRST	Microcontroller Reset	I/O	Low
NTRST	Test Reset Signal	Input	–
BMS	Boot Mode Select	**Input**	–
Debug Unit - DBGU			
DRXD	Debug Receive Data	Input	–
DTXD	Debug Transmit Data	Output	–
Advanced Interrupt Controller - AIC			
IRQ	External Interrupt Input	Input	–
FIQ	Fast Interrupt Input	Input	–
PIO Controller - PIOA / PIOB / PIOC / PIOD			
PA0–PA31	Parallel IO Controller A	I/O	–
PB0–PB18	Parallel IO Controller B	I/O	–
PC0–PC31	Parallel IO Controller C	I/O	–
PD0–PD21	Parallel IO Controller D	I/O	–

© 2017 Microchip Technology Inc.

Table 2-1: Signal Description List (Continued)

Signal Name	Function	Type	Active Level
External Bus Interface - EBI			
D0–D15	Data Bus	I/O	–
D16–D31	Data Bus	I/O	–
A0–A25	Address Bus	Output	–
NWAIT	External Wait Signal	Input	Low
Static Memory Controller - SMC			
NCS0–NCS5	Chip Select Lines	Output	Low
NWR0–NWR3	Write Signal	Output	Low
NRD	Read Signal	Output	Low
NWE	Write Enable	Output	Low
NBS0–NBS3	Byte Mask Signal	Output	Low
NAND Flash Support			
NFD0–NFD15	NAND Flash I/O	I/O	–
NANDCS	NAND Flash Chip Select	Output	Low
NANDOE	NAND Flash Output Enable	Output	Low
NANDWE	NAND Flash Write Enable	Output	Low
DDR2/SDRAM/LPDDR Controller			
SDCK,#SDCK	DDR2/SDRAM differential clock	Output	–
SDCKE	DDR2/SDRAM Clock Enable	Output	High
SDCS	DDR2/SDRAM Controller Chip Select	Output	Low
BA[0..2]	Bank Select	Output	Low
SDWE	DDR2/SDRAM Write Enable	Output	Low
RAS - CAS	Row and Column Signal	Output	Low
SDA10	SDRAM Address 10 Line	Output	–
DQS[0..1]	Data Strobe	I/O	–
DQM[0..3]	Write Data Mask	Output	–
High Speed Multimedia Card Interface - HSMCI			
MCI_CK	Multimedia Card Clock	I/O	–
MCI_CDA	Multimedia Card Slot Command	I/O	–
MCI_DA0–MCI_DA7	Multimedia Card Slot Data	I/O	–
Universal Synchronous Asynchronous Receiver Transmitter - USARTx			
SCKx	USARTx Serial Clock	I/O	–
TXDx	USARTx Transmit Data	Output	–
RXDx	USARTx Receive Data	Input	–
RTSx	USARTx Request To Send	Output	–
CTSx	USARTx Clear To Send	Input	–

© 2017 Microchip Technology Inc.

Table 2-1: Signal Description List (Continued)

Signal Name	Function	Type	Active Level
Universal Asynchronous Receiver Transmitter - UARTx			
UTXDx	UARTx Transmit Data	Output	–
URXDx	UARTx Receive Data	Input	–
Synchronous Serial Controller - SSC			
TD	SSC Transmit Data	Output	–
RD	SSC Receive Data	Input	–
TK	SSC Transmit Clock	I/O	–
RK	SSC Receive Clock	I/O	–
TF	SSC Transmit Frame Sync	I/O	–
RF	SSC Receive Frame Sync	I/O	–
Timer Counter - TCx (x=0..5)			
TCLKx	TC Channel x External Clock Input	Input	–
TIOAx	TC Channel x I/O Line A	I/O	–
TIOBx	TC Channel x I/O Line B	I/O	–
Serial Peripheral Interface - SPIx			
SPIx_MISO	Master In Slave Out	I/O	–
SPIx_MOSI	Master Out Slave In	I/O	–
SPIx_SPCK	SPI Serial Clock	I/O	–
SPIx_NPCS0	SPI Peripheral Chip Select 0	I/O	Low
SPIx_NPCS1–SPIx_NPCS3	SPI Peripheral Chip Select	Output	Low
Two-wire Interface - TWIx			
TWDx	Two-wire Serial Data	I/O	–
TWCKx	Two-wire Serial Clock	I/O	–
Pulse Width Modulation Controller - PWM			
PWM0–PWM3	Pulse Width Modulation Output	Output	–
USB Device Full Speed Port - UDP			
DDP	USB Device Data +	Analog	–
DDM	USB Device Data -	Analog	–
USB Host Full Speed Port - UHP			
HDP	USB Host Data +	Analog	–
HDM	USB Host Data -	Analog	–

© 2017 Microchip Technology Inc.

Table 2-1: Signal Description List (Continued)

Signal Name	Function	Type	Active Level
LCD Controller - LCDC			
LCDDAT 0–23	LCD Data Bus	Output	–
LCDVSYNC	LCD Vertical Synchronization	Output	–
LCDHSYNC	LCD Horizontal Synchronization	Output	–
LCDPCK	LCD Pixel Clock	Output	–
LCDDEN	LCD Data Enable	Output	–
LCDPWM	LCD Contrast Control	Output	–
LCDDISP	LCD Display Enable	Output	–
Analog-to-Digital Converter - ADC			
AD0/XP/UL	Top/Upper Left Channel	Analog	–
AD1/XM/UR	Bottom/Upper Right Channel	Analog	–
AD2/YP/LL	Right/Lower Left Channel	Analog	–
AD3/YM/SENSE	Left/Sense Channel	Analog	–
AD4/LR	Lower Right Channel	Analog	–
AD5–AD11	7 Analog Inputs	Analog	–
ADTRG	ADC Trigger	Input	–
ADVREF	ADC Reference	Analog	–

Table 2-2: SAM9N12/CN11/CN12 I/O Type Description

I/O Type	Signal Name	Voltage Range	Analog	Pull-up[1]	Pull-up Value (Ohm)	Pull-down[1]	Pull-down Value (Ohm)	Schmitt Trigger[1]
GPIO	All PIO lines except the lines indicated further on in this table	1.65–3.6V	–	Switchable	50–100K	Switchable	50–100K	Switchable
GPIO_CLK	MCICK, SPI0SPCK, SPI1SPCK	1.65–3.6V	–	Switchable	50–100K	Switchable	50–100K	Switchable
GPIO_CLK2	LCDDOTCK	1.65–3.6V	–	Switchable	50–100K	Switchable	50–100K	Switchable
GPIO_ANA	ADx, GPADx	3.0–3.6V	I	Switchable	50–100K			Switchable
EBI	All data lines (input/output) except the lines indicated further on in this table	1.65–1.95V, 3.0–3.6V	–	Switchable	50–100K	Switchable	50–100K	–
EBI_O	All address and control lines (output only) except the lines indicated further on in this table	1.65–1.95V, 3.0–3.6V	–	Reset State	50–100K	Reset State	50–100K	–

© 2017 Microchip Technology Inc.

Table 2-2: SAM9N12/CN11/CN12 I/O Type Description (Continued)

I/O Type	Signal Name	Voltage Range	Analog	Pull-up[1]	Pull-up Value (Ohm)	Pull-down[1]	Pull-down Value (Ohm)	Schmitt Trigger[1]
EBI_CLK	SDCK, #SDCK	1.65–1.95V, 3.0–3.6V	–	–	–	–	–	–
RSTJTAG	NRST, NTRST, BMS, TCK, TDI, TMS, TDO, RTCK	3.0–3.6V	–	Reset State	100K	Reset State	100K	Reset State
SYSC	WKUP, SHDN, JTAGSEL	1.65–3.6V	–	Reset State	100k	Reset State	15K	Reset State
USBFS	HDP, HDM, DDP, DDM	3.0–3.6V	I/O	–	–	–	–	–
CLOCK	XIN, XOUT, XIN32, XOUT32	1.65–3.6V	I/O	–	–	–	–	–

Note 1: When “Reset State” is stated, the configuration is defined by the “Reset State” column of the Pin Description table.

© 2017 Microchip Technology Inc.

3. Package and Pinout

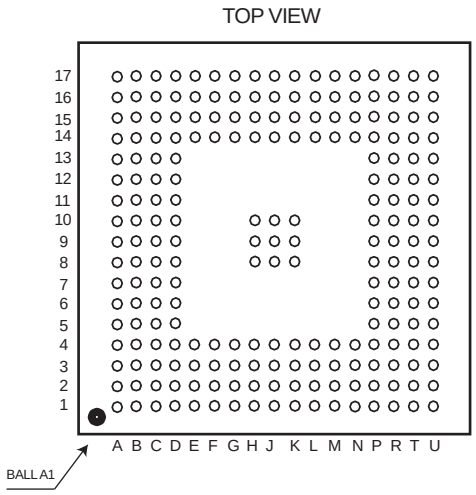
The SAM9N12/SAM9CN11/SAM9CN12 is available in the following Green-compliant packages:

- 217-ball BGA, pitch 0.8 mm
- 247-ball BGA, pitch 0.5 mm

3.1 217-ball BGA Package Outline

Figure 3-1 shows the orientation of the 217-ball BGA package.

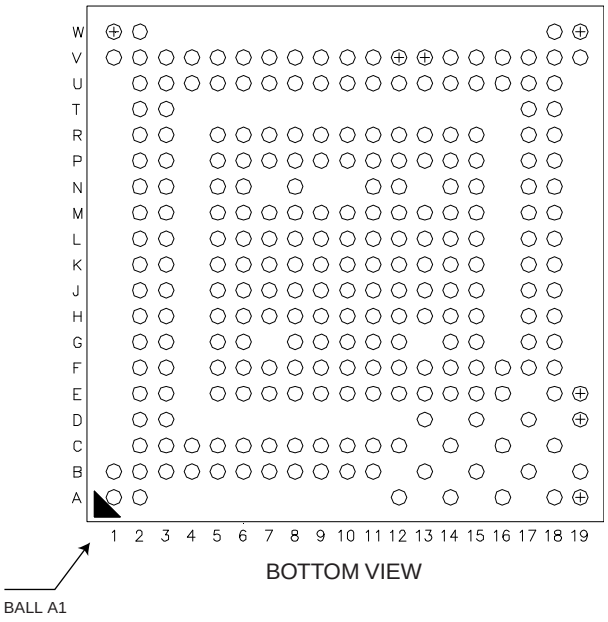
Figure 3-1: Orientation of the 217-ball BGA Package



3.2 247-ball BGA Package Outline

Figure 3-2 shows the orientation of the 247-ball BGA package.

Figure 3-2: Orientation of the 247-ball BGA Package



3.3 217-ball BGA Package Pinout

Table 3-1: BGA217 Pin Description

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
T3	VDDIOP0	GPIO	PA0	I/O	–	–	TXD0	O	SPI1_NPCS1	O	–	–	PIO, I, PU, ST
U2	VDDIOP0	GPIO	PA1	I/O	–	–	RXD0	I	SPI0_NPCS2	O	–	–	PIO, I, PU, ST
U3	VDDIOP0	GPIO	PA2	I/O	–	–	RTS0	O	–	–	–	–	PIO, I, PU, ST
P4	VDDIOP0	GPIO	PA3	I/O	–	–	CTS0	I	–	–	–	–	PIO, I, PU, ST
T4	VDDIOP0	GPIO	PA4	I/O	–	–	SCK0	I/O	–	–	–	–	PIO, I, PU, ST
U4	VDDIOP0	GPIO	PA5	I/O	–	–	TXD1	O	–	–	–	–	PIO, I, PU, ST
P5	VDDIOP0	GPIO	PA6	I/O	–	–	RXD1	I	–	–	–	–	PIO, I, PU, ST
R4	VDDIOP0	GPIO	PA7	I/O	–	–	TXD2	O	SPI0_NPCS1	O	–	–	PIO, I, PU, ST
U6	VDDIOP0	GPIO	PA8	I/O	–	–	RXD2	I	SPI1_NPCS0	I/O	–	–	PIO, I, PU, ST
R5	VDDIOP0	GPIO	PA9	I/O	–	–	DRXD	I	–	–	–	–	PIO, I, PU, ST
R6	VDDIOP0	GPIO	PA10	I/O	–	–	DTXD	O	–	–	–	–	PIO, I, PU, ST
T5	VDDIOP0	GPIO	PA11	I/O	–	–	SPI0_MISO	I/O	MCDA4	I/O	–	–	PIO, I, PU, ST
T6	VDDIOP0	GPIO	PA12	I/O	–	–	SPI0_MOSI	I/O	MCDA5	I/O	–	–	PIO, I, PU, ST
U5	VDDIOP0	GPIO_CLK	PA13	I/O	–	–	SPI0_SPCK	I/O	MCDA6	I/O	–	–	PIO, I, PU, ST
U7	VDDIOP0	GPIO	PA14	I/O	–	–	SPI0_NPCS0	I/O	MCDA7	I/O	–	–	PIO, I, PU, ST
T7	VDDIOP0	GPIO	PA15	I/O	–	–	MCDA0	I/O	–	–	–	–	PIO, I, PU, ST
R7	VDDIOP0	GPIO	PA16	I/O	–	–	MCCDA	I/O	–	–	–	–	PIO, I, PU, ST
U8	VDDIOP0	GPIO_CLK	PA17	I/O	–	–	MCCK	I/O	–	–	–	–	PIO, I, PU, ST
P8	VDDIOP0	GPIO	PA18	I/O	–	–	MCDA1	I/O	–	–	–	–	PIO, I, PU, ST
T8	VDDIOP0	GPIO	PA19	I/O	–	–	MCDA2	I/O	–	–	–	–	PIO, I, PU, ST
R8	VDDIOP0	GPIO	PA20	I/O	–	–	MCDA3	I/O	–	–	–	–	PIO, I, PU, ST
U9	VDDIOP0	GPIO	PA21	I/O	–	–	TIOA0	I/O	SPI1_MISO	I/O	–	–	PIO, I, PU, ST
U10	VDDIOP0	GPIO	PA22	I/O	–	–	TIOA1	I/O	SPI1_MOSI	I/O	–	–	PIO, I, PU, ST
T9	VDDIOP0	GPIO_CLK	PA23	I/O	–	–	TIOA2	I/O	SPI1_SPCK	I/O	–	–	PIO, I, PU, ST
U11	VDDIOP0	GPIO	PA24	I/O	–	–	TCLK0	I	TK	I/O	–	–	PIO, I, PU, ST
T10	VDDIOP0	GPIO	PA25	I/O	–	–	TCLK1	I	TF	I/O	–	–	PIO, I, PU, ST
R9	VDDIOP0	GPIO	PA26	I/O	–	–	TCLK2	I	TD	O	–	–	PIO, I, PU, ST
U12	VDDIOP0	GPIO	PA27	I/O	–	–	TIOB0	I/O	RD	I	–	–	PIO, I, PU, ST
T11	VDDIOP0	GPIO	PA28	I/O	–	–	TIOB1	I/O	RK	I/O	–	–	PIO, I, PU, ST
U13	VDDIOP0	GPIO	PA29	I/O	–	–	TIOB2	I/O	RF	I/O	–	–	PIO, I, PU, ST
R10	VDDIOP0	GPIO	PA30	I/O	–	–	TWD0	I/O	SPI1_NPCS3	O	–	–	PIO, I, PU, ST
T12	VDDIOP0	GPIO	PA31	I/O	–	–	TWCK0	O	SPI1_NPCS2	O	–	–	PIO, I, PU, ST
E4	VDDANA	GPIO	PB0	I/O	–	–	–	–	RTS2	O	–	–	PIO, I, PU, ST
F3	VDDANA	GPIO	PB1	I/O	–	–	–	–	CTS2	I	–	–	PIO, I, PU, ST
F4	VDDANA	GPIO	PB2	I/O	–	–	–	–	SCK2	I/O	–	–	PIO, I, PU, ST
F2	VDDANA	GPIO	PB3	I/O	–	–	–	–	SPI0_NPCS3	O	–	–	PIO, I, PU, ST
G4	VDDANA	GPIO_CLK	PB4	I/O	–	–	–	–	–	–	–	–	PIO, I, PU, ST
G3	VDDANA	GPIO	PB5	I/O	–	–	–	–	–	–	–	–	PIO, I, PU, ST

© 2017 Microchip Technology Inc.

Table 3-1: BGA217 Pin Description (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
D2	VDDANA	GPIO_ANA	PB6	I/O	AD7	I	–	–	–	–	–	–	PIO, I, PU, ST
E2	VDDANA	GPIO_ANA	PB7	I/O	AD8	I	–	–	–	–	–	–	PIO, I, PU, ST
D1	VDDANA	GPIO_ANA	PB8	I/O	AD9	I	–	–	–	–	–	–	PIO, I, PU, ST
F1	VDDANA	GPIO_ANA	PB9	I/O	AD10	I	–	–	PCK1	O	–	–	PIO, I, PU, ST
E1	VDDANA	GPIO_ANA	PB10	I/O	AD11	I	–	–	PCK0	O	–	–	PIO, I, PU, ST
A1	VDDANA	GPIO_ANA	PB11	I/O	AD0	I	–	–	PWM0	O	–	–	PIO, I, PU, ST
C3	VDDANA	GPIO_ANA	PB12	I/O	AD1	I	–	–	PWM1	O	–	–	PIO, I, PU, ST
B1	VDDANA	GPIO_ANA	PB13	I/O	AD2	I	–	–	PWM2	O	–	–	PIO, I, PU, ST
C2	VDDANA	GPIO_ANA	PB14	I/O	AD3	I	–	–	PWM3	O	–	–	PIO, I, PU, ST
D3	VDDANA	GPIO_ANA	PB15	I/O	AD4	I	–	–	–	–	–	–	PIO, I, PU, ST
C1	VDDANA	GPIO_ANA	PB16	I/O	AD5	I	–	–	–	–	–	–	PIO, I, PU, ST
E3	VDDANA	GPIO_ANA	PB17	I/O	AD6	I	–	–	–	–	–	–	PIO, I, PU, ST
D4	VDDANA	GPIO	PB18	I/O	–	–	IRQ	I	ADTRG	I	–	–	PIO, I, PU, ST
G2	VDDIOP1	GPIO	PC0	I/O	–	–	LCDDAT0	O	–	–	TWD1	I/O	PIO, I, PU, ST
G1	VDDIOP1	GPIO	PC1	I/O	–	–	LCDDAT1	O	–	–	TWCK1	O	PIO, I, PU, ST
H4	VDDIOP1	GPIO	PC2	I/O	–	–	LCDDAT2	O	–	–	TIOA3	I/O	PIO, I, PU, ST
J1	VDDIOP1	GPIO	PC3	I/O	–	–	LCDDAT3	O	–	–	TIOB3	I/O	PIO, I, PU, ST
H3	VDDIOP1	GPIO	PC4	I/O	–	–	LCDDAT4	O	–	–	TCLK3	I	PIO, I, PU, ST
J3	VDDIOP1	GPIO	PC5	I/O	–	–	LCDDAT5	O	–	–	TIOA4	I/O	PIO, I, PU, ST
H2	VDDIOP1	GPIO	PC6	I/O	–	–	LCDDAT6	O	–	–	TIOB4	I/O	PIO, I, PU, ST
H1	VDDIOP1	GPIO	PC7	I/O	–	–	LCDDAT7	O	–	–	TCLK4	I	PIO, I, PU, ST
K2	VDDIOP1	GPIO	PC8	I/O	–	–	LCDDAT8	O	–	–	UTXD0	O	PIO, I, PU, ST
J2	VDDIOP1	GPIO	PC9	I/O	–	–	LCDDAT9	O	–	–	URXD0	I	PIO, I, PU, ST
L1	VDDIOP1	GPIO	PC10	I/O	–	–	LCDDAT10	O	–	–	PWM0	O	PIO, I, PU, ST
K1	VDDIOP1	GPIO	PC11	I/O	–	–	LCDDAT11	O	–	–	PWM1	O	PIO, I, PU, ST
L2	VDDIOP1	GPIO	PC12	I/O	–	–	LCDDAT12	O	–	–	TIOA5	I/O	PIO, I, PU, ST
K3	VDDIOP1	GPIO	PC13	I/O	–	–	LCDDAT13	O	–	–	TIOB5	I/O	PIO, I, PU, ST
M1	VDDIOP1	GPIO	PC14	I/O	–	–	LCDDAT14	O	–	–	TCLK5	I	PIO, I, PU, ST
M2	VDDIOP1	GPIO_CLK	PC15	I/O	–	–	LCDDAT15	O	–	–	PCK0	O	PIO, I, PU, ST
K4	VDDIOP1	GPIO	PC16	I/O	–	–	LCDDAT16	O	–	–	UTXD1	O	PIO, I, PU, ST
M3	VDDIOP1	GPIO	PC17	I/O	–	–	LCDDAT17	O	–	–	URXD1	I	PIO, I, PU, ST
N1	VDDIOP1	GPIO	PC18	I/O	–	–	LCDDAT18	O	–	–	PWM0	O	PIO, I, PU, ST
N2	VDDIOP1	GPIO	PC19	I/O	–	–	LCDDAT19	O	–	–	PWM1	O	PIO, I, PU, ST
N3	VDDIOP1	GPIO	PC20	I/O	–	–	LCDDAT20	O	–	–	PWM2	O	PIO, I, PU, ST
P1	VDDIOP1	GPIO	PC21	I/O	–	–	LCDDAT21	O	–	–	PWM3	O	PIO, I, PU, ST
P2	VDDIOP1	GPIO	PC22	I/O	–	–	LCDDAT22	O	TXD3	O	–	–	PIO, I, PU, ST
P3	VDDIOP1	GPIO	PC23	I/O	–	–	LCDDAT23	O	RXD3	I	–	–	PIO, I, PU, ST
R1	VDDIOP1	GPIO	PC24	I/O	–	–	LCDDISP	O	RTS3	O	–	–	PIO, I, PU, ST
R3	VDDIOP1	GPIO	PC25	I/O	–	–	–		CTS3	I	–	–	PIO, I, PU, ST
R2	VDDIOP1	GPIO	PC26	I/O	–	–	LCDPWM	O	SCK3	I/O			PIO, I, PU, ST

Table 3-1: BGA217 Pin Description (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
T1	VDDIOP1	GPIO	PC27	I/O	–	–	LCDVSYNC	O	–	–	RTS1	O	PIO, I, PU, ST
M4	VDDIOP1	GPIO	PC28	I/O	–	–	LCDHSYNC	O	–	–	CTS1	I	PIO, I, PU, ST
N4	VDDIOP1	GPIO_CLK	PC29	I/O	–	–	LCDDEN	O	–	–	SCK1	I/O	PIO, I, PU, ST
T2	VDDIOP1	GPIO_CLK2	PC30	I/O	–	–	LCDPCK	O	–	–	–	–	PIO, I, PU, ST
U1	VDDIOP1	GPIO	PC31	I/O	–	–	FIQ	I	–	–	PCK1	O	PIO, I, PU, ST
P15	VDDNF	EBI	PD0	I/O	–	–	NANDOE	O	–	–	–	–	PIO, I, PU
N14	VDDNF	EBI	PD1	I/O	–	–	NANDWE	O	–	–	–	–	PIO, I, PU
M15	VDDNF	EBI	PD2	I/O	–	–	A21/NANDALE	O	–	–	–	–	A21,O, PD
M14	VDDNF	EBI	PD3	I/O	–	–	A22/NANDCLE	O	–	–	–	–	A22,O, PD
P16	VDDNF	EBI	PD4	I/O	–	–	NCS3	O	–	–	–	–	PIO, I, PU
M17	VDDNF	EBI	PD5	I/O	–	–	NWAIT	I	–	–	–	–	PIO, I, PU
L15	VDDNF	EBI	PD6	I/O	–	–	D16	O	–	–	–	–	PIO, I, PU
L16	VDDNF	EBI	PD7	I/O	–	–	D17	O	–	–	–	–	PIO, I, PU
L17	VDDNF	EBI	PD8	I/O	–	–	D18	O	–	–	–	–	PIO, I, PU
K17	VDDNF	EBI	PD9	I/O	–	–	D19	O	–	–	–	–	PIO, I, PU
K16	VDDNF	EBI	PD10	I/O	–	–	D20	O	–	–	–	–	PIO, I, PU
K15	VDDNF	EBI	PD11	I/O	–	–	D21	O	–	–	–	–	PIO, I, PU
J17	VDDNF	EBI	PD12	I/O	–	–	D22	O	–	–	–	–	PIO, I, PU
J16	VDDNF	EBI	PD13	I/O	–	–	D23	O	–	–	–	–	PIO, I, PU
H17	VDDNF	EBI	PD14	I/O	–	–	D24	O	–	–	–	–	PIO, I, PU
J15	VDDNF	EBI	PD15	I/O	–	–	D25	O	A20	O	–	–	A20, O, PD
G17	VDDNF	EBI	PD16	I/O	–	–	D26	O	A23	O	–	–	A23, O, PD
H16	VDDNF	EBI	PD17	I/O	–	–	D27	O	A24	O	–	–	A24, O, PD
H15	VDDNF	EBI	PD18	I/O	–	–	D28	O	A25	O	–	–	A25, O, PD
F17	VDDNF	EBI	PD19	I/O	–	–	D29	O	NCS2	O	–	–	PIO, I, PU
G16	VDDNF	EBI	PD20	I/O	–	–	D30	O	NCS4	O	–	–	PIO, I, PU
E17	VDDNF	EBI	PD21	I/O	–	–	D31	O	NCS5	O	–	–	PIO, I, PU
H8 H9 H10	VDDIOM	POWER	VDDIOM	I	–	–	–	–	–	–	–	–	I
J14 K14 L14	VDDNF	POWER	VDDNF	I	–	–	–	–	–	–	–	–	I
J8 J9 J10 K9 K10	GNDIOM	GND	GNDIOM	I	–	–	–	–	–	–	–	–	I
P9 P12	VDDIOP0	POWER	VDDIOP0	I	–	–	–	–	–	–	–	–	I
L3 L4	VDDIOP1	POWER	VDDIOP1	I	–	–	–	–	–	–	–	–	I
P6 P7 P13	GNDIOP	GND	GNDIOP	I	–	–	–	–	–	–	–	–	I

© 2017 Microchip Technology Inc.

Table 3-1: BGA217 Pin Description (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
D6	VDDBU	POWER	VDDBU	I	–	–	–	–	–	–	–	–	I
D5 B3	GNDBU	GND	GNDBU	I	–	–	–	–	–	–	–	–	I
C4	VDDANA	POWER	VDDANA	I	–	–	–	–	–	–	–	–	I
B2	GNDANA	GND	GNDANA	I	–	–	–	–	–	–	–	–	I
T16	VDDPLL	POWER	VDDPLL	I	–	–	–	–	–	–	–	–	I
P14	GNDPLL	GND	GNDPLL	I	–	–	–	–	–	–	–	–	I
R14	VDDOSC	POWER	VDDOSC	I	–	–	–	–	–	–	–	–	I
R15	VDDUSB	POWER	VDDUSB	I	–	–	–	–	–	–	–	–	I
N16	VDDFUSE	POWER	VDDFUSE	I	–	–	–	–	–	–	–	–	I
M16	GNDFUSE	GND	GNDFUSE	I	–	–	–	–	–	–	–	–	I
T17	GNDUSB	GND	GNDUSB	I	–	–	–	–	–	–	–	–	I
C8 G15 J4 P10	VDDCORE	POWER	VDDCORE	I	–	–	–	–	–	–	–	–	I
D8 H14 K8 P11	GNDCORE	GND	GNDCORE	I	–	–	–	–	–	–	–	–	I
B14	VDDIOM	EBI	D0	I/O	–	–	–	–	–	–	–	–	O, PD
A14	VDDIOM	EBI	D1	I/O	–	–	–	–	–	–	–	–	O, PD
C14	VDDIOM	EBI	D2	I/O	–	–	–	–	–	–	–	–	O, PD
D13	VDDIOM	EBI	D3	I/O	–	–	–	–	–	–	–	–	O, PD
C13	VDDIOM	EBI	D4	I/O	–	–	–	–	–	–	–	–	O, PD
B13	VDDIOM	EBI	D5	I/O	–	–	–	–	–	–	–	–	O, PD
A13	VDDIOM	EBI	D6	I/O	–	–	–	–	–	–	–	–	O, PD
C12	VDDIOM	EBI	D7	I/O	–	–	–	–	–	–	–	–	O, PD
D12	VDDIOM	EBI	D8	I/O	–	–	–	–	–	–	–	–	O, PD
B12	VDDIOM	EBI	D9	I/O	–	–	–	–	–	–	–	–	O, PD
C11	VDDIOM	EBI	D10	I/O	–	–	–	–	–	–	–	–	O, PD
D11	VDDIOM	EBI	D11	I/O	–	–	–	–	–	–	–	–	O, PD
A12	VDDIOM	EBI	D12	I/O	–	–	–	–	–	–	–	–	O, PD
B11	VDDIOM	EBI	D13	I/O	–	–	–	–	–	–	–	–	O, PD
A11	VDDIOM	EBI	D14	I/O	–	–	–	–	–	–	–	–	O, PD
C10	VDDIOM	EBI	D15	I/O	–	–	–	–	–	–	–	–	O, PD
D17	VDDIOM	EBI_O	A0	O	NBS0	O	–	–	–	–	–	–	O, PD
C17	VDDIOM	EBI_O	A1	O	NBS2/ DQM2/ NWR2	O	–	–	–	–	–	–	O, PD
F16	VDDIOM	EBI_O	A2	O	–	–	–	–	–	–	–	–	O, PD
B17	VDDIOM	EBI_O	A3	O	–	–	–	–	–	–	–	–	O, PD
A17	VDDIOM	EBI_O	A4	O	–	–	–	–	–	–	–	–	O, PD

Table 3-1: BGA217 Pin Description (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
F15	VDDIOM	EBI_O	A5	O	–	–	–	–	–	–	–	–	O, PD
E16	VDDIOM	EBI_O	A6	O	–	–	–	–	–	–	–	–	O, PD
D16	VDDIOM	EBI_O	A7	O	–	–	–	–	–	–	–	–	O, PD
E15	VDDIOM	EBI_O	A8	O	–	–	–	–	–	–	–	–	O, PD
G14	VDDIOM	EBI_O	A9	O	–	–	–	–	–	–	–	–	O, PD
C16	VDDIOM	EBI_O	A10	O	–	–	–	–	–	–	–	–	O, PD
F14	VDDIOM	EBI_O	A11	O	–	–	–	–	–	–	–	–	O, PD
B16	VDDIOM	EBI_O	A12	O	–	–	–	–	–	–	–	–	O, PD
A16	VDDIOM	EBI_O	A13	O	–	–	–	–	–	–	–	–	O, PD
C15	VDDIOM	EBI_O	A14	O	–	–	–	–	–	–	–	–	O, PD
D15	VDDIOM	EBI_O	A15	O	–	–	–	–	–	–	–	–	O, PD
B15	VDDIOM	EBI_O	A16	O	BA0	O	–	–	–	–	–	–	O, PD
E14	VDDIOM	EBI_O	A17	O	BA1	O	–	–	–	–	–	–	O, PD
A15	VDDIOM	EBI_O	A18	O	BA2	O	–	–	–	–	–	–	O, PD
D14	VDDIOM	EBI_O	A19	O	–	–	–	–	–	–	–	–	O, PD
B7	VDDIOM	EBI_O	NCS0	O	–	–	–	–	–	–	–	–	O, PU
C5	VDDIOM	EBI_O	NCS1	O	SDCS	O	–	–	–	–	–	–	O, PU
C7	VDDIOM	EBI_O	NRD	O	–	–	–	–	–	–	–	–	O, PU
A6	VDDIOM	EBI_O	NWR0	O	NWRE	O	–	–	–	–	–	–	O, PU
C6	VDDIOM	EBI_O	NWR1	O	NBS1	O	–	–	–	–	–	–	O, PU
D7	VDDIOM	EBI_O	NWR3	O	NBS3/ DQM3	O	–	–	–	–	–	–	O, PU
A10	VDDIOM	EBI_CLK	SDCK	O	–	–	–	–	–	–	–	–	O
A9	VDDIOM	EBI_CLK	#SDCK	O	–	–	–	–	–	–	–	–	O
D10	VDDIOM	EBI_O	SDCKE	O	–	–	–	–	–	–	–	–	O, PU
B9	VDDIOM	EBI_O	RAS	O	–	–	–	–	–	–	–	–	O, PU
D9	VDDIOM	EBI_O	CAS	O	–	–	–	–	–	–	–	–	O, PU
B10	VDDIOM	EBI_O	SDWE	O	–	–	–	–	–	–	–	–	O, PU
B6	VDDIOM	EBI_O	SDA10	O	–	–	–	–	–	–	–	–	O, PU
C9	VDDIOM	EBI_O	DQM0	O	–	–	–	–	–	–	–	–	O, PU
A8	VDDIOM	EBI_O	DQM1	O	–	–	–	–	–	–	–	–	O, PU
B8	VDDIOM	EBI	DQS0	I/O	–	–	–	–	–	–	–	–	O, PD
A7	VDDIOM	EBI	DQS1	I/O	–	–	–	–	–	–	–	–	O, PD
A2	VDDANA	POWER	ADVREF	I	–	–	–	–	–	–	–	–	I
P17	VDDUSB	USBFS	HDP	I/O	–	–	–	–	–	–	–	–	O, PD
N17	VDDUSB	USBFS	HDM	I/O	–	–	–	–	–	–	–	–	O, PD
R17	VDDUSB	USBFS	DDP	I/O	–	–	–	–	–	–	–	–	O, PD
R16	VDDUSB	USBFS	DDM	I/O	–	–	–	–	–	–	–	–	O, PD
A5	VDDBU	SYSC	WKUP	I	–	–	–	–	–	–	–	–	I, ST
B5	VDDBU	SYSC	SHDN	O	–	–	–	–	–	–	–	–	O, PU

© 2017 Microchip Technology Inc.

Table 3-1: BGA217 Pin Description (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
U15	VDDIOP0	RSTJTAG	BMS	I	–	–	–	–	–	–	–	–	I, PU, ST
B4	VDDBU	SYSC	JTAGSEL	I	–	–	–	–	–	–	–	–	I, PD
R12	VDDIOP0	RSTJTAG	TCK	I	–	–	–	–	–	–	–	–	I, ST
R11	VDDIOP0	RSTJTAG	TDI	I	–	–	–	–	–	–	–	–	I, ST
U14	VDDIOP0	RSTJTAG	TDO	O	–	–	–	–	–	–	–	–	O
T13	VDDIOP0	RSTJTAG	TMS	I	–	–	–	–	–	–	–	–	I, ST
T14	VDDIOP0	RSTJTAG	RTCK	O	–	–	–	–	–	–	–	–	O
R13	VDDIOP0	RSTJTAG	NRST	I/O	–	–	–	–	–	–	–	–	I, PU, ST
T15	VDDIOP0	RSTJTAG	NTRST	I	–	–	–	–	–	–	–	–	I, PU, ST
A4	VDDBU	CLOCK	XIN32	I	–	–	–	–	–	–	–	–	I
A3	VDDBU	CLOCK	XOUT32	O	–	–	–	–	–	–	–	–	O
U17	VDDIOP0	CLOCK	XIN	I	–	–	–	–	–	–	–	–	I
U16	VDDIOP0	CLOCK	XOUT	O	–	–	–	–	–	–	–	–	O
N15	NC	–	–	–	–	–	–	–	–	–	–	–	–

3.4 247-ball BGA Package Pinout

Table 3-2: BGA247 Pin Description

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
P3	VDDIOP0	GPIO	PA0	I/O	–	–	TXD0	O	SPI1_NPCS1	O	–	–	PIO, I, PU, ST
R2	VDDIOP0	GPIO	PA1	I/O	–	–	RXD0	I	SPI0_NPCS2	O	–	–	PIO, I, PU, ST
R9	VDDIOP0	GPIO	PA2	I/O	–	–	RTS0	O	–	–	–	–	PIO, I, PU, ST
N5	VDDIOP0	GPIO	PA3	I/O	–	–	CTS0	I	–	–	–	–	PIO, I, PU, ST
P10	VDDIOP0	GPIO	PA4	I/O	–	–	SCK0	I/O	–	–	–	–	PIO, I, PU, ST
R3	VDDIOP0	GPIO	PA5	I/O	–	–	TXD1	O	–	–	–	–	PIO, I, PU, ST
R10	VDDIOP0	GPIO	PA6	I/O	–	–	RXD1	I	–	–	–	–	PIO, I, PU, ST
T2	VDDIOP0	GPIO	PA7	I/O	–	–	TXD2	O	SPI0_NPCS1	O	–	–	PIO, I, PU, ST
P6	VDDIOP0	GPIO	PA8	I/O	–	–	RXD2	I	SPI1_NPCS0	I/O	–	–	PIO, I, PU, ST
T3	VDDIOP0	GPIO	PA9	I/O	–	–	DRXD	I	–	–	–	–	PIO, I, PU, ST
U2	VDDIOP0	GPIO	PA10	I/O	–	–	DTXD	O	–	–	–	–	PIO, I, PU, ST
P5	VDDIOP0	GPIO	PA11	I/O	–	–	SPI0_MISO	I/O	MCDA4	I/O	–	–	PIO, I, PU, ST
V2	VDDIOP0	GPIO	PA12	I/O	–	–	SPI0_MOSI	I/O	MCDA5	I/O	–	–	PIO, I, PU, ST
V1	VDDIOP0	GPIO_CLK	PA13	I/O	–	–	SPI0_SPCK	I/O	MCDA6	I/O	–	–	PIO, I, PU, ST
W2	VDDIOP0	GPIO	PA14	I/O	–	–	SPI0_NPCS0	I/O	MCDA7	I/O	–	–	PIO, I, PU, ST
W1	VDDIOP0	GPIO	PA15	I/O	–	–	MCDA0	I/O	–	–	–	–	PIO, I, PU, ST
V3	VDDIOP0	GPIO	PA16	I/O	–	–	MCCDA	I/O	–	–	–	–	PIO, I, PU, ST
R5	VDDIOP0	GPIO_CLK	PA17	I/O	–	–	MCCK	I/O	–	–	–	–	PIO, I, PU, ST
U3	VDDIOP0	GPIO	PA18	I/O	–	–	MCDA1	I/O	–	–	–	–	PIO, I, PU, ST
V4	VDDIOP0	GPIO	PA19	I/O	–	–	MCDA2	I/O	–	–	–	–	PIO, I, PU, ST
U4	VDDIOP0	GPIO	PA20	I/O	–	–	MCDA3	I/O	–	–	–	–	PIO, I, PU, ST
V5	VDDIOP0	GPIO	PA21	I/O	–	–	TIOA0	I/O	SPI1_MISO	I/O	–	–	PIO, I, PU, ST
U5	VDDIOP0	GPIO	PA22	I/O	–	–	TIOA1	I/O	SPI1_MOSI	I/O	–	–	PIO, I, PU, ST
R6	VDDIOP0	GPIO_CLK	PA23	I/O	–	–	TIOA2	I/O	SPI1_SPCK	I/O	–	–	PIO, I, PU, ST
R7	VDDIOP0	GPIO	PA24	I/O	–	–	TCLK0	I	TK	I/O	–	–	PIO, I, PU, ST
U6	VDDIOP0	GPIO	PA25	I/O	–	–	TCLK1	I	TF	I/O	–	–	PIO, I, PU, ST
V6	VDDIOP0	GPIO	PA26	I/O	–	–	TCLK2	I	TD	O	–	–	PIO, I, PU, ST
R8	VDDIOP0	GPIO	PA27	I/O	–	–	TIOB0	I/O	RD	I	–	–	PIO, I, PU, ST
U7	VDDIOP0	GPIO	PA28	I/O	–	–	TIOB1	I/O	RK	I/O	–	–	PIO, I, PU, ST
P11	VDDIOP0	GPIO	PA29	I/O	–	–	TIOB2	I/O	RF	I/O	–	–	PIO, I, PU, ST
V7	VDDIOP0	GPIO	PA30	I/O	–	–	TWD0	I/O	SPI1_NPCS3	O	–	–	PIO, I, PU, ST
N12	VDDIOP0	GPIO	PA31	I/O	–	–	TWCK0	O	SPI1_NPCS2	O	–	–	PIO, I, PU, ST
G6	VDDANA	GPIO	PB0	I/O	–	–	–	–	RTS2	O	–	–	PIO, I, PU, ST
E3	VDDANA	GPIO	PB1	I/O	–	–	–	–	CTS2	I	–	–	PIO, I, PU, ST
G5	VDDANA	GPIO	PB2	I/O	–	–	–	–	SCK2	I/O	–	–	PIO, I, PU, ST
F2	VDDANA	GPIO	PB3	I/O	–	–	–	–	SPI0_NPCS3	O	–	–	PIO, I, PU, ST
E2	VDDANA	GPIO_CLK	PB4	I/O	–	–	–	–	–	–	–	–	PIO, I, PU, ST
E5	VDDANA	GPIO	PB5	I/O			–	–	–	–	–	–	PIO, I, PU, ST

© 2017 Microchip Technology Inc.

Table 3-2: BGA247 Pin Description (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
C2	VDDANA	GPIO_ANA	PB6	I/O	AD7	I	–	–	–	–	–	–	PIO, I, PU, ST
B2	VDDANA	GPIO_ANA	PB7	I/O	AD8	I	–	–	–	–	–	–	PIO, I, PU, ST
A2	VDDANA	GPIO_ANA	PB8	I/O	AD9	I	–	–	–	–	–	–	PIO, I, PU, ST
B1	VDDANA	GPIO_ANA	PB9	I/O	AD10	I	–	–	PCK1	O	–	–	PIO, I, PU, ST
A1	VDDANA	GPIO_ANA	PB10	I/O	AD11	I	–	–	PCK0	O	–	–	PIO, I, PU, ST
C7	VDDANA	GPIO_ANA	PB11	I/O	AD0	I	–	–	PWM0	O	–	–	PIO, I, PU, ST
C8	VDDANA	GPIO_ANA	PB12	I/O	AD1	I	–	–	PWM1	O	–	–	PIO, I, PU, ST
D3	VDDANA	GPIO_ANA	PB13	I/O	AD2	I	–	–	PWM2	O	–	–	PIO, I, PU, ST
F5	VDDANA	GPIO_ANA	PB14	I/O	AD3	I	–	–	PWM3	O	–	–	PIO, I, PU, ST
E6	VDDANA	GPIO_ANA	PB15	I/O	AD4	I	–	–	–	–	–	–	PIO, I, PU, ST
C9	VDDANA	GPIO_ANA	PB16	I/O	AD5	I	–	–	–	I	–	–	PIO, I, PU, ST
D2	VDDANA	GPIO_ANA	PB17	I/O	AD6	I	–	–	–	I	–	–	PIO, I, PU, ST
E7	VDDANA	GPIO	PB18	I/O	–	–	IRQ	I	ADTRG	I	–	–	PIO, I, PU, ST
F3	VDDIOP1	GPIO	PC0	I/O	–	–	LCDDAT0	O	–	–	TWD1	I/O	PIO, I, PU, ST
G2	VDDIOP1	GPIO	PC1	I/O	–	–	LCDDAT1	O	–	–	TWCK1	O	PIO, I, PU, ST
L7	VDDIOP1	GPIO	PC2	I/O	–	–	LCDDAT2	O	–	–	TIOA3	I/O	PIO, I, PU, ST
G3	VDDIOP1	GPIO	PC3	I/O	–	–	LCDDAT3	O	–	–	TIOB3	I/O	PIO, I, PU, ST
H5	VDDIOP1	GPIO	PC4	I/O	–	–	LCDDAT4	O	–	–	TCLK3	I	PIO, I, PU, ST
M7	VDDIOP1	GPIO	PC5	I/O	–	–	LCDDAT5	O	–	–	TIOA4	I/O	PIO, I, PU, ST
H3	VDDIOP1	GPIO	PC6	I/O	–	–	LCDDAT6	O	–	–	TIOB4	I/O	PIO, I, PU, ST
H2	VDDIOP1	GPIO	PC7	I/O	–	–	LCDDAT7	O	–	–	TCLK4	I	PIO, I, PU, ST
J3	VDDIOP1	GPIO	PC8	I/O	–	–	LCDDAT8	O	–	–	UTXD0	O	PIO, I, PU, ST
M8	VDDIOP1	GPIO	PC9	I/O	–	–	LCDDAT9	O	–	–	URXD0	I	PIO, I, PU, ST
J5	VDDIOP1	GPIO	PC10	I/O	–	–	LCDDAT10	O	–	–	PWM0	O	PIO, I, PU, ST
K6	VDDIOP1	GPIO	PC11	I/O	–	–	LCDDAT11	O	–	–	PWM1	O	PIO, I, PU, ST
P9	VDDIOP1	GPIO	PC12	I/O	–	–	LCDDAT12	O	–	–	TIOA5	I/O	PIO, I, PU, ST
L6	VDDIOP1	GPIO	PC13	I/O	–	–	LCDDAT13	O	–	–	TIOB5	I/O	PIO, I, PU, ST
J2	VDDIOP1	GPIO	PC14	I/O	–	–	LCDDAT14	O	–	–	TCLK5	I	PIO, I, PU, ST
K3	VDDIOP1	GPIO_CLK	PC15	I/O	–	–	LCDDAT15	O	–	–	PCK0	O	PIO, I, PU, ST
K2	VDDIOP1	GPIO	PC16	I/O	–	–	LCDDAT16	O	–	–	UTXD1	O	PIO, I, PU, ST
K5	VDDIOP1	GPIO	PC17	I/O	–	–	LCDDAT17	O	–	–	URXD1	I	PIO, I, PU, ST
L3	VDDIOP1	GPIO	PC18	I/O	–	–	LCDDAT18	O	–	–	PWM0	O	PIO, I, PU, ST
N8	VDDIOP1	GPIO	PC19	I/O	–	–	LCDDAT19	O	–	–	PWM1	O	PIO, I, PU, ST
L2	VDDIOP1	GPIO	PC20	I/O	–	–	LCDDAT20	O	–	–	PWM2	O	PIO, I, PU, ST
P8	VDDIOP1	GPIO	PC21	I/O	–	–	LCDDAT21	O	–	–	PWM3	O	PIO, I, PU, ST
M3	VDDIOP1	GPIO	PC22	I/O	–	–	LCDDAT22	O	TXD3	O	–	–	PIO, I, PU, ST
L5	VDDIOP1	GPIO	PC23	I/O	–	–	LCDDAT23	O	RXD3	I	–	–	PIO, I, PU, ST
N6	VDDIOP1	GPIO	PC24	I/O	–	–	LCDDISP	O	RTS3	O	–	–	PIO, I, PU, ST
N2	VDDIOP1	GPIO	PC25	I/O	–	–	–	–	CTS3	I	–	–	PIO, I, PU, ST
P7	VDDIOP1	GPIO	PC26	I/O	–	–	LCDPWM	O	SCK3	I/O	–	–	PIO, I, PU, ST

Table 3-2: BGA247 Pin Description (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
M2	VDDIOP1	GPIO	PC27	I/O	–	–	LCDVSYNC	O	–	–	RTS1	O	PIO, I, PU, ST
M5	VDDIOP1	GPIO	PC28	I/O	–	–	LCDHSYNC	O	–	–	CTS1	I	PIO, I, PU, ST
N3	VDDIOP1	GPIO_CLK	PC29	I/O	–	–	LCDDEN	O	–	–	SCK1	I/O	PIO, I, PU, ST
M6	VDDIOP1	GPIO_CLK2	PC30	I/O	–	–	LCDPCK	O	–	–	–	–	PIO, I, PU, ST
P2	VDDIOP1	GPIO	PC31	I/O	–	–	FIQ	I	–	–	PCK1	O	PIO, I, PU, ST
R14	VDDNF	EBI	PD0	I/O	–	–	NANDOE	O	–	–	–	–	PIO, I, PU
R15	VDDNF	EBI	PD1	I/O	–	–	NANDWE	O	–	–	–	–	PIO, I, PU
T17	VDDNF	EBI	PD2	I/O	–	–	A21/NANDALE	O	–	–	–	–	A21,O, PD
P15	VDDNF	EBI	PD3	I/O	–	–	A22/NANDCLE	O	–	–	–	–	A22,O, PD
R17	VDDNF	EBI	PD4	I/O	–	–	NCS3	O	–	–	–	–	PIO, I, PU
M15	VDDNF	EBI	PD5	I/O	–	–	NWAIT	I	–	–	–	–	PIO, I, PU
N15	VDDNF	EBI	PD6	I/O	–	–	D16	O	–	–	–	–	PIO, I, PU
V13	VDDNF	EBI	PD7	I/O	–	–	D17	O	–	–	–	–	PIO, I, PU
L14	VDDNF	EBI	PD8	I/O	–	–	D18	O	–	–	–	–	PIO, I, PU
W18	VDDNF	EBI	PD9	I/O	–	–	D19	O	–	–	–	–	PIO, I, PU
V18	VDDNF	EBI	PD10	I/O	–	–	D20	O	–	–	–	–	PIO, I, PU
W19	VDDNF	EBI	PD11	I/O	–	–	D21	O	–	–	–	–	PIO, I, PU
V19	VDDNF	EBI	PD12	I/O	–	–	D22	O	–	–	–	–	PIO, I, PU
N18	VDDNF	EBI	PD13	I/O	–	–	D23	O	–	–	–	–	PIO, I, PU
L15	VDDNF	EBI	PD14	I/O	–	–	D24	O	–	–	–	–	PIO, I, PU
N17	VDDNF	EBI	PD15	I/O	–	–	D25	O	A20	O	–	–	A20, O, PD
M18	VDDNF	EBI	PD16	I/O	–	–	D26	O	A23	O	–	–	A23, O, PD
M17	VDDNF	EBI	PD17	I/O	–	–	D27	O	A24	O	–	–	A24, O, PD
P17	VDDNF	EBI	PD18	I/O	–	–	D28	O	A25	O	–	–	A25, O, PD
L18	VDDNF	EBI	PD19	I/O	–	–	D29	O	NCS2	O	–	–	PIO, I, PU
K15	VDDNF	EBI	PD20	I/O	–	–	D30	O	NCS4	O	–	–	PIO, I, PU
L17	VDDNF	EBI	PD21	I/O	–	–	D31	O	NCS5	O	–	–	PIO, I, PU
E8 E9 E13 F7 F8 F9 G14	VDDIOM	POWER	VDDIOM	I	–	–	–	–	–	–	–	–	I
M14 P13 U10 V9 V10 V11	VDDNF	POWER	VDDNF	I	–	–	–	–	–	–	–	–	I

© 2017 Microchip Technology Inc.

Table 3-2: BGA247 Pin Description (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
H6 H7 J6 J7 J8 F10 F11 F12 F13 F14 F15 F16	GNDIOM	GND	GNDIOM	I	–	–	–	–	–	–	–	–	I
N11 M12 M13	VDDIOP0	POWER	VDDIOP0	I	–	–	–	–	–	–	–	–	I
M9 M10 M11	VDDIOP1	POWER	VDDIOP1	I	–	–	–	–	–	–	–	–	I
L10 L11 L12 L13 V14	GNDIOP	GND	GNDIOP	I	–	–	–	–	–	–	–	–	I
B6	VDDBU	POWER	VDDBU	I	–	–	–	–	–	–	–	–	I
B7	GNDBU	GND	GNDBU	I	–	–	–	–	–	–	–	–	I
F6	VDDANA	POWER	VDDANA	I	–	–	–	–	–	–	–	–	I
C3	GNDANA	GND	GNDANA	I	–	–	–	–	–	–	–	–	I
V17	VDDPLL	POWER	VDDPLL	I	–	–	–	–	–	–	–	–	I
U16	GNDPLL	GND	GNDPLL	I	–	–	–	–	–	–	–	–	I
P14	VDDFUSE	POWER	VDDFUSE	I	–	–	–	–	–	–	–	–	I
N14	GNDFUSE	GND	GNDFUSE	I	–	–	–	–	–	–	–	–	I
R12	VDDOSC	POWER	VDDOSC	I	–	–	–	–	–	–	–	–	I
U13	VDDUSB	POWER	VDDUSB	I	–	–	–	–	–	–	–	–	I
U17	GNDUSB	GND	GNDUSB	I	–	–	–	–	–	–	–	–	I
J12 J13 J14 K10 K11 K12 K13 K14 U15	VDDCORE	POWER	VDDCORE	I	–	–	–	–	–	–	–	–	I
H9 J9 J10 J11 K7 K8 K9 L8 L9	GNDCORE	GND	GNDCORE	I	–	–	–	–	–	–	–	–	I
A19	VDDIOM	EBI	D0	I/O	–	–	–	–	–	–	–	–	O, PD
E15	VDDIOM	EBI	D1	I/O	–	–	–	–	–	–	–	–	O, PD

Table 3-2: BGA247 Pin Description (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
C18	VDDIOM	EBI	D2	I/O	–	–	–	–	–	–	–	–	O, PD
D15	VDDIOM	EBI	D3	I/O	–	–	–	–	–	–	–	–	O, PD
B17	VDDIOM	EBI	D4	I/O	–	–	–	–	–	–	–	–	O, PD
E14	VDDIOM	EBI	D5	I/O	–	–	–	–	–	–	–	–	O, PD
C16	VDDIOM	EBI	D6	I/O	–	–	–	–	–	–	–	–	O, PD
A18	VDDIOM	EBI	D7	I/O	–	–	–	–	–	–	–	–	O, PD
B15	VDDIOM	EBI	D8	I/O	–	–	–	–	–	–	–	–	O, PD
G12	VDDIOM	EBI	D9	I/O	–	–	–	–	–	–	–	–	O, PD
C14	VDDIOM	EBI	D10	I/O	–	–	–	–	–	–	–	–	O, PD
D13	VDDIOM	EBI	D11	I/O	–	–	–	–	–	–	–	–	O, PD
A16	VDDIOM	EBI	D12	I/O	–	–	–	–	–	–	–	–	O, PD
A14	VDDIOM	EBI	D13	I/O	–	–	–	–	–	–	–	–	O, PD
B13	VDDIOM	EBI	D14	I/O	–	–	–	–	–	–	–	–	O, PD
H13	VDDIOM	EBI	D15	I/O	–	–	–	–	–	–	–	–	O, PD
J15	VDDIOM	EBI_O	A0	O	NBS0	O	–	–	–	–	–	–	O
K18	VDDIOM	EBI_O	A1	O	NBS2/ DQM2/ NWR2	O	–	–	–	–	–	–	O
K17	VDDIOM	EBI_O	A2	O	–	–	–	–	–	–	–	–	O
H15	VDDIOM	EBI_O	A3	O	–	–	–	–	–	–	–	–	O
J18	VDDIOM	EBI_O	A4	O	–	–	–	–	–	–	–	–	O
J17	VDDIOM	EBI_O	A5	O	–	–	–	–	–	–	–	–	O
G17	VDDIOM	EBI_O	A6	O	–	–	–	–	–	–	–	–	O
H17	VDDIOM	EBI_O	A7	O	–	–	–	–	–	–	–	–	O
H18	VDDIOM	EBI_O	A8	O	–	–	–	–	–	–	–	–	O
H14	VDDIOM	EBI_O	A9	O	–	–	–	–	–	–	–	–	O
G18	VDDIOM	EBI_O	A10	O	–	–	–	–	–	–	–	–	O
F18	VDDIOM	EBI_O	A11	O	–	–	–	–	–	–	–	–	O
F17	VDDIOM	EBI_O	A12	O	–	–	–	–	–	–	–	–	O
E19	VDDIOM	EBI_O	A13	O	–	–	–	–	–	–	–	–	O
D19	VDDIOM	EBI_O	A14	O	–	–	–	–	–	–	–	–	O
E18	VDDIOM	EBI_O	A15	O	–	–	–	–	–	–	–	–	O
G15	VDDIOM	EBI_O	A16	O	BA0	O	–	–	–	–	–	–	O
E16	VDDIOM	EBI_O	A17	O	BA1	O	–	–	–	–	–	–	O
B19	VDDIOM	EBI_O	A18	O	BA2	O	–	–	–	–	–	–	O
D17	VDDIOM	EBI_O	A19	O	–	–	–	–	–	–	–	–	O
B9	VDDIOM	EBI_O	NCS0	O	–	–	–	–	–	–	–	–	O, PU
B8	VDDIOM	EBI_O	NCS1	O	SDCS	O	–	–	–	–	–	–	O, PU
E10	VDDIOM	EBI_O	NRD	O	–	–	–	–	–	–	–	–	O, PU
G10	VDDIOM	EBI_O	NWR0	O	NWRE	O	–	–	–	–	–	–	O, PU
C10	VDDIOM	EBI_O	NWR1	O	NBS1	O	–	–	–	–	–	–	O, PU

© 2017 Microchip Technology Inc.

Table 3-2: BGA247 Pin Description (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
G9	VDDIOM	EBI_O	NWR3	O	NBS3/ DQM3	O	–	–	–	–	–	–	O, PU
B10	VDDIOM	EBI_CLK	SDCK	O	–	–	–	–	–	–	–	–	O
B11	VDDIOM	EBI_CLK	#SDCK	O	–	–	–	–	–	–	–	–	O
C12	VDDIOM	EBI_O	SDCKE	O	–	–	–	–	–	–	–	–	O, PU
G11	VDDIOM	EBI_O	RAS	O	–	–	–	–	–	–	–	–	O, PU
E12	VDDIOM	EBI_O	CAS	O	–	–	–	–	–	–	–	–	O, PU
H12	VDDIOM	EBI_O	SDWE	O	–	–	–	–	–	–	–	–	O, PU
H10	VDDIOM	EBI_O	SDA10	O	–	–	–	–	–	–	–	–	O, PU
A12	VDDIOM	EBI_O	DQM0	O	–	–	–	–	–	–	–	–	O, PU
C11	VDDIOM	EBI_O	DQM1	O	–	–	–	–	–	–	–	–	O, PU
H11	VDDIOM	EBI	DQS0	I/O	–	–	–	–	–	–	–	–	I, PD
E11	VDDIOM	EBI	DQS1	I/O	–	–	–	–	–	–	–	–	I, PD
B3	VDDANA	POWER	ADVREF	I	–	–	–	–	–	–	–	–	I
T18	VDDUSB	USBFS	HDP	I/O	–	–	–	–	–	–	–	–	O, PD
U18	VDDUSB	USBFS	HDM	I/O	–	–	–	–	–	–	–	–	O, PD
P18	VDDUSB	USBFS	DDP	I/O	–	–	–	–	–	–	–	–	O, PD
R18	VDDUSB	USBFS	DDM	I/O	–	–	–	–	–	–	–	–	O, PD
C6	VDDBU	SYSC	WKUP	I	–	–	–	–	–	–	–	–	I, ST
G8	VDDBU	SYSC	SHDN	O	–	–	–	–	–	–	–	–	O, PU
U14	VDDIOP0	RSTJTAG	BMS	I	–	–	–	–	–	–	–	–	I, PU, ST
C4	VDDBU	SYSC	JTAGSEL	I	–	–	–	–	–	–	–	–	I, PD
C5	VDDBU	SYSC	TST	I	–	–	–	–	–	–	–	–	I, PD, ST
V8	VDDIOP0	RSTJTAG	TCK	I	–	–	–	–	–	–	–	–	I, ST
U8	VDDIOP0	RSTJTAG	TDI	I	–	–	–	–	–	–	–	–	I, ST
P12	VDDIOP0	RSTJTAG	TDO	O	–	–	–	–	–	–	–	–	O
R11	VDDIOP0	RSTJTAG	TMS	I	–	–	–	–	–	–	–	–	I, ST
V12	VDDIOP0	RSTJTAG	RTCK	O	–	–	–	–	–	–	–	–	O
U11	VDDIOP0	RSTJTAG	NRST	I/O	–	–	–	–	–	–	–	–	I, PU, ST
U9	VDDIOP0	RSTJTAG	NTRST	I	–	–	–	–	–	–	–	–	I, PU, ST
B4	VDDBU	CLOCK	XIN32	I	–	–	–	–	–	–	–	–	I
B5	VDDBU	CLOCK	XOUT32	O	–	–	–	–	–	–	–	–	O
V16	VDDIOP0	CLOCK	XIN	I	–	–	–	–	–	–	–	–	I
V15	VDDIOP0	CLOCK	XOUT	O	–	–	–	–	–	–	–	–	O
H8	–	–	NC	–	–	–	–	–	–	–	–	–	–
U12	–	–	NC	–	–	–	–	–	–	–	–	–	–
R13	–	–	NC	–	–	–	–	–	–	–	–	–	–

4. Power Considerations

4.1 Power Supplies

The SAM9N12/CN11/CN12 has several types of power supply pins.

Table 4-1 defines the different power supplies rails and the estimated power consumption at typical voltage. For details about power-up and power-down sequences, refer to Section 4.2 ”Power Sequence Requirements”.

Table 4-1: SAM9N12/CN11/CN12 Power Supplies

Name	Voltage Range, Nominal	Associated Ground	Powers
VDDCORE	0.9–1.1V, 1.0V	GNDCORE	Core, including the processor, the embedded memories and the peripherals, the internal 12 MHz RC
VDDIOM	1.65–1.95V, 1.8V 3.0–3.6V, 3.3V	GNDIOM	External Memory Interface I/O lines
VDDNF	1.65–1.95V, 1.8V 3.0–3.6V, 3.3V	GNDIOM	NAND Flash I/O and control, D16–D32 and multiplexed SMC lines
VDDIOP0	1.65–3.6V	GNDIOP	Part of Peripherals I/O lines
VDDIOP1	1.65–3.6V	GNDIOP	Part of Peripherals I/O lines
VDDBU	1.65–3.6V	GNDBU	Slow Clock oscillator, the internal 32 Kbyte RC and a part of the System Controller
VDDUSB	3.0–3.6V, 3.3V	GNDUSB	USB interface
VDDPLL	0.9–1.1V, 1.0V	GNDPLL	PLL cells
VDDOSC	1.65–3.6V	GNDPLL	Main Oscillator cells
VDDANA	3.0–3.6V, 3.3V	GNDANA	Analog to Digital Converter
VDDFUSE	3.0–3.6V, 3.3V	GNDFUSE	Fuse box for programming

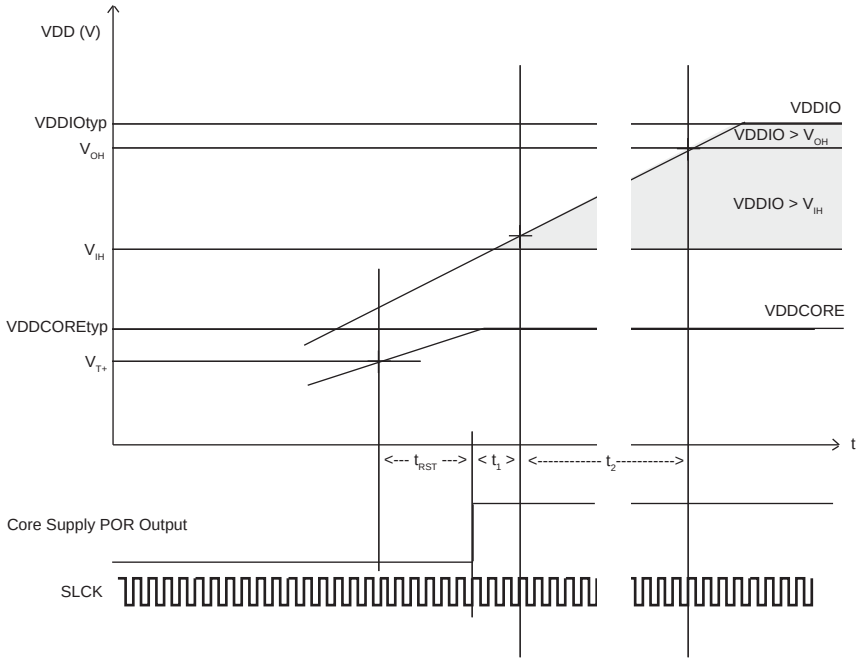
4.2 Power Sequence Requirements

The AT91 board design must comply with the power-up guidelines below to guarantee reliable operation of the device. Any deviation from these sequences may prevent the device from booting.

© 2017 Microchip Technology Inc.

4.2.1 Power-Up Sequence

Figure 4-1: VDDCORE and VDDIO Constraints at Startup



VDDCORE and VDDBU are controlled by internal POR (Power-On-Reset) to guarantee that these power sources reach their target values prior to the release of POR.

- VDDIOP must be ≥ V_{IH} (refer to Table 47-2 “DC Characteristics”, for more details), (t_{RST} + t_1) at the latest, after VDDCORE has reached V_{T+}.
- VDDIOM must reach V_{OH} (refer to Table 47-2 “DC Characteristics”, for more details), (t_{RST} + t_1 + t_2) at the latest, after VDDCORE has reached V_{T+}
  - t_{RST} is a POR characteristic
  - $t_1 = 3 \times t_{SLCK}$
  - $t_2 = 16 \times t_{SLCK}$

The t_{SLCK} min (22 µs) is obtained for the maximum frequency of the internal RC oscillator (44 kHz).

  - t_{RST} = 30 µs
  - t_1 = 66 µs
  - t_2 = 352 µs
- VDDPLL is to be established prior to VDDCORE to ensure the PLL is powered once enabled into the ROM code.

As a conclusion, establish VDDIOP and VDDIOM first, then VDDPLL, and VDDCORE at last, to ensure a reliable operation of the device.

4.2.2 Power-Down Sequence

To ensure that the device does not operate outside the operating conditions defined in Table 4-1 “SAM9N12/CN11/CN12 Power Supplies”, it is good practice to first place the device in reset state before removing its power supplies. No specific sequencing is required with respect to its supply channels as long as the NRST line is held active during the the power-down phase.

Figure 4-2: Recommended Power-Down Sequence

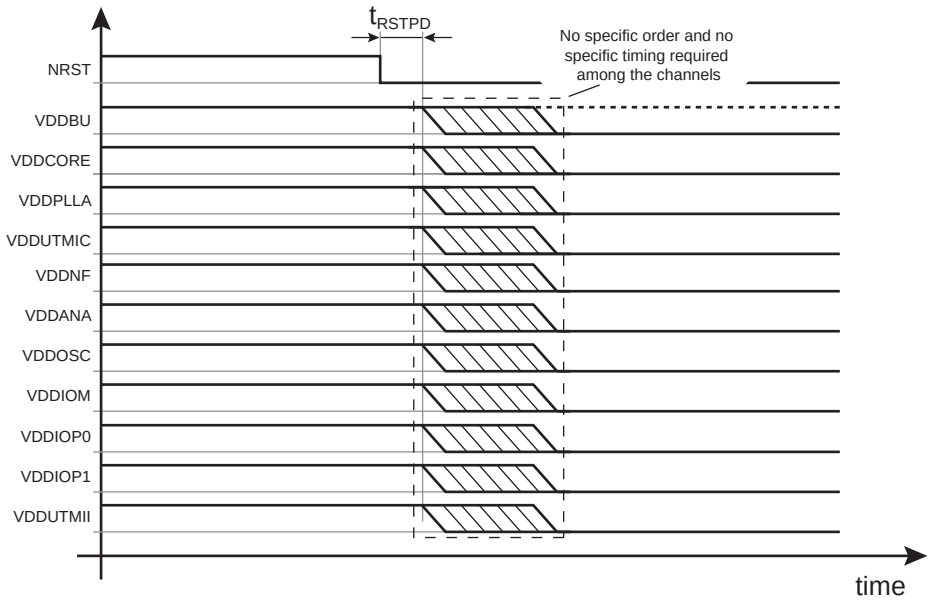


Table 4-2: Power-down Timing Specification

Symbol	Parameter	Conditions	Min	Max	Unit
t_{RSTPD}	Reset Delay at Power-Down	From NRST low to the first supply turn-off	0	–	ms

© 2017 Microchip Technology Inc.

5. Memories

5.1 Memory Mapping

Figure 5-1 provides the device memory map.

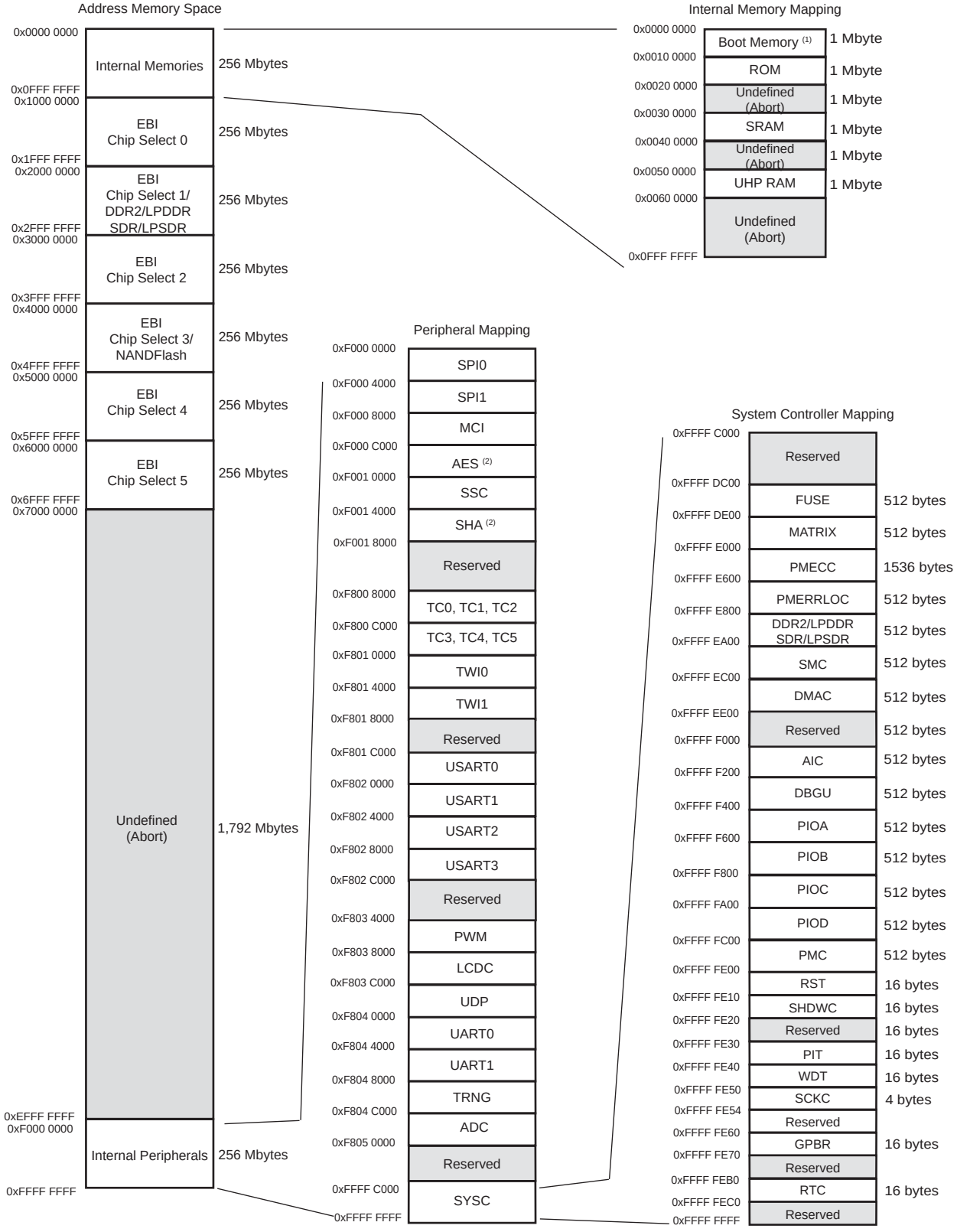
A first level of address decoding is performed by the AHB Bus Matrix, i.e., the implementation of the Advanced High performance Bus (AHB) for its Master and Slave interfaces with additional features.

Decoding breaks up the 4 Gbytes of address space into 16 banks of 256 Mbytes. The banks 1 to 6 are directed to the EBI that associates these banks to the external chip selects EBI_NCS0 to EBI_NCS5. The bank 0 is reserved for the addressing of the internal memories, and a second level of decoding provides 1 Mbyte of internal memory area. Bank 15 is reserved for the peripherals and provides access to the Advanced Peripheral Bus (APB).

Other areas are unused and performing an access within them provides an abort to the master requesting such an access.

© 2017 Microchip Technology Inc.

Figure 5-1: SAM9N12/CN11/CN12 Memory Map



Notes:
1. Can be ROM, EBI1_NCS0 or SRAM depending on BMS and REMAP
2. Reserved for SAM9N12

© 2017 Microchip Technology Inc.

5.2 Embedded Memories

5.2.1 Internal SRAM

The SAM9N12/CN11/CN12 embeds a total of 32 Kbytes high-speed SRAM.

After reset and until the Remap Command is performed, the SRAM is only accessible at address 0x0030 0000.

After Remap, the SRAM also becomes available at address 0x0.

5.2.2 Internal ROM

The SAM9CN12 contains the secure bootloader (standard bootloader for SAM9N12 and SAM9CN11) and specific tables used to compute SLC and MLC NAND Flash ECC.

The ROM is mapped at address 0x0010 0000. It is also accessible at address 0x0 (BMS = 1) after the reset and before the Remap Command.

5.3 External Memories Overview

The SAM9N12/CN11/CN12 features an External Bus Interface to provide interface to a wide range of external memories and to any parallel peripheral.

5.3.1 External Bus Interface

- Integrates three External Memory Controllers:
  - Static Memory Controller
  - DDR2/SDRAM Controller
  - MLC NAND Flash ECC Controller
- Up to 26-bit Address Bus (up to 64 Mbytes linear per chip select)
- Up to 6 chips selects, Configurable Assignment:
  - Static Memory Controller on NCS0, NCS1, NCS2, NCS3, NCS4, NCS5
  - DDR2/SDRAM Controller (SDCS) or Static Memory Controller on NCS1
  - NAND Flash support on NCS3

5.3.2 Static Memory Controller

- 8- or 16-bit Data Bus
- Multiple Access Modes supported
  - Byte Write or Byte Select Lines
  - Asynchronous read in Page Mode supported (4- up to 16-byte page size)
- Multiple device adaptability
  - Control signals programmable setup, pulse and hold time for each Memory Bank
- Multiple Wait State Management
  - Programmable Wait State Generation
  - External Wait Request
  - Programmable Data Float Time
- Slow Clock mode supported

5.3.3 DDR-SDRAM Controller

- Supports DDR2-SDRAM, Low-power DDR1-SDRAM, SDR-SDRAM and Low-power SDR-SDRAM
- Numerous Configurations Supported
  - 2K, 4K, 8K, 16K Row Address Memory Parts
  - SDRAM with 4 Internal Banks
  - SDR-SDRAM with 16-bit or 32-bit Data Path
  - DDR-SDRAM with 16-bit Data Path
  - One Chip Select for SDRAM Device (256 Mbytes Address Space)
- Programming Facilities
  - Multibank Ping-pong Access (Up to 4 Banks or 8 Banks Opened at Same Time = Reduced Average Latency of Transactions)
  - Timing Parameters Specified by Software
  - Automatic Refresh Operation, Refresh Rate is Programmable
  - Automatic Update of DS, TCR and PASR Parameters (Low-power SDRAM Devices)

- Energy-saving Capabilities
  - Self-refresh, Power-down, Active Power-down and Deep Power-down Modes Supported
- SDRAM Power-up Initialization by Software
- CAS Latency of 2, 3 Supported
- Reset Function Supported (DDR2-SDRAM)
- ODT (On-die Termination) Not Supported
- **Auto Precharge Command Not Used**
- SDR-SDRAM with 16-bit Datapath and Eight Columns Not Supported
- DDR2-SDRAM with Eight **Internal Banks Supported**
- Linear and interleaved decoding supported
- Clock Frequency Change in Precharge Power-down Mode Not Supported
- OCD (Off-chip Driver) Mode Not Supported

5.3.4 Programmable Multibit Error Correction Code (PMECC)

- Multibit Error Correcting Code
- Algorithm based on binary shortened Bose, Chaudhuri and Hocquenghem (BCH) codes
- Programmable Error Correcting Capability: 2, 4, 8, 16 and 24 bit of errors per block
- Programmable block size: 512 bytes or 1024 bytes
- Programmable number of block per page: 1, 2, 4 or 8 blocks of data per page
- Programmable spare area size
- Supports spare area ECC protection
- Supports 8 Kbytes page size using 1024 bytes/block and 4 Kbytes page size using 512 bytes/block
- Multibit Error detection is interrupt driven

5.3.5 Programmable Multi-bit ECC Error Location (PMERRLOC)

- Provides hardware acceleration for determining roots of polynomials defined over a finite field
- Programmable finite Field GF(2^13) or GF(2^14)
- Finds roots of error-locator polynomial.
- Programmable number of roots.

© 2017 Microchip Technology Inc.

6. System Controller

The System Controller is a set of peripherals that allows handling of key elements of the system, such as power, resets, clocks, time, interrupts, watchdog, etc.

The System Controller User Interface also embeds the registers that configure the Matrix and a set of registers for the chip configuration. The chip configuration registers configure the EBI chip select assignment and voltage range for external memories.

6.1 System Controller Mapping

The System Controller's peripherals are all mapped within the highest 16 Kbytes of address space, between addresses 0xFFFF E400 and 0xFFFF FFFF.

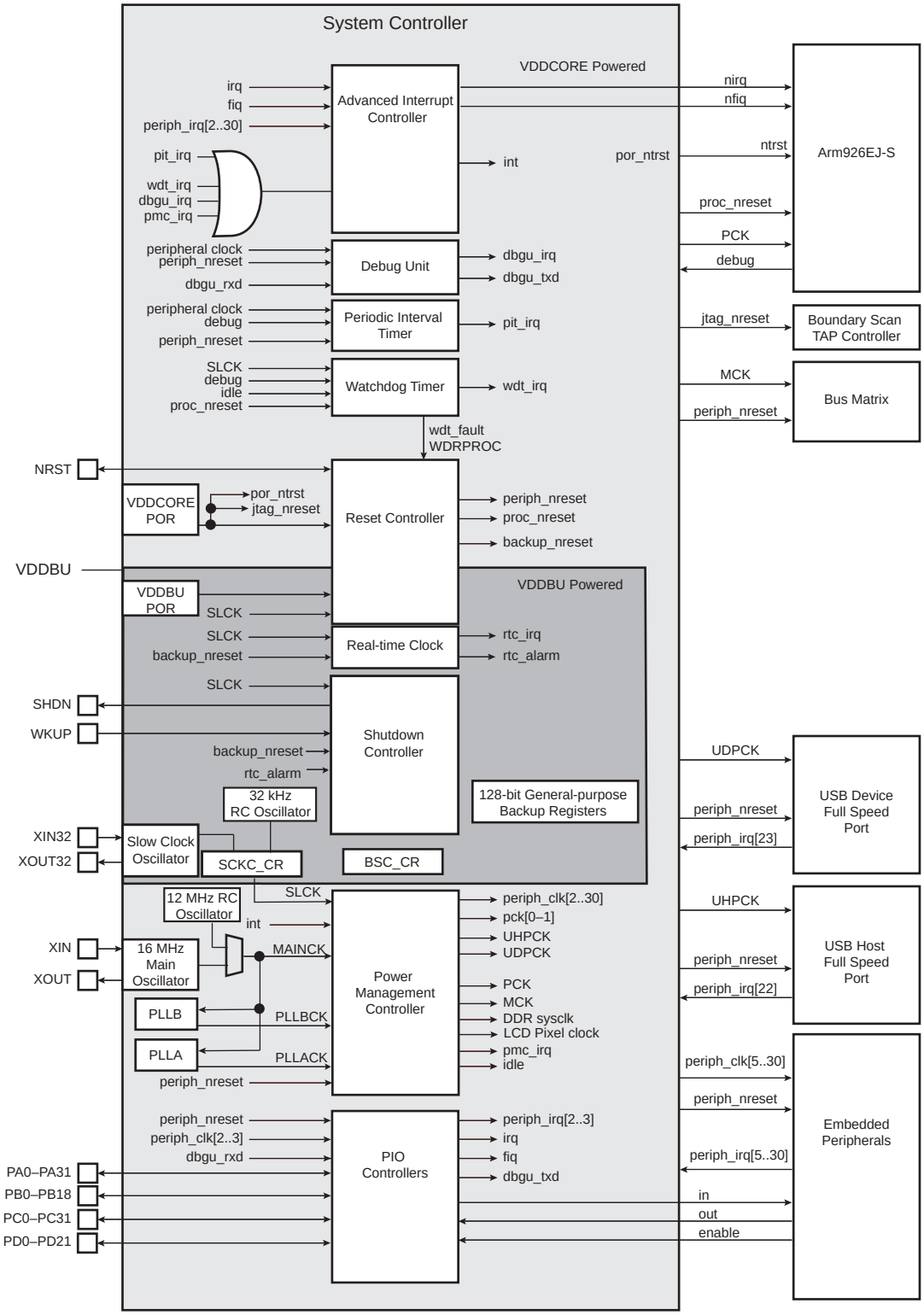
However, all the registers of the System Controller are mapped on the top of the address space. All the registers of the System Controller can be addressed from a single pointer by using the standard Arm instruction set, as the Load/Store instruction have an indexing mode of ±4 Kbytes.

Figure 6-1 shows the System Controller block diagram.

Figure 5-1 shows the mapping of the User Interfaces of the System Controller peripherals.

6.2 System Controller Block Diagram

Figure 6-1: SAM9N12/CN11/CN12 System Controller Block Diagram



© 2017 Microchip Technology Inc.

6.3 Chip Identification

- Chip ID: 0x819A_07A1
- SAM9CN12 Chip ID Extension: 5
- SAM9CN11 Chip ID Extension: 9
- SAM9N12 Chip ID Extension: 6
- JTAG ID: 0x05B3_003F
- Arm926 TAP ID: 0x0792_603F

6.4 Backup Section

The SAM9N12/CN11/CN12 features a backup section that embeds:

- RC Oscillator
- Slow Clock Oscillator
- Real Time Clock (RTC)
- Shutdown Controller (SHDWC)
- 128-bit General-purpose Backup Registers (GPBR)
- Slow Clock Controller Configuration Register (SCKC_CR)
- A part of the Reset Controller (RSTC)

This backup section is powered by the VDDBU rail.

7. Peripherals

7.1 Peripheral Mapping

As shown in Figure 5-1, the Peripherals are mapped in the upper 256 Mbytes of the address space between the addresses 0xFFF7 8000 and 0xFFFC FFFF.

Each User Peripheral is allocated 16 Kbytes of address space.

7.2 Peripheral Identifiers

Table 7-1 defines the Peripheral Identifiers of the SAM9N12/CN11/CN12. A peripheral identifier is required for the control of the peripheral interrupt with the Advanced Interrupt Controller and for the control of the peripheral clock with the Power Management Controller.

Table 7-1: SAM9N12/CN11/CN12 Peripheral Identifiers

Instance ID	Instance Name	Instance Description	External Interrupt	Wired-or Interrupt
0	AIC	Advanced Interrupt Controller	FIQ	–
1	SYS	System Controller Interrupt	–	DBGU, PMC, SYSC, PMECC, PMERRLOC
2	PIOA, PIOB	Parallel I/O Controller A and B	–	–
3	PIOC, PIOD	Parallel I/O Controller C and D	–	–
4	FUSE	FUSE Controller	–	–
5	USART0	Universal Synchronous Asynchronous Receiver Transceiver 0	–	–
6	USART1	Universal Synchronous Asynchronous Receiver Transceiver 1	–	–
7	USART2	Universal Synchronous Asynchronous Receiver Transceiver 2	–	–
8	USART3	Universal Synchronous Asynchronous Receiver Transceiver 3	–	–
9	TWI0	Two-wire Interface 0	–	–
10	TWI1	Two-wire Interface 1	–	–
11	Reserved	–	–	–
12	HSMCI	High Speed Multimedia Card Interface	–	–
13	SPI0	Serial Peripheral Interface 0	–	–
14	SPI1	Serial Peripheral Interface 1	–	–
15	UART0	Universal Asynchronous Receiver Transmitter 0	–	–
16	UART1	Universal Asynchronous Receiver Transmitter 1	–	–
17	TC0 TC1	Timer Counter 0 (channels 0, 1, 2) Timer Counter 1 (channels 3, 4, 5)	–	–
18	PWM	Pulse Width Modulation Controller	–	–
19	ADC	ADC Controller	–	–
20	DMAC	DMA Controller	–	–
21	Reserved	–	–	–
22	UHP	USB Host Port	–	–
23	UDP	USB Device Port	–	–

© 2017 Microchip Technology Inc.

Table 7-1: SAM9N12/CN11/CN12 Peripheral Identifiers (Continued)

Instance ID	Instance Name	Instance Description	External Interrupt	Wired-or Interrupt
24	Reserved	–	–	–
25	LCDC	LCD Controller	–	–
26	Reserved	–	–	–
27	SHA	Secure Hash Algorithm	–	–
28	SSC	Synchronous Serial Controller	–	–
29	AES	Advanced Encryption Standard	–	–
30	TRNG	True Random Number Generator	–	–
31	AIC	Advanced Interrupt Controller	IRQ	–

7.3 Peripheral Interrupts and Clock Control

7.3.1 System Interrupt

The System Interrupt in Source 1 is the wired-OR of the interrupt signals coming from:

- the DDR2/LPDDR Controller
- the Debug Unit
- the Periodic Interval Timer
- the Real-Time Clock
- the Watchdog Timer
- the Reset Controller
- the Power Management Controller

The clock of these peripherals cannot be deactivated and Peripheral ID 1 can only be used within the Advanced Interrupt Controller.

7.3.2 External Interrupts

All external interrupt signals, i.e., the Fast Interrupt signal FIQ or the Interrupt signal IRQ, use a dedicated Peripheral ID. However, there is no clock control associated with these peripheral IDs.

7.4 Peripheral Signal Multiplexing on I/O Lines

The SAM9N12/CN11/CN12 features four PIO controllers, PIOA, PIOB, PIOC and PIOD, which multiplex the I/O lines of the peripheral set.

Each PIO controller controls 32 lines for PIOA, 19 lines for PIOB, 32 lines for PIOC, and 22 lines for PIOD. Each line can be assigned to one of three peripheral functions: A, B or C.

Refer to Section 3. “Package and Pinout” and the package pinouts in Table 3-1 “BGA217 Pin Description” and Table 3-2 “BGA247 Pin Description”.

7.4.1 Reset State

The column “Reset State” in Table 3-1 and Table 3-2 indicates the reset state of the line with mnemonics.

- “PIO”/”signal”

  Indicates whether the PIO Line resets in I/O mode or in peripheral mode. If “PIO” is mentioned, the PIO Line is maintained in a static state as soon as the reset is released. As a result, the bit corresponding to the PIO Line in the register PIO_PSR (Peripheral Status Register) resets low.

  If a signal name is mentioned in the “Reset State” column, the PIO Line is assigned to this function and the corresponding bit in PIO_PSR resets high. This is the case on pins controlling memories, in particular the address lines, which require the pin to be driven as soon as the reset is released.

- ‘I’/’O’

  Indicates whether the signal is input or output state.

- “PU”/”PD”

  Indicates whether Pull-up or Pull-down, or nothing is enabled.

- “ST”

  Indicates if Schmitt Trigger is enabled.

Note: Example: The PB18 “Reset State” column shows “PIO, I, PU, ST”. That means the line PIO18 is configured as an Input with Pull-Up and Schmitt Trigger enabled. PD14 reset state is “PIO, I, PU”. That means PIO Input with Pull-Up. PD15 reset state is “A20, O, PD” which means output address line 20 with Pull-Down.

7.4.2 PIO Line Selection

Peripheral A, B or C is selected via the PIO_ABCDSR1 and PIO_ABCDSR2 registers in the PIO Controller Interface.

Table 7-2: PIO Line Selection

Px value in PIO_ABCDSR2	Px value in PIO_ABCDSR1	A, B or C
0	0	A
0	1	B
1	0	C

7.5 Fuse Box Features

SAM9CN12 embeds 320 One Time Programming (OTP) bits. When the OTP bit is set, it is seen as ‘1’. The user interface allows the user to perform the following operations:

7.5.1 Read

- 10 registers SR0–SR9 that reflect OTP bit state
- MSK field (write-once) allow user to mask registers SR1 to SR9
- All OTP bits are read as ‘1’ when VDDFUSE is floating, all security features are set.

7.5.2 Write

- Done in one 32-bit DATA register
- SEL field to select the 32-bit word 0 to 9

7.5.3 Fuse Mapping

Table 7-3: OTP Mapping

<table>
<tr><th>Bits</th><th colspan="3">Functions</th></tr>
<tr><td>[319:288]</td><td>config[319:312]</td><td>RC_trim[311:305]</td><td>RFU[304:288]</td></tr>
<tr><td>[287:256]</td><td colspan="3" rowspan="10">RFU[287:0](1)</td></tr>
<tr><td>[255:224]</td></tr>
<tr><td>[223:192]</td></tr>
<tr><td>[191:160]</td></tr>
<tr><td>[159:128]</td></tr>
<tr><td>[127:96]</td></tr>
<tr><td>[95:64]</td></tr>
<tr><td>[63:32]</td></tr>
<tr><td>[31:0]</td></tr>
</table>

Note 1: For SAM9CN12 these bits are used for Keys and cannot be used by user for another purpose

© 2017 Microchip Technology Inc.

Table 7-4: Device Configuration OTP Bits Description

Bit Number	Acronym	Function
312	W	OTP bit writing is disabled if set OTP bits are not accessible in test mode
313	B	BMS sampling option is disabled if set, system boots only into ROM code
314	J	JTAG and tap controller are disabled if set
315	S	Secure boot Loader is disabled if set
316	K	OTP keys are locked
317–319	Reserved	Reserved

8. Arm926EJ-S Processor Overview

8.1 Description

The Arm926EJ-S processor is a member of the Arm9™ family of general-purpose microprocessors. The Arm926EJ-S implements Arm architecture version 5TEJ and is targeted at multi-tasking applications where full memory management, high performance, low die size and low power are all important features.

The Arm926EJ-S processor supports the 32-bit Arm and 16-bit Thumb instruction sets, enabling the user to trade off between high performance and high code density. It also supports 8-bit Java instruction set and includes features for efficient execution of Java bytecode, providing a Java performance similar to a JIT (Just-In-Time compilers), for the next generation of Java-powered wireless and embedded devices. It includes an enhanced multiplier design for improved DSP performance.

The Arm926EJ-S processor supports the Arm debug architecture and includes logic to assist in both hardware and software debug.

The Arm926EJ-S provides a complete high performance processor subsystem, including:

- Arm9EJ-S™ integer core
- Memory Management Unit (MMU)
- Separate instruction and data AMBA AHB bus interfaces
- Separate instruction and data TCM interfaces

8.2 Embedded Characteristics

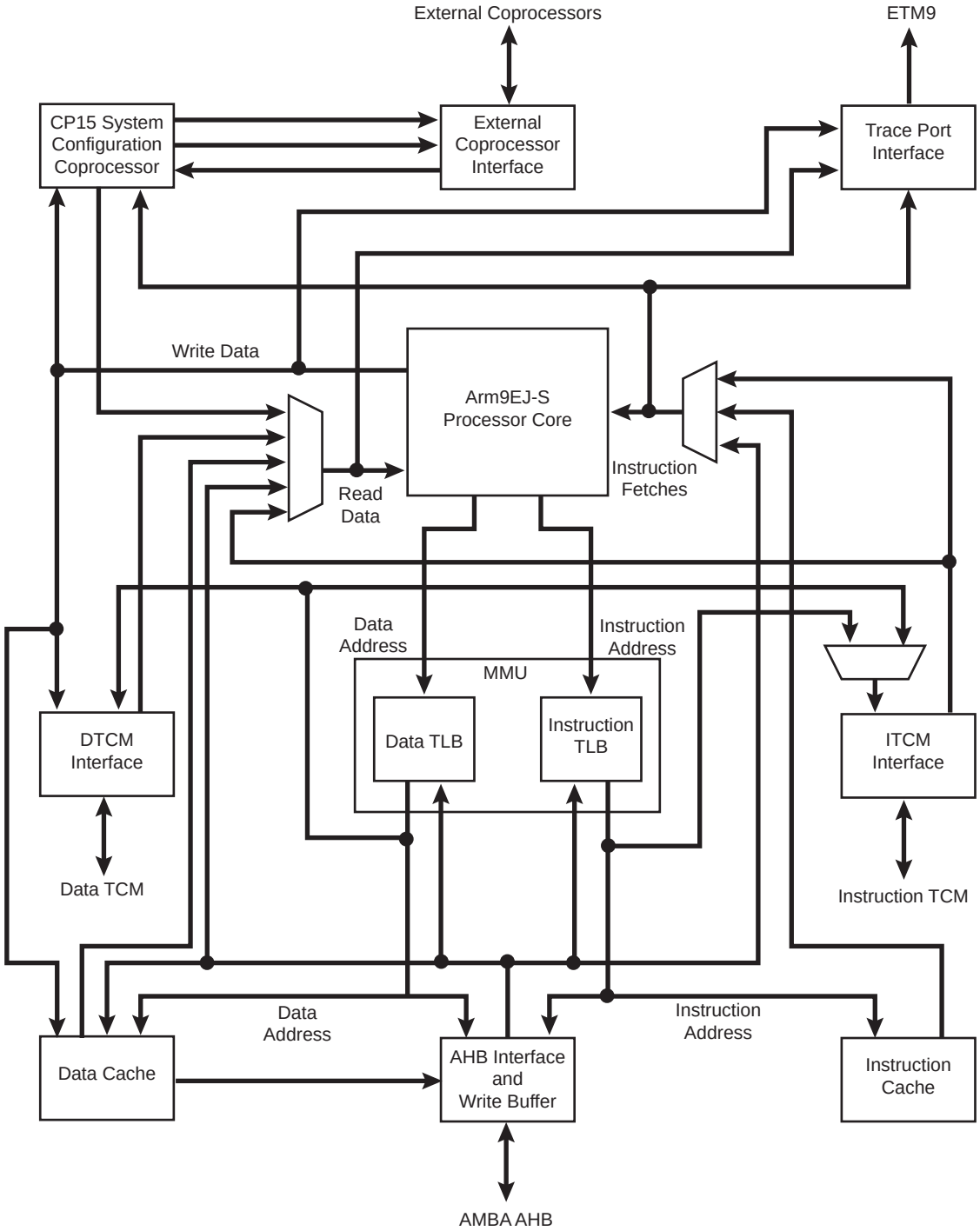
- Arm9EJ-S Based on Arm Architecture v5TEJ with Jazelle® Technology
  - Three Instruction Sets
  - Arm High-performance 32-bit Instruction Set
  - Thumb High Code Density 16-bit Instruction Set
  - Jazelle 8-bit Instruction Set
- 5-Stage Pipeline Architecture when Jazelle is not Used
  - Fetch (F)
  - Decode (D)
  - Execute (E)
  - Memory (M)
  - Writeback (W)
- 6-Stage Pipeline when Jazelle is Used
  - Fetch
  - Jazelle/Decode (Two Cycles)
  - Execute
  - Memory
  - Writeback
- ICache and DCache
  - Virtually-addressed 4-way Set Associative Caches
  - 8 Words per Line
  - Critical-word First Cache Refilling
  - Write-though and Write-back Operation for DCache Only
  - Pseudo-random or Round-robin Replacement
  - Cache Lockdown Registers
  - Cache Maintenance
- Write Buffer
  - 16-word Data Buffer
  - 4-address Address Buffer
  - Software Control Drain
- DCache Write-back Buffer
  - 8 Data Word Entries
  - One Address Entry
  - Software Control Drain
- Tightly-coupled Memory (TCM)
  - Separate Instruction and Data TCM Interfaces

© 2017 Microchip Technology Inc.

  - Provides a Mechanism for DMA Support
- Memory Management Unit (MMU)
  - Access Permission for Sections
  - Access Permission for Large Pages and Small Pages
  - 16 Embedded Domains
  - 64 Entry Instruction TLB and 64 Entry Data TLB
- Memory Access
  - 8-, 16-, and 32-bit Data Types
  - Separate AMBA AHB Buses for Both the 32-bit Data Interface and the 32-bit Instructions Interface
- Bus Interface Unit
  - Arbitrates and Schedules AHB Requests
  - Enables Multi-layer AHB to be Implemented
  - Increases Overall Bus Bandwidth
  - Makes System Architecture Mode Flexible

8.3 Block Diagram

Figure 8-1: Arm926EJ-S Internal Functional Block Diagram



© 2017 Microchip Technology Inc.

8.4 Arm9EJ-S Processor

8.4.1 Arm9EJ-S Operating States

The Arm9EJ-S processor can operate in three different states, each with a specific instruction set:

- Arm state: 32-bit, word-aligned Arm instructions.
- Thumb state: 16-bit, halfword-aligned Thumb instructions.
- Jazelle state: variable length, byte-aligned Jazelle instructions.

In Jazelle state, all instruction Fetches are in words.

8.4.2 Switching State

The operating state of the Arm9EJ-S core can be switched between:

- Arm state and Thumb state using the BX and BLX instructions, and loads to the PC
- Arm state and Jazelle state using the BXJ instruction

All exceptions are entered, handled and exited in Arm state. If an exception occurs in Thumb or Jazelle states, the processor reverts to Arm state. The transition back to Thumb or Jazelle states occurs automatically on return from the exception handler.

8.4.3 Instruction Pipelines

The Arm9EJ-S core uses two kinds of pipelines to increase the speed of the flow of instructions to the processor.

A five-stage (five clock cycles) pipeline is used for Arm and Thumb states. It consists of Fetch, Decode, Execute, Memory and Writeback stages.

A six-stage (six clock cycles) pipeline is used for Jazelle state It consists of Fetch, Jazelle/Decode (two clock cycles), Execute, Memory and Writeback stages.

8.4.4 Memory Access

The Arm9EJ-S core supports byte (8-bit), half-word (16-bit) and word (32-bit) access. Words must be aligned to four-byte boundaries, half-words must be aligned to two-byte boundaries and bytes can be placed on any byte boundary.

Because of the nature of the pipelines, it is possible for a value to be required for use before it has been placed in the register bank by the actions of an earlier instruction. The Arm9EJ-S control logic automatically detects these cases and stalls the core or forward data.

8.4.5 Jazelle Technology

The Jazelle technology enables direct and efficient execution of Java byte codes on Arm processors, providing high performance for the next generation of Java-powered wireless and embedded devices.

The new Java feature of Arm9EJ-S can be described as a hardware emulation of a JVM (Java Virtual Machine). Java mode will appear as another state: instead of executing Arm or Thumb instructions, it executes Java byte codes. The Java byte code decoder logic implemented in Arm9EJ-S decodes 95% of executed byte codes and turns them into Arm instructions without any overhead, while less frequently used byte codes are broken down into optimized sequences of Arm instructions. The hardware/software split is invisible to the programmer, invisible to the application and invisible to the operating system. All existing Arm registers are re-used in Jazelle state and all registers then have particular functions in this mode.

Minimum interrupt latency is maintained across both Arm state and Java state. Since byte codes execution can be restarted, an interrupt automatically triggers the core to switch from Java state to Arm state for the execution of the interrupt handler. This means that no special provision has to be made for handling interrupts while executing byte codes, whether in hardware or in software.

8.4.6 Arm9EJ-S Operating Modes

In all states, there are seven operation modes:

- User mode is the usual Arm program execution state. It is used for executing most application programs
- Fast Interrupt (FIQ) mode is used for handling fast interrupts. It is suitable for high-speed data transfer or channel process
- Interrupt (IRQ) mode is used for general-purpose interrupt handling
- Supervisor mode is a protected mode for the operating system
- Abort mode is entered after a data or instruction prefetch abort
- System mode is a privileged user mode for the operating system
- Undefined mode is entered when an undefined instruction exception occurs

Mode changes may be made under software control, or may be brought about by external interrupts or exception processing. Most application programs execute in User Mode. The non-user modes, known as privileged modes, are entered in order to service interrupts or exceptions or to access protected resources.

8.4.7 Arm9EJ-S Registers

The Arm9EJ-S core has a total of 37 registers.

- 31 general-purpose 32-bit registers
- Six 32-bit status registers

Table 8-1 shows all the registers in all modes.

Table 8-1: Arm9TDMI Modes and Registers Layout

User and System Mode	Supervisor Mode	Abort Mode	Undefined Mode	Interrupt Mode	Fast Interrupt Mode
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8_FIQ
R9	R9	R9	R9	R9	R9_FIQ
R10	R10	R10	R10	R10	R10_FIQ
R11	R11	R11	R11	R11	R11_FIQ
R12	R12	R12	R12	R12	R12_FIQ
R13	R13_SVC	R13_ABORT	R13_UNDEF	R13_IRQ	R13_FIQ
R14	R14_SVC	R14_ABORT	R14_UNDEF	R14_IRQ	R14_FIQ
PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_SVC	SPSR_ABORT	SPSR_UNDEF	SPSR_IRQ	SPSR_FIQ

Mode-specific banked registers

The Arm state register set contains 16 directly-accessible registers, r0 to r15, and an additional register, the Current Program Status Register (CPSR). Registers r0 to r13 are general-purpose registers used to hold either data or address values. Register r14 is used as a Link register that holds a value (return address) of r15 when BL or BLX is executed. Register r15 is used as a program counter (PC), whereas the Current Program Status Register (CPSR) contains condition code flags and the current mode bits.

In privileged modes (FIQ, Supervisor, Abort, IRQ, Undefined), mode-specific banked registers (r8 to r14 in FIQ mode or r13 to r14 in the other modes) become available. The corresponding banked registers r14_fiq, r14_svc, r14_abt, r14_irq, r14_und are similarly used to hold the values (return address for each mode) of r15 (PC) when interrupts and exceptions arise, or when BL or BLX instructions are executed within interrupt or exception routines. There is another register called Saved Program Status Register (SPSR) that becomes available in privileged modes instead of CPSR. This register contains condition code flags and the current mode bits saved as a result of the exception that caused entry to the current (privileged) mode.

In all modes and due to a software agreement, register r13 is used as stack pointer.

© 2017 Microchip Technology Inc.

The use and the function of all the registers described above should obey Arm Procedure Call Standard (APCS) which defines:

- constraints on the use of registers
- stack conventions
- argument passing and result return

For more details, refer to Arm Software Development Kit.

The Thumb state register set is a subset of the Arm state set. The programmer has direct access to:

- Eight general-purpose registers r0-r7
- Stack pointer, SP
- Link register, LR (Arm r14)
- PC
- CPSR

There are banked registers SPs, LRs and SPSRs for each privileged mode (for more details see the Arm9EJ-S Technical Reference Manual, revision r1p2 page 2-12).

8.4.7.1 Status Registers

The Arm9EJ-S core contains one CPSR, and five SPSRs for exception handlers to use. The program status registers:

- hold information about the most recently performed ALU operation
- control the enabling and disabling of interrupts
- set the processor operation mode

Figure 8-2: Status Register Format

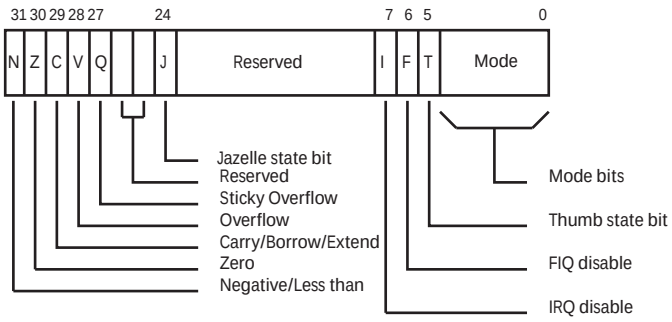


Figure 8-2 shows the status register format, where:

- N: Negative, Z: Zero, C: Carry, and V: Overflow are the four ALU flags
- The Sticky Overflow (Q) flag can be set by certain multiply and fractional arithmetic instructions like QADD, QDADD, QSUB, QDSUB, SMLAxy, and SMLAWy needed to achieve DSP operations.
  The Q flag is sticky in that, when set by an instruction, it remains set until explicitly cleared by an MSR instruction writing to the CPSR. Instructions cannot execute conditionally on the status of the Q flag.
- The J bit in the CPSR indicates when the Arm9EJ-S core is in Jazelle state, where:
  - J = 0: The processor is in Arm or Thumb state, depending on the T bit
  - J = 1: The processor is in Jazelle state.
- Mode: five bits to encode the current processor mode

8.4.8 Exceptions

8.4.8.1 Exception Types and Priorities

The Arm9EJ-S supports five types of exceptions. Each type drives the Arm9EJ-S in a privileged mode. The types of exceptions are:

- Fast interrupt (FIQ)
- Normal interrupt (IRQ)
- Data and Prefetched aborts (Abort)
- Undefined instruction (Undefined)
- Software interrupt and Reset (Supervisor)

When an exception occurs, the banked version of R14 and the SPSR for the exception mode are used to save the state.

More than one exception can happen at a time, therefore the Arm9EJ-S takes the arisen exceptions according to the following priority order:

- Reset (highest priority)
- Data Abort
- FIQ
- IRQ
- Prefetch Abort
- BKPT, Undefined instruction, and Software Interrupt (SWI) (Lowest priority)

The BKPT, or Undefined instruction, and SWI exceptions are mutually exclusive.

Note that there is one exception in the priority scheme: when FIQs are enabled and a Data Abort occurs at the same time as an FIQ, the Arm9EJ-S core enters the Data Abort handler, and proceeds immediately to FIQ vector. A normal return from the FIQ causes the Data Abort handler to resume execution. Data Aborts must have higher priority than FIQs to ensure that the transfer error does not escape detection.

8.4.8.2 Exception Modes and Handling

Exceptions arise whenever the normal flow of a program must be halted temporarily, for example, to service an interrupt from a peripheral.

When handling an Arm exception, the Arm9EJ-S core performs the following operations:

1. Preserves the address of the next instruction in the appropriate Link Register that corresponds to the new mode that has been entered. When the exception entry is from:
   - Arm and Jazelle states, the Arm9EJ-S copies the address of the next instruction into LR (current PC(r15) + 4 or PC + 8 depending on the exception).
   - THUMB state, the Arm9EJ-S writes the value of the PC into LR, offset by a value (current PC + 2, PC + 4 or PC + 8 depending on the exception) that causes the program to resume from the correct place on return.
2. Copies the CPSR into the appropriate SPSR.
3. Forces the CPSR mode bits to a value that depends on the exception.
4. Forces the PC to fetch the next instruction from the relevant exception vector.

The register r13 is also banked across exception modes to provide each exception handler with private stack pointer.

The Arm9EJ-S can also set the interrupt disable flags to prevent otherwise unmanageable nesting of exceptions.

When an exception has completed, the exception handler must move both the return value in the banked LR minus an offset to the PC and the SPSR to the CPSR. The offset value varies according to the type of exception. This action restores both PC and the CPSR.

The fast interrupt mode has seven private registers r8 to r14 (banked registers) to reduce or remove the requirement for register saving which minimizes the overhead of context switching.

The Prefetch Abort is one of the aborts that indicates that the current memory access cannot be completed. When a Prefetch Abort occurs, the Arm9EJ-S marks the prefetched instruction as invalid, but does not take the exception until the instruction reaches the Execute stage in the pipeline. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the abort does not take place.

The breakpoint (BKPT) instruction is a new feature of Arm9EJ-S that is destined to solve the problem of the Prefetch Abort. A breakpoint instruction operates as though the instruction caused a Prefetch Abort.
A breakpoint instruction does not cause the Arm9EJ-S to take the Prefetch Abort exception until the instruction reaches the Execute stage of the pipeline. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the breakpoint does not take place.

8.4.9 Arm Instruction Set Overview

The Arm instruction set is divided into:

- Branch instructions
- Data processing instructions
- Status register transfer instructions
- Load and Store instructions
- Coprocessor instructions
- Exception-generating instructions

Arm instructions can be executed conditionally. Every instruction contains a 4-bit condition code field (bits[31:28]).

© 2017 Microchip Technology Inc.

Table 8-2 gives the Arm instruction mnemonic list.

Table 8-2: Arm Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
RSB	Reverse Subtract	RSC	Reverse Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	TEQ	Test Equivalence
AND	Logical AND	BIC	Bit Clear
EOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
MUL	Multiply	MLA	Multiply Accumulate
SMULL	Sign Long Multiply	UMULL	Unsigned Long Multiply
SMLAL	Signed Long Multiply Accumulate	UMLAL	Unsigned Long Multiply Accumulate
MSR	Move to Status Register	MRS	Move From Status Register
B	Branch	BL	Branch and Link
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRSH	Load Signed Halfword		
LDRSB	Load Signed Byte		
LDRH	Load Half Word	STRH	Store Half Word
LDRB	Load Byte	STRB	Store Byte
LDRBT	Load Register Byte with Translation	STRBT	Store Register Byte with Translation
LDRT	Load Register with Translation	STRT	Store Register with Translation
LDM	Load Multiple	STM	Store Multiple
SWP	Swap Word	SWPB	Swap Byte
MCR	Move To Coprocessor	MRC	Move From Coprocessor
LDC	Load To Coprocessor	STC	Store From Coprocessor
CDP	Coprocessor Data Processing		

8.4.10 New Arm Instruction Set

Table 8-3: New Arm Instruction Mnemonic List

Mnemonic	Operation
BXJ	Branch and exchange to Java
BLX (1)	Branch, Link and exchange
SMLAxy	Signed Multiply Accumulate 16 * 16 bit
SMLAL	Signed Multiply Accumulate Long
SMLAWy	Signed Multiply Accumulate 32 * 16 bit
SMULxy	Signed Multiply 16 * 16 bit
SMULWy	Signed Multiply 32 * 16 bit
QADD	Saturated Add
QDADD	Saturated Add with Double
QSUB	Saturated subtract
QDSUB	Saturated Subtract with double

Mnemonic	Operation
MRRC	Move double from coprocessor
MCR2	Alternative move of Arm reg to coprocessor
MCRR	Move double to coprocessor
CDP2	Alternative Coprocessor Data Processing
BKPT	Breakpoint
PLD	Soft Preload, Memory prepare to load from address
STRD	Store Double
STC2	Alternative Store from Coprocessor
LDRD	Load Double
LDC2	Alternative Load to Coprocessor
CLZ	Count Leading Zeroes

Note 1: A Thumb BLX contains two consecutive Thumb instructions, and takes four cycles.

© 2017 Microchip Technology Inc.

8.4.11 Thumb Instruction Set Overview

The Thumb instruction set is a re-encoded subset of the Arm instruction set.

The Thumb instruction set is divided into:

- Branch instructions
- Data processing instructions
- Load and Store instructions
- Load and Store multiple instructions
- Exception-generating instruction

Table 8-4 gives the Thumb instruction mnemonic list.

Table 8-4: Thumb Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	NEG	Negate
AND	Logical AND	BIC	Bit Clear
EOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
LSL	Logical Shift Left	LSR	Logical Shift Right
ASR	Arithmetic Shift Right	ROR	Rotate Right
MUL	Multiply	BLX	Branch, Link, and Exchange
B	Branch	BL	Branch and Link
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRH	Load Half Word	STRH	Store Half Word
LDRB	Load Byte	STRB	Store Byte
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDMIA	Load Multiple	STMIA	Store Multiple
PUSH	Push Register to stack	POP	Pop Register from stack
BCC	Conditional Branch	BKPT	Breakpoint

8.5 CP15 Coprocessor

Coprocessor 15, or System Control Coprocessor CP15, is used to configure and control all the items in the list below:

- Arm9EJ-S
- Caches (ICache, DCache and write buffer)
- TCM
- MMU
- Other system options

To control these features, CP15 provides 16 additional registers. See Table 8-5.

Table 8-5: CP15 Registers

Register	Name	Read/Write
0	ID Code(1)	Read/Unpredictable
0	Cache type(1)	Read/Unpredictable
0	TCM status(1)	Read/Unpredictable
1	Control	Read/write
2	Translation Table Base	Read/write
3	Domain Access Control	Read/write
4	Reserved	None
5	Data fault Status(1)	Read/write
5	Instruction fault status(1)	Read/write
6	Fault Address	Read/write
7	Cache Operations	Read/Write
8	TLB operations	Unpredictable/Write
9	cache lockdown(2)	Read/write
9	TCM region	Read/write
10	TLB lockdown	Read/write
11	Reserved	None
12	Reserved	None
13	FCSE PID(1)	Read/write
13	Context ID(1)	Read/Write
14	Reserved	None
15	Test configuration	Read/Write

Note 1: Register locations 0,5, and 13 each provide access to more than one register. The register accessed depends on the value of the opcode_2 field.

2: Register location 9 provides access to more than one register. The register accessed depends on the value of the CRm field.

© 2017 Microchip Technology Inc.

8.5.1 CP15 Registers Access

CP15 registers can only be accessed in privileged mode by:

- MCR (Move to Coprocessor from Arm Register) instruction is used to write an Arm register to CP15.
- MRC (Move to Arm Register from Coprocessor) instruction is used to read the value of CP15 to an Arm register.

Other instructions like CDP, LDC, STC can cause an undefined instruction exception.

The assembler code for these instructions is:

MCR/MRC{cond} p15, opcode_1, Rd, CRn, CRm, opcode_2.

The MCR, MRC instructions bit pattern is shown below:

31	30	29	28	27	26	25	24
cond				1	1	1	0

23	22	21	20	19	18	17	16
opcode_1			L	CRn			

15	14	13	12	11	10	9	8
Rd				1	1	1	1

7	6	5	4	3	2	1	0
opcode_2			1	CRm			

CRm[3:0]: Specified Coprocessor Action

Determines specific coprocessor action. Its value is dependent on the CP15 register used. For details, refer to CP15 specific register behavior.

opcode_2[7:5]

Determines specific coprocessor operation code. By default, set to 0.

Rd[15:12]: Arm Register

Defines the Arm register whose value is transferred to the coprocessor. If R15 is chosen, the result is unpredictable.

CRn[19:16]: Coprocessor Register

Determines the destination coprocessor register.

L: Instruction Bit

0: MCR instruction

1: MRC instruction

opcode_1[23:20]: Coprocessor Code

Defines the coprocessor specific code. Value is c15 for CP15.

cond [31:28]: Condition

For more details, see Chapter 2 in Arm926EJ-S TRM.

8.6 Memory Management Unit (MMU)

The Arm926EJ-S processor implements an enhanced Arm architecture v5 MMU to provide virtual memory features required by operating systems like Symbian OS, WindowsCE, and Linux. These virtual memory features are memory access permission controls and virtual to physical address translations.

The Virtual Address generated by the CPU core is converted to a Modified Virtual Address (MVA) by the FCSE (Fast Context Switch Extension) using the value in CP15 register13. The MMU translates modified virtual addresses to physical addresses by using a single, two-level page table set stored in physical memory. Each entry in the set contains the access permissions and the physical address that correspond to the virtual address.

The first level translation tables contain 4096 entries indexed by bits [31:20] of the MVA. These entries contain a pointer to either a 1 MB section of physical memory along with attribute information (access permissions, domain, etc.) or an entry in the second level translation tables; coarse table and fine table.

The second level translation tables contain two subtables, coarse table and fine table. An entry in the coarse table contains a pointer to both large pages and small pages along with access permissions. An entry in the fine table contains a pointer to large, small and tiny pages.

Table 7 shows the different attributes of each page in the physical memory.

Table 8-6: Mapping Details

Mapping Name	Mapping Size	Access Permission By	Subpage Size
Section	1M byte	Section	-
Large Page	64K bytes	4 separated subpages	16K bytes
Small Page	4K bytes	4 separated subpages	1K byte
Tiny Page	1K byte	Tiny Page	-

The MMU consists of:

- Access control logic
- Translation Look-aside Buffer (TLB)
- Translation table walk hardware

8.6.1 Access Control Logic

The access control logic controls access information for every entry in the translation table. The access control logic checks two pieces of access information: domain and access permissions. The domain is the primary access control mechanism for a memory region; there are 16 of them. It defines the conditions necessary for an access to proceed. The domain determines whether the access permissions are used to qualify the access or whether they should be ignored.

The second access control mechanism is access permissions that are defined for sections and for large, small and tiny pages. Sections and tiny pages have a single set of access permissions whereas large and small pages can be associated with 4 sets of access permissions, one for each subpage (quarter of a page).

© 2017 Microchip Technology Inc.

8.6.2 Translation Look-aside Buffer (TLB)

The Translation Look-aside Buffer (TLB) caches translated entries and thus avoids going through the translation process every time. When the TLB contains an entry for the MVA (Modified Virtual Address), the access control logic determines if the access is permitted and outputs the appropriate physical address corresponding to the MVA. If access is not permitted, the MMU signals the CPU core to abort.

If the TLB does not contain an entry for the MVA, the translation table walk hardware is invoked to retrieve the translation information from the translation table in physical memory.

8.6.3 Translation Table Walk Hardware

The translation table walk hardware is a logic that traverses the translation tables located in physical memory, gets the physical address and access permissions and updates the TLB.

The number of stages in the hardware table walking is one or two depending whether the address is marked as a section-mapped access or a page-mapped access.

There are three sizes of page-mapped accesses and one size of section-mapped access. Page-mapped accesses are for large pages, small pages and tiny pages. The translation process always begins with a level one fetch. A section-mapped access requires only a level one fetch, but a page-mapped access requires an additional level two fetch. For further details on the MMU, refer to chapter 3 in Arm926EJ-S Technical Reference Manual.

8.6.4 MMU Faults

The MMU generates an abort on the following types of faults:

- Alignment faults (for data accesses only)
- Translation faults
- Domain faults
- Permission faults

The access control mechanism of the MMU detects the conditions that produce these faults. If the fault is a result of memory access, the MMU aborts the access and signals the fault to the CPU core.The MMU retains status and address information about faults generated by the data accesses in the data fault status register and fault address register. It also retains the status of faults generated by instruction fetches in the instruction fault status register.

The fault status register (register 5 in CP15) indicates the cause of a data or prefetch abort, and the domain number of the aborted access when it happens. The fault address register (register 6 in CP15) holds the MVA associated with the access that caused the Data Abort. For further details on MMU faults, refer to chapter 3 in Arm926EJ-S Technical Reference Manual.

8.7 Caches and Write Buffer

The Arm926EJ-S contains a 16KB Instruction Cache (ICache), a 16KB Data Cache (DCache), and a write buffer. Although the ICache and DCache share common features, each still has some specific mechanisms.

The caches (ICache and DCache) are four-way set associative, addressed, indexed and tagged using the Modified Virtual Address (MVA), with a cache line length of eight words with two dirty bits for the DCache. The ICache and DCache provide mechanisms for cache lock-down, cache pollution control, and line replacement.

A new feature is now supported by Arm926EJ-S caches called allocate on read-miss commonly known as wrapping. This feature enables the caches to perform critical word first cache refilling. This means that when a request for a word causes a read-miss, the cache performs an AHB access. Instead of loading the whole line (eight words), the cache loads the critical word first, so the processor can reach it quickly, and then the remaining words, no matter where the word is located in the line.

The caches and the write buffer are controlled by the CP15 register 1 (Control), CP15 register 7 (cache operations) and CP15 register 9 (cache lockdown).

8.7.1 Instruction Cache (ICache)

The ICache caches fetched instructions to be executed by the processor. The ICache can be enabled by writing 1 to I bit of the CP15 Register 1 and disabled by writing 0 to this same bit.

When the MMU is enabled, all instruction fetches are subject to translation and permission checks. If the MMU is disabled, all instructions fetches are cachable, no protection checks are made and the physical address is flat-mapped to the modified virtual address. With the MVA use disabled, context switching incurs ICache cleaning and/or invalidating.

When the ICache is disabled, all instruction fetches appear on external memory (AHB) (see Tables 4-1 and 4-2 in page 4-4 in Arm926EJ-S TRM).

On reset, the ICache entries are invalidated and the ICache is disabled. For best performance, ICache should be enabled as soon as possible after reset.

8.7.2 Data Cache (DCache) and Write Buffer

Arm926EJ-S includes a DCache and a write buffer to reduce the effect of main memory bandwidth and latency on data access performance. The operations of DCache and write buffer are closely connected.

8.7.2.1 DCache

The DCache needs the MMU to be enabled. All data accesses are subject to MMU permission and translation checks. Data accesses that are aborted by the MMU do not cause linefills or data accesses to appear on the AMBA ASB interface. If the MMU is disabled, all data accesses are noncachable, nonbufferable, with no protection checks, and appear on the AHB bus. All addresses are flat-mapped, VA = MVA = PA, which incurs DCache cleaning and/or invalidating every time a context switch occurs.

The DCache stores the Physical Address Tag (PA Tag) from which every line was loaded and uses it when writing modified lines back to external memory. This means that the MMU is not involved in write-back operations.

Each line (8 words) in the DCache has two dirty bits, one for the first four words and the other one for the second four words. These bits, if set, mark the associated half-lines as dirty. If the cache line is replaced due to a linefill or a cache clean operation, the dirty bits are used to decide whether all, half or none is written back to memory.

DCache can be enabled or disabled by writing either 1 or 0 to bit C in register 1 of CP15 (see Tables 4-3 and 4-4 on page 4-5 in Arm926EJ-S TRM).

The DCache supports write-through and write-back cache operations, selected by memory region using the C and B bits in the MMU translation tables.

The DCache contains an eight data word entry, single address entry write-back buffer used to hold write-back data for cache line eviction or cleaning of dirty cache lines.

The Write Buffer can hold up to 16 words of data and four separate addresses. DCache and Write Buffer operations are closely connected as their configuration is set in each section by the page descriptor in the MMU translation table.

8.7.2.2 Write Buffer

The Arm926EJ-S contains a write buffer that has a 16-word data buffer and a four- address buffer. The write buffer is used for all writes to a bufferable region, write-through region and write-back region. It also allows to avoid stalling the processor when writes to external memory are performed. When a store occurs, data is written to the write buffer at core speed (high speed). The write buffer then completes the store to external memory at bus speed (typically slower than the core speed). During this time, the Arm9EJ-S processor can preform other tasks.

DCache and Write Buffer support write-back and write-through memory regions, controlled by C and B bits in each section and page descriptor within the MMU translation tables.

8.7.2.3 Write-though Operation

When a cache write hit occurs, the DCache line is updated. The updated data is then written to the write buffer which transfers it to external memory.

When a cache write miss occurs, a line, chosen by round robin or another algorithm, is stored in the write buffer which transfers it to external memory.

8.7.2.4 Write-back Operation

When a cache write hit occurs, the cache line or half line is marked as dirty, meaning that its contents are not up-to-date with those in the external memory.

When a cache write miss occurs, a line, chosen by round robin or another algorithm, is stored in the write buffer which transfers it to external memory.

8.8 Bus Interface Unit

The Arm926EJ-S features a Bus Interface Unit (BIU) that arbitrates and schedules AHB requests. The BIU implements a multi-layer AHB, based on the AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system. This is achieved by using a more complex interconnection matrix and gives the benefit of increased overall bus bandwidth, and a more flexible system architecture.

© 2017 Microchip Technology Inc.

The multi-master bus architecture has a number of benefits:

- It allows the development of multi-master systems with an increased bus bandwidth and a flexible architecture.
- Each AHB layer becomes simple because it only has one master, so no arbitration or master-to-slave muxing is required. AHB layers, implementing AHB-Lite protocol, do not have to support request and grant, nor do they have to support retry and split transactions.
- The arbitration becomes effective when more than one master wants to access the same slave simultaneously.

8.8.1 Supported Transfers

The Arm926EJ-S processor performs all AHB accesses as single word, bursts of four words, or bursts of eight words. Any Arm9EJ-S core request that is not 1, 4, 8 words in size is split into packets of these sizes. Note that the Microchip bus is AHB-Lite protocol compliant, hence it does not support split and retry requests.

Table 8-7 gives an overview of the supported transfers and different kinds of transactions they are used for.

Table 8-7: Supported Transfers

HBurst[2:0]	Description	
SINGLE	Single transfer	Single transfer of word, half word, or byte: • data write (NCNB, NCB, WT, or WB that has missed in DCache) • data read (NCNB or NCB) • NC instruction fetch (prefetched and non-prefetched) • page table walk read
INCR4	Four-word incrementing burst	Half-line cache write-back, Instruction prefetch, if enabled. Four-word burst NCNB, NCB, WT, or WB write.
INCR8	Eight-word incrementing burst	Full-line cache write-back, eight-word burst NCNB, NCB, WT, or WB write.
WRAP8	Eight-word wrapping burst	Cache linefill

8.8.2 Thumb Instruction Fetches

All instructions fetches, regardless of the state of Arm9EJ-S core, are made as 32-bit accesses on the AHB. If the Arm9EJ-S is in Thumb state, then two instructions can be fetched at a time.

8.8.3 Address Alignment

The Arm926EJ-S BIU performs address alignment checking and aligns AHB addresses to the necessary boundary. 16-bit accesses are aligned to halfword boundaries, and 32-bit accesses are aligned to word boundaries.

9. Debug and Test

9.1 Description

The SAM9CN12 features a number of complementary debug and test capabilities. A common JTAG/ICE (In-Circuit Emulator) port is used for standard debugging functions, such as downloading code and single-stepping through programs. The Debug Unit provides a two-pin UART that can be used to upload an application into internal SRAM. It manages the interrupt handling of the internal COMMTX and COMMRX signals that trace the activity of the Debug Communication Channel.

A set of dedicated debug and test input/output pins gives direct access to these capabilities from a PC-based test environment.

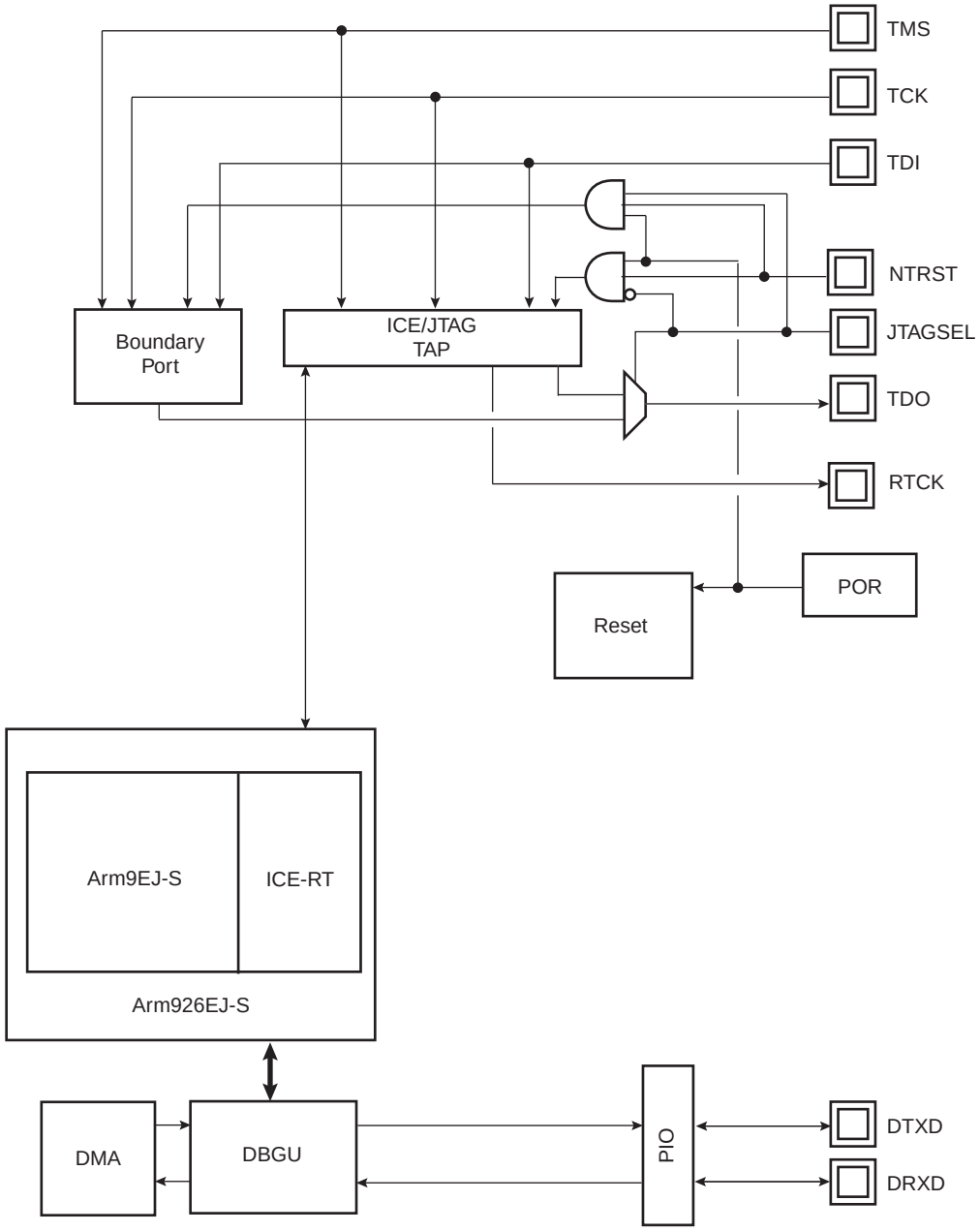
9.2 Embedded Characteristics

- Debug capabilities can be forbidden with a fuse bit.
- Arm926 Real-time In-circuit Emulator
  - Two real-time Watchpoint Units
  - Two Independent Registers: Debug Control Register and Debug Status Register
  - Test Access Port Accessible through JTAG Protocol
  - Debug Communications Channel
- Debug Unit
  - Two-pin UART
  - Debug Communication Channel Interrupt Handling
  - Chip ID Register
- IEEE1149.1 JTAG Boundary-scan on All Digital Pins

© 2017 Microchip Technology Inc.

9.3 Block Diagram

Figure 9-1: Debug and Test Block Diagram



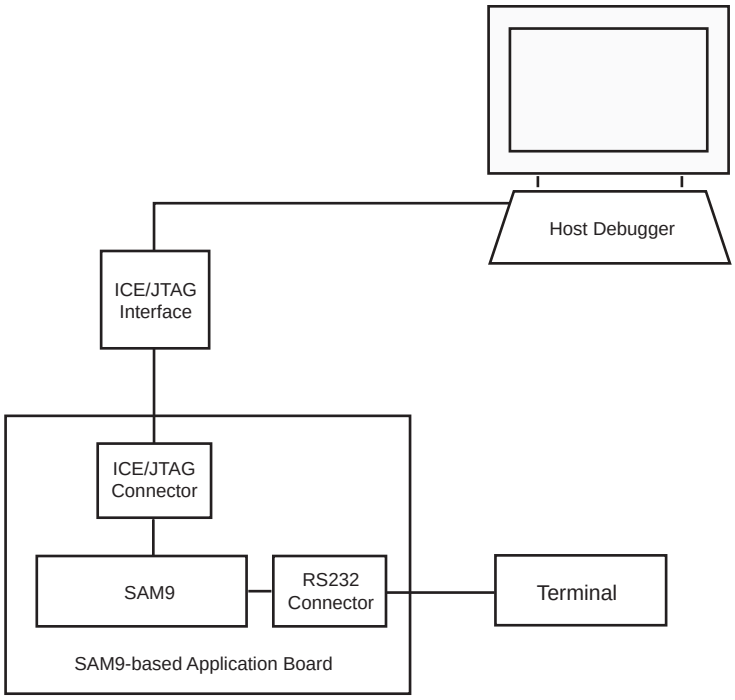
TAP: Test Access Port

9.4 Application Examples

9.4.1 Debug Environment

Figure 9-2 shows a complete debug environment example. The ICE/JTAG interface is used for standard debugging functions, such as downloading code and single-stepping through the program. A software debugger running on a personal computer provides the user interface for configuring a Trace Port interface utilizing the ICE/JTAG interface.

Figure 9-2: Application Debug and Trace Environment Example

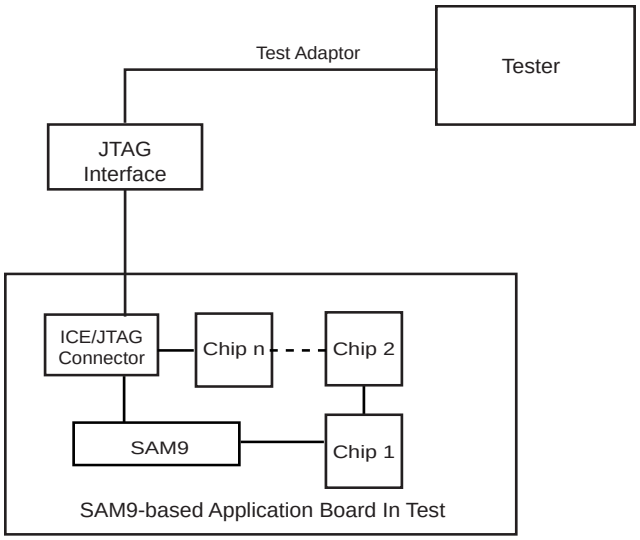


© 2017 Microchip Technology Inc.

9.4.2 Test Environment

Figure 9-3 shows a test environment example. Test vectors are sent and interpreted by the tester. In this example, the “board in test” is designed using a number of JTAG-compliant devices. These devices can be connected to form a single scan chain.

Figure 9-3: Application Test Environment Example



9.5 Debug and Test Pin Description

Table 9-1: Debug and Test Pin List

Pin Name	Function	Type	Active Level
Reset/Test			
NRST	Microcontroller Reset	Input/Output	Low
ICE and JTAG			
NTRST	Test Reset Signal	Input	Low
TCK	Test Clock	Input	
TDI	Test Data In	Input	
TDO	Test Data Out	Output	
TMS	Test Mode Select	Input	
RTCK	Returned Test Clock	Output	
JTAGSEL	JTAG Selection	Input	
Debug Unit			
DRXD	Debug Receive Data	Input	
DTXD	Debug Transmit Data	Output	

9.6 Functional Description

9.6.1 EmbeddedICE

The Arm9EJ-S EmbeddedICE-RT is supported via the ICE/JTAG port. It is connected to a host computer via an ICE interface. Debug support is implemented using an Arm9EJ-S core embedded within the Arm926EJ-S. The internal state of the Arm926EJ-S is examined through an ICE/JTAG port which allows instructions to be serially inserted into the pipeline of the core without using the external data bus. Therefore, when in debug state, a store-multiple (STM) can be inserted into the instruction pipeline. This exports the contents of the Arm9EJ-S registers. This data can be serially shifted out without affecting the rest of the system.

There are two scan chains inside the Arm9EJ-S processor which support testing, debugging, and programming of the EmbeddedICE-RT. The scan chains are controlled by the ICE/JTAG port.

EmbeddedICE mode is selected when JTAGSEL is low. It is not possible to switch directly between ICE and JTAG operations. A chip reset must be performed after JTAGSEL is changed.

For further details on the EmbeddedICE-RT, see the Arm document:

Arm9EJ-S Technical Reference Manual (DDI 0222A).

9.6.2 JTAG Signal Description

TMS is the Test Mode Select input which controls the transitions of the test interface state machine.

TDI is the Test Data Input line which supplies the data to the JTAG registers (Boundary Scan Register, Instruction Register, or other data registers).

TDO is the Test Data Output line which is used to serially output the data from the JTAG registers to the equipment controlling the test. It carries the sampled values from the boundary scan chain (or other JTAG registers) and propagates them to the next chip in the serial test circuit.

NTRST (optional in IEEE Standard 1149.1) is a Test-ReSeT input which is mandatory in Arm cores and used to reset the debug logic. On Microchip Arm926EJ-S-based cores, NTRST is a Power On Reset output. It is asserted on power on. If necessary, the user can also reset the debug logic with the NTRST pin assertion during 2.5 MCK periods.

TCK is the Test ClocK input which enables the test interface. TCK is pulsed by the equipment controlling the test and not by the tested device. It can be pulsed at any frequency. Note the maximum JTAG clock rate on Arm926EJ-S cores is 1/6th the clock of the CPU. This gives 5.45 kHz maximum initial JTAG clock rate for an Arm9E running from the 32.768 kHz slow clock.

RTCK is the Return Test Clock. Not an IEEE Standard 1149.1 signal added for a better clock handling by emulators. From some ICE Interface probes, this return signal can be used to synchronize the TCK clock and take not care about the given ratio between the ICE Interface clock and system clock equal to 1/6th. This signal is only available in JTAG ICE Mode and not in boundary scan mode.

9.6.3 Debug Unit

The Debug Unit provides a two-pin (DXRD and TXRD) USART that can be used for several debug and trace purposes and offers an ideal means for in-situ programming solutions and debug monitor communication. Moreover, the association with two peripheral data controller channels permits packet handling of these tasks with processor time reduced to a minimum.

The Debug Unit also manages the interrupt handling of the COMMTX and COMMRX signals that come from the ICE and that trace the activity of the Debug Communication Channel.The Debug Unit allows blockage of access to the system through the ICE interface.

A specific register, the Debug Unit Chip ID Register, gives information about the product version and its internal configuration.

For further details on the Debug Unit, see Section 23. “Debug Unit (DBGU)”.

9.6.4 IEEE 1149.1 JTAG Boundary Scan

IEEE 1149.1 JTAG Boundary Scan allows pin-level access independent of the device packaging technology.

IEEE 1149.1 JTAG Boundary Scan is enabled when JTAGSEL is high. The SAMPLE, EXTEST and BYPASS functions are implemented. In ICE debug mode, the Arm processor responds with a non-JTAG chip ID that identifies the processor to the ICE system. This is not IEEE 1149.1 JTAG-compliant.

It is not possible to switch directly between JTAG and ICE operations. A chip reset must be performed after JTAGSEL is changed.

A Boundary-scan Descriptor Language (BSDL) file is provided to set up test.

© 2017 Microchip Technology Inc.

9.6.5 JTAG ID Code Register

Access: Read-only

31	30	29	28	27	26	25	24
VERSION				PART NUMBER			
23	22	21	20	19	18	17	16
PART NUMBER							
15	14	13	12	11	10	9	8
PART NUMBER				MANUFACTURER IDENTITY			
7	6	5	4	3	2	1	0
MANUFACTURER IDENTITY							1

VERSION[31:28]: Product Version Number

Set to 0x0.

PART NUMBER[27:12]: Product Part Number

Product part Number is 0x05B3

MANUFACTURER IDENTITY[11:1]

Set to 0x01F.

Bit[0] required by IEEE Std. 1149.1.

Set to 0x1.

JTAG ID Code value is 0x05B3_003F.

10. Advanced Interrupt Controller (AIC)

10.1 Description

The Advanced Interrupt Controller (AIC) is an 8-level priority, individually maskable, vectored interrupt controller, providing handling of up to 32 interrupt sources. It is designed to substantially reduce the software and real-time overhead in handling internal and external interrupts.

The AIC drives the nFIQ (fast interrupt request) and the nIRQ (standard interrupt request) inputs of an Arm processor. Inputs of the AIC are either internal peripheral interrupts or external interrupts coming from the product's pins.

The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

Internal interrupt sources can be programmed to be level sensitive or edge triggered. External interrupt sources can be programmed to be positive-edge or negative-edge triggered or high-level or low-level sensitive.

The Fast Forcing feature redirects any internal or external interrupt source to provide a fast interrupt rather than a normal interrupt.

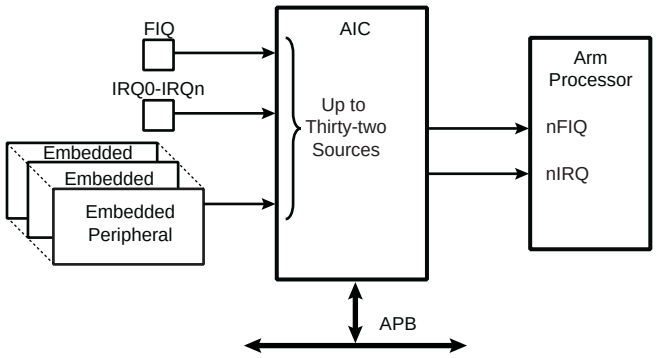
10.2 Embedded Characteristics

- Controls the Interrupt Lines (nIRQ and nFIQ) of an Arm Processor
- 32 Individually Maskable and Vectored Interrupt Sources
  - Source 0 is Reserved for the Fast Interrupt Input (FIQ)
  - Source 1 is Reserved for System Peripherals
  - Source 2 to Source 31 Control up to 30 Embedded Peripheral Interrupts or External Interrupts
  - Programmable Edge-triggered or Level-sensitive Internal Sources
  - Programmable Positive/Negative Edge-triggered or High/Low Level-sensitive External Sources
- 8-level Priority Controller
  - Drives the Normal Interrupt of the Processor
  - Handles Priority of the Interrupt Sources 1 to 31
  - Higher Priority Interrupts Can Be Served During Service of Lower Priority Interrupt
- Vectoring
  - Optimizes Interrupt Service Routine Branch and Execution
  - One 32-bit Vector Register per Interrupt Source
  - Interrupt Vector Register Reads the Corresponding Current Interrupt Vector
- Protect Mode
  - Easy Debugging by Preventing Automatic Operations when Protect Models Are Enabled
- Fast Forcing
  - Permits Redirecting any Normal Interrupt Source to the Fast Interrupt of the Processor
- General Interrupt Mask
  - Provides Processor Synchronization on Events Without Triggering an Interrupt
- Register Write Protection

© 2017 Microchip Technology Inc.

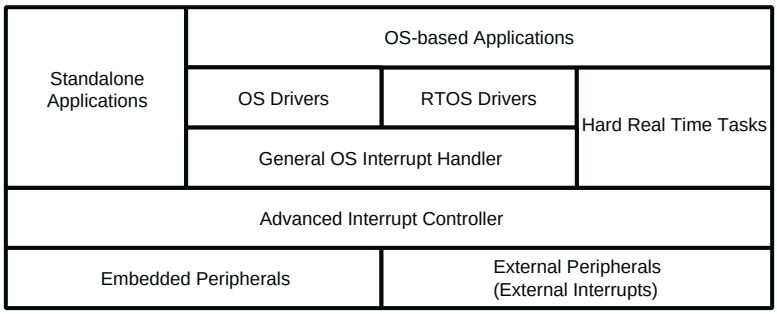
10.3 Block Diagram

Figure 10-1: Block Diagram



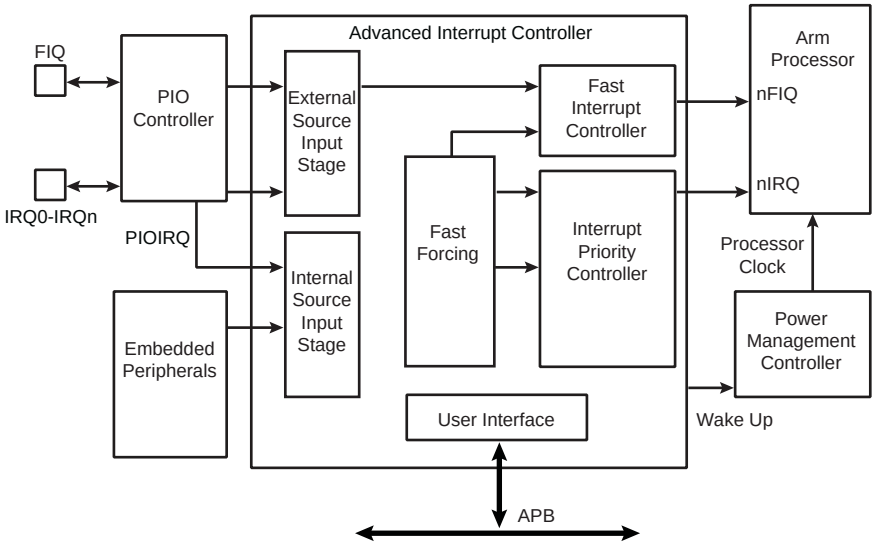
10.4 Application Block Diagram

Figure 10-2: Description of the Application Block



10.5 AIC Detailed Block Diagram

Figure 10-3: AIC Detailed Block Diagram



10.6 I/O Line Description

Table 10-1: I/O Line Description

Pin Name	Pin Description	Type
FIQ	Fast Interrupt	Input
IRQ0–IRQn	Interrupt 0–Interrupt n	Input

10.7 Product Dependencies

10.7.1 I/O Lines

The interrupt signals FIQ and IRQ0 to IRQn are normally multiplexed through the PIO controllers. Depending on the features of the PIO controller used in the product, the pins must be programmed in accordance with their assigned interrupt function. This is not applicable when the PIO controller used in the product is transparent on the input path.

Table 10-2: I/O Lines

Instance	Signal	I/O Line	Peripheral
AIC	FIQ	PC31	A
AIC	IRQ	PB18	A

10.7.2 Power Management

The Advanced Interrupt Controller is continuously clocked. The Power Management Controller has no effect on the Advanced Interrupt Controller behavior.

The assertion of the Advanced Interrupt Controller outputs, either nIRQ or nFIQ, wakes up the Arm processor while it is in Idle Mode. The General Interrupt Mask feature enables the AIC to wake up the processor without asserting the interrupt line of the processor, thus providing synchronization of the processor on an event.

10.7.3 Interrupt Sources

The Interrupt Source 0 is always located at FIQ. If the product does not feature an FIQ pin, the Interrupt Source 0 cannot be used.

The Interrupt Source 1 is always located at System Interrupt. This is the result of the OR-wiring of the system peripheral interrupt lines. When a system interrupt occurs, the service routine must first distinguish the cause of the interrupt. This is performed by reading successively the status registers of the above mentioned system peripherals.

The interrupt sources 2 to 31 can either be connected to the interrupt outputs of an embedded user peripheral or to external interrupt lines. The external interrupt lines can be connected directly, or through the PIO Controller.

The PIO Controllers are considered as user peripherals in the scope of interrupt handling. Accordingly, the PIO Controller interrupt lines are connected to the Interrupt Sources 2 to 31.

The peripheral identification defined at the product level corresponds to the interrupt source number (as well as the bit number controlling the clock of the peripheral). Consequently, to simplify the description of the functional operations and the user interface, the interrupt sources are named FIQ, SYS, and PID2 to PID31.

© 2017 Microchip Technology Inc.

10.8 Functional Description

10.8.1 Interrupt Source Control

10.8.1.1 Interrupt Source Mode

The AIC independently programs each interrupt source. The SRCTYPE field of the corresponding Source Mode Register (AIC_SMR) selects the interrupt condition of each source.

The internal interrupt sources wired on the interrupt outputs of the embedded peripherals can be programmed either in level-sensitive mode or in edge-triggered mode. The active level of the internal interrupts is not important for the user.

The external interrupt sources can be programmed either in high level-sensitive or low level-sensitive modes, or in positive edge-triggered or negative edge-triggered modes.

10.8.1.2 Interrupt Source Enabling

Each interrupt source, including the FIQ in source 0, can be enabled or disabled by using the command registers AIC_IECR (Interrupt Enable Command Register) and AIC_IDCR (Interrupt Disable Command Register). This set of registers conducts enabling or disabling in one instruction. The interrupt mask can be read in the Interrupt Mask Register (AIC_IMR). A disabled interrupt does not affect servicing of other interrupts.

10.8.1.3 Interrupt Clearing and Setting

All interrupt sources programmed to be edge-triggered (including the FIQ in source 0) can be individually set or cleared by writing respectively the Interrupt Set Command Register (AIC_ISCR) and the Interrupt Clear Command Register (AIC_ICCR). Clearing or setting interrupt sources programmed in level-sensitive mode has no effect.

The clear operation is perfunctory, as the software must perform an action to reinitialize the “memorization” circuitry activated when the source is programmed in edge-triggered mode. However, the set operation is available for auto-test or software debug purposes. It can also be used to execute an AIC implementation of a software interrupt.

The AIC features an automatic clear of the current interrupt when the AIC_IVR (Interrupt Vector Register) is read. Only the interrupt source being detected by the AIC as the current interrupt is affected by this operation (see Section 10.8.3.1 “Priority Controller”). The automatic clear reduces the operations required by the interrupt service routine entry code to reading the AIC_IVR. Note that the automatic interrupt clear is disabled if the interrupt source has the Fast Forcing feature enabled as it is considered uniquely as a FIQ source. (For further details, see “Fast Forcing”).

The automatic clear of the interrupt source 0 is performed when the FIQ Vector Register (AIC_FVR) is read.

10.8.1.4 Interrupt Status

For each interrupt, the AIC operation originates in the Interrupt Pending Register (AIC_IPR) and its mask in the AIC_IMR. The AIC_IPR enables the actual activity of the sources, whether masked or not.

The Interrupt Status Register (AIC_ISR) reads the number of the current interrupt (see “Priority Controller”) and the Core Interrupt Status Register (AIC_CISR) gives an image of the signals nIRQ and nFIQ driven on the processor.

Each status referred to above can be used to optimize the interrupt handling of the systems.

Figure 10-4: Internal Interrupt Source Input Stage

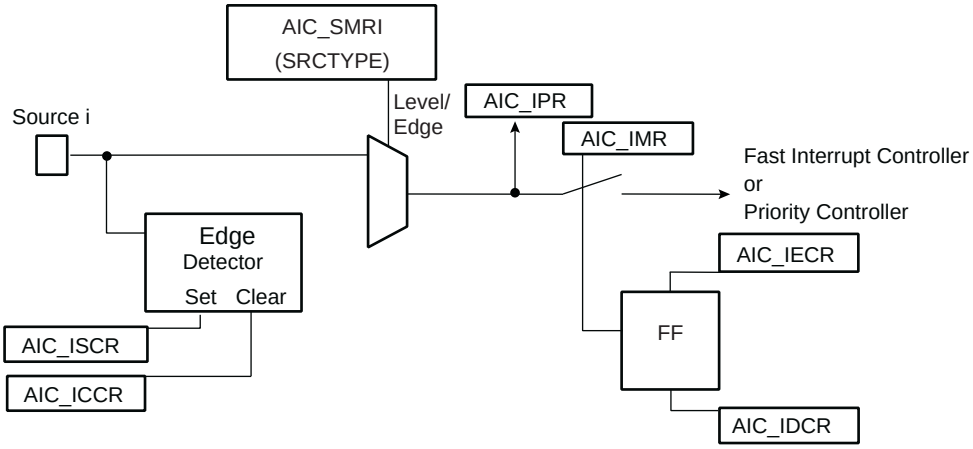
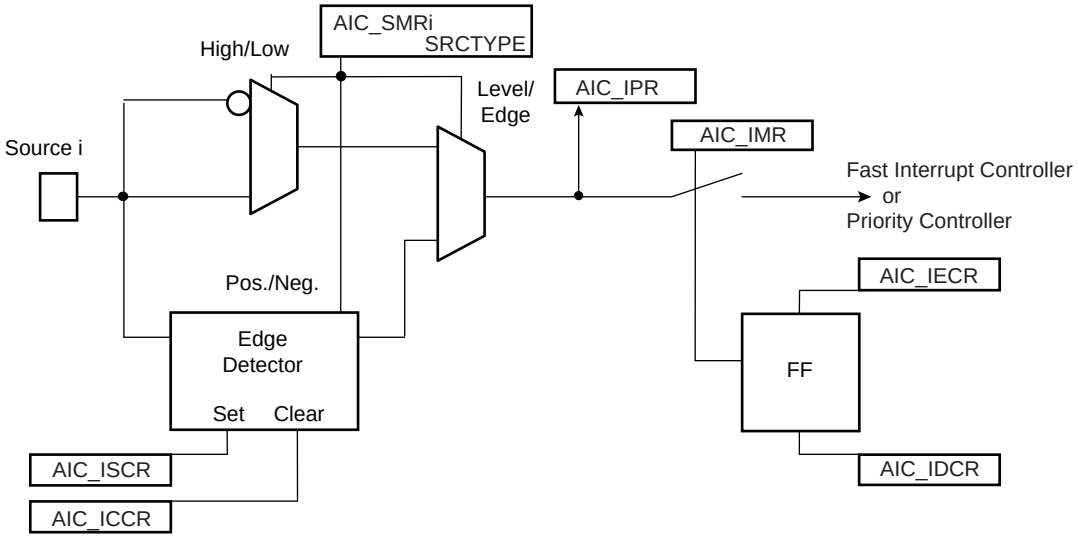


Figure 10-5: External Interrupt Source Input Stage



10.8.2 Interrupt Latencies

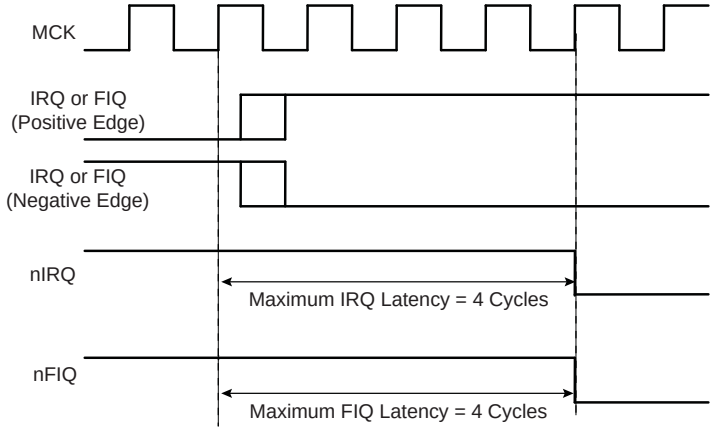
Global interrupt latencies depend on several parameters, including:

- The time the software masks the interrupts.
- Occurrence, either at the processor level or at the AIC level.
- The execution time of the instruction in progress when the interrupt occurs.
- The treatment of higher priority interrupts and the resynchronization of the hardware signals.

This section addresses only the hardware resynchronizations. It gives details of the latency times between the event on an external interrupt leading in a valid interrupt (edge or level) or the assertion of an internal interrupt source and the assertion of the nIRQ or nFIQ line on the processor. The resynchronization time depends on the programming of the interrupt source and on its type (internal or external). For the standard interrupt, resynchronization times are given assuming there is no higher priority in progress.

The PIO Controller multiplexing has no effect on the interrupt latencies of the external interrupt sources.

Figure 10-6: External Interrupt Edge Triggered Source



© 2017 Microchip Technology Inc.

Figure 10-7: External Interrupt Level Sensitive Source

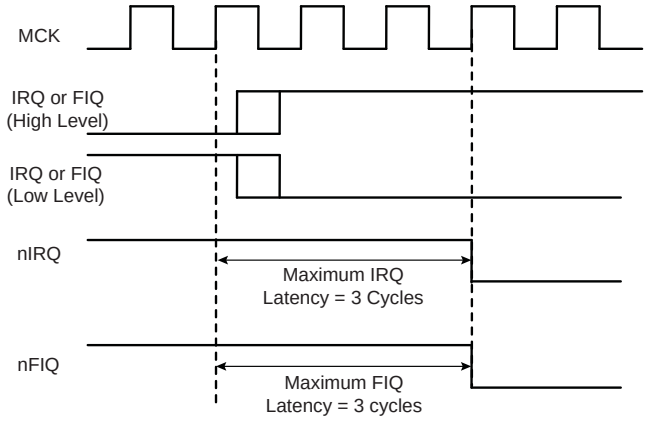


Figure 10-8: Internal Interrupt Edge Triggered Source

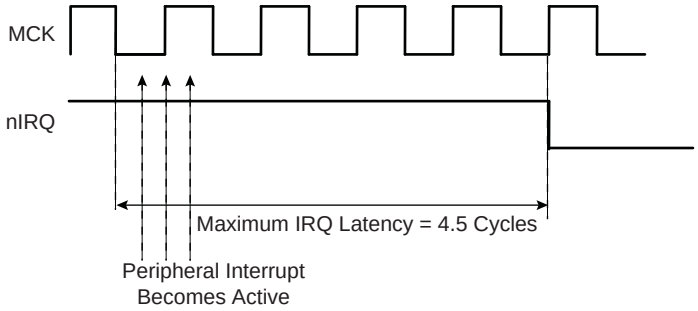
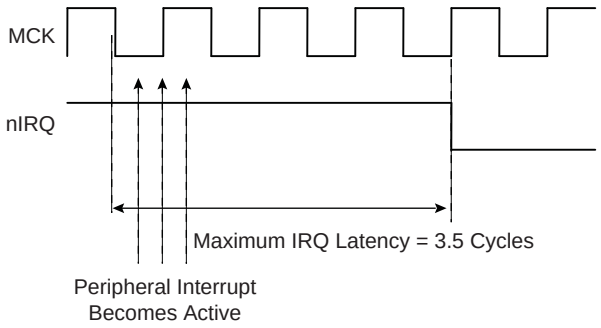


Figure 10-9: Internal Interrupt Level Sensitive Source



10.8.3 Normal Interrupt

10.8.3.1 Priority Controller

An 8-level priority controller drives the nIRQ line of the processor, depending on the interrupt conditions occurring on the interrupt sources 1 to 31 (except for those programmed in Fast Forcing).

Each interrupt source has a programmable priority level of 7 to 0, which is user-definable by writing the PRIOR field of the corresponding AIC_SMR. Level 7 is the highest priority and level 0 the lowest.

As soon as an interrupt condition occurs, as defined by the SRCTYPE field of the AIC_SMR, the nIRQ line is asserted. As a new interrupt condition might have happened on other interrupt sources since the nIRQ has been asserted, the priority controller determines the current interrupt at the time the Interrupt Vector Register (AIC_IVR) is read. **The read of AIC_IVR is the entry point of the interrupt handling** which allows the AIC to consider that the interrupt has been taken into account by the software.

The current priority level is defined as the priority level of the current interrupt.

If several interrupt sources of equal priority are pending and enabled when the AIC_IVR is read, the interrupt with the lowest interrupt source number is serviced first.

The nIRQ line can be asserted only if an interrupt condition occurs on an interrupt source with a higher priority. If an interrupt condition happens (or is pending) during the interrupt treatment in progress, it is delayed until the software indicates to the AIC the end of the current service by writing the End of Interrupt Command Register (AIC_EOICR). **The write of AIC_EOICR is the exit point of the interrupt handling**.

10.8.3.2 Interrupt Nesting

The priority controller utilizes interrupt nesting in order for the high priority interrupt to be handled during the service of lower priority interrupts. This requires the interrupt service routines of the lower interrupts to re-enable the interrupt at the processor level.

When an interrupt of a higher priority happens during an already occurring interrupt service routine, the nIRQ line is re-asserted. If the interrupt is enabled at the core level, the current execution is interrupted and the new interrupt service routine should read the AIC_IVR. At this time, the current interrupt number and its priority level are pushed into an embedded hardware stack, so that they are saved and restored when the higher priority interrupt servicing is finished and the AIC_EOICR is written.

The AIC is equipped with an 8-level wide hardware stack in order to support up to eight interrupt nestings pursuant to having eight priority levels.

10.8.3.3 Interrupt Vectoring

The interrupt handler addresses corresponding to each interrupt source can be stored in the registers AIC_SVR1 to AIC_SVR31 (Source Vector Register 1 to 31). When the processor reads AIC_IVR (Interrupt Vector Register), the value written into AIC_SVR corresponding to the current interrupt is returned.

This feature offers a way to branch in one single instruction to the handler corresponding to the current interrupt, as AIC_IVR is mapped at the absolute address 0xFFFFF100 and thus accessible from the Arm interrupt vector at address 0x00000018 through the following instruction:

```
LDR         PC,[PC,# -&F20]
```

When the processor executes this instruction, it loads the read value in AIC_IVR in its program counter, thus branching the execution on the correct interrupt handler.

This feature is often not used when the application is based on an operating system (either real time or not). Operating systems often have a single entry point for all the interrupts and the first task performed is to discern the source of the interrupt.

However, it is strongly recommended to port the operating system on AT91 products by supporting the interrupt vectoring. This can be performed by defining all the AIC_SVR of the interrupt source to be handled by the operating system at the address of its interrupt handler. When doing so, the interrupt vectoring permits a critical interrupt to transfer the execution on a specific very fast handler and not onto the operating system's general interrupt handler. This facilitates the support of hard real-time tasks (input/outputs of voice/audio buffers and software peripheral handling) to be handled efficiently and independently of the application running under an operating system.

10.8.3.4 Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the Arm processor, and especially the processor interrupt modes and the associated status bits.

It is assumed that:

- The Advanced Interrupt Controller has been programmed, Source Vector registers are loaded with corresponding interrupt service routine addresses and interrupts are enabled.
- The instruction at the Arm interrupt exception vector address is required to work with the vectoring

```
LDR PC, [PC, # -&F20]
```

When nIRQ is asserted, if the bit "I" of CPSR is 0, the sequence is as follows:

1. The CPSR is stored in SPSR_irq, the current value of the Program Counter is loaded in the Interrupt link register (R14_irq) and the Program Counter (R15) is loaded with 0x18. In the following cycle during fetch at address 0x1C, the Arm core adjusts R14_irq, decrementing it by four.
2. The Arm core enters Interrupt mode, if it has not already done so.
3. When the instruction loaded at address 0x18 is executed, the program counter is loaded with the value read in AIC_IVR. Reading the AIC_IVR has the following effects:
   - Sets the current interrupt to be the pending and enabled interrupt with the highest priority. The current level is the priority level of the current interrupt.
   - De-asserts the nIRQ line on the processor. Even if vectoring is not used, AIC_IVR must be read in order to de-assert nIRQ.
   - Automatically clears the interrupt, if it has been programmed to be edge-triggered.

© 2017 Microchip Technology Inc.

- Pushes the current level and the current interrupt number on to the stack.
- Returns the value written in the AIC_SVR corresponding to the current interrupt.

4. The previous step has the effect of branching to the corresponding interrupt service routine. This should start by saving the link register (R14_irq) and SPSR_IRQ. The link register must be decremented by four when it is saved if it is to be restored directly into the program counter at the end of the interrupt. For example, the instruction SUB PC, LR, #4 may be used.
5. Further interrupts can then be unmasked by clearing the “I” bit in CPSR, allowing re-assertion of the nIRQ to be taken into account by the core. This can happen if an interrupt with a higher priority than the current interrupt occurs.
6. The interrupt handler can then proceed as required, saving the registers that will be used and restoring them at the end. During this phase, an interrupt of higher priority than the current level will restart the sequence from step 1.

Note: If the interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase.

7. The “I” bit in CPSR must be set in order to mask interrupts before exiting to ensure that the interrupt is completed in an orderly manner.
8. The AIC_EOICR must be written in order to indicate to the AIC that the current interrupt is finished. This causes the current level to be popped from the stack, restoring the previous current level if one exists on the stack. If another interrupt is pending, with lower or equal priority than the old current level but with higher priority than the new current level, the nIRQ line is re-asserted, but the interrupt sequence does not immediately start because the “I” bit is set in the core. SPSR_irq is restored. Finally, the saved value of the link register is restored directly into the PC. This has the effect of returning from the interrupt to whatever was being executed before, and of loading the CPSR with the stored SPSR, masking or unmasking the interrupts depending on the state saved in SPSR_irq.

Note: The “I” bit in SPSR is significant. If it is set, it indicates that the Arm core was on the verge of masking an interrupt when the mask instruction was interrupted. Hence, when SPSR is restored, the mask instruction is completed (interrupt is masked).

10.8.4 Fast Interrupt

10.8.4.1 Fast Interrupt Source

The interrupt source 0 is the only source which can raise a fast interrupt request to the processor except if Fast Forcing is used. The interrupt source 0 is generally connected to a FIQ pin of the product, either directly or through a PIO Controller.

10.8.4.2 Fast Interrupt Control

The fast interrupt logic of the AIC has no priority controller. The mode of interrupt source 0 is programmed with the AIC_SMR0 and the field PRIOR of this register is not used even if it reads what has been written. The field SRCTYPE of AIC_SMR0 enables programming the fast interrupt source to be positive-edge triggered or negative-edge triggered or high-level sensitive or low-level sensitive

Writing 0x1 in the AIC_IECR and AIC_IDCR respectively enables and disables the fast interrupt. The bit 0 of AIC_IMR indicates whether the fast interrupt is enabled or disabled.

10.8.4.3 Fast Interrupt Vectoring

The fast interrupt handler address can be stored in AIC_SVR0 (Source Vector Register 0). The value written into this register is returned when the processor reads AIC_FVR. This offers a way to branch in one single instruction to the interrupt handler, as AIC_FVR is mapped at the absolute address 0xFFFFF104 and thus accessible from the Arm fast interrupt vector at address 0x0000001C through the following instruction:

```
LDR         PC,[PC,# -&F20]
```

When the processor executes this instruction it loads the value read in AIC_FVR in its program counter, thus branching the execution on the fast interrupt handler. It also automatically performs the clear of the fast interrupt source if it is programmed in edge-triggered mode.

10.8.4.4 Fast Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the Arm processor, and especially the processor interrupt modes and associated status bits.

It is assumed that:

- The Advanced Interrupt Controller has been programmed, AIC_SVR0 is loaded with the fast interrupt service routine address, and the interrupt source 0 is enabled.
- The Instruction at address 0x1C (FIQ exception vector address) is required to vector the fast interrupt:
- LDR PC, [PC, # -&F20]
- The user does not need nested fast interrupts.

When nFIQ is asserted, if the bit “F” of CPSR is 0, the sequence is:

1. The CPSR is stored in SPSR_fiq, the current value of the program counter is loaded in the FIQ link register (R14_FIQ) and the program counter (R15) is loaded with 0x1C. In the following cycle, during fetch at address 0x20, the Arm core adjusts R14_fiq, decrementing it by four.
2. The Arm core enters FIQ mode.
3. When the instruction loaded at address 0x1C is executed, the program counter is loaded with the value read in AIC_FVR. Reading the AIC_FVR has effect of automatically clearing the fast interrupt, if it has been programmed to be edge triggered. In this case only, it de-asserts the nFIQ line on the processor.
4. The previous step enables branching to the corresponding interrupt service routine. It is not necessary to save the link register R14_fiq and SPSR_fiq if nested fast interrupts are not needed.
5. The Interrupt Handler can then proceed as required. It is not necessary to save registers R8 to R13 because FIQ mode has its own dedicated registers and the user R8 to R13 are banked. The other registers, R0 to R7, must be saved before being used, and restored at the end (before the next step). Note that if the fast interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase in order to de-assert the interrupt source 0.
6. Finally, the Link Register R14_fiq is restored into the PC after decrementing it by four (with instruction SUB PC, LR, #4 for example). This has the effect of returning from the interrupt to whatever was being executed before, loading the CPSR with the SPSR and masking or unmasking the fast interrupt depending on the state saved in the SPSR.

Note: The “F” bit in SPSR is significant. If it is set, it indicates that the Arm core was just about to mask FIQ interrupts when the mask instruction was interrupted. Hence when the SPSR is restored, the interrupted instruction is completed (FIQ is masked).

Another way to handle the fast interrupt is to map the interrupt service routine at the address of the Arm vector 0x1C. This method does not use the vectoring, so that reading AIC_FVR must be performed at the very beginning of the handler operation. However, this method saves the execution of a branch instruction.

10.8.4.5 Fast Forcing

The Fast Forcing feature of the advanced interrupt controller provides redirection of any normal Interrupt source on the fast interrupt controller.

Fast Forcing is enabled or disabled by writing to the Fast Forcing Enable Register (AIC_FFER) and the Fast Forcing Disable Register (AIC_FFDR). Writing to these registers results in an update of the Fast Forcing Status Register (AIC_FFSR) that controls the feature for each internal or external interrupt source.

When Fast Forcing is disabled, the interrupt sources are handled as described in the previous pages.

When Fast Forcing is enabled, the edge/level programming and, in certain cases, edge detection of the interrupt source is still active but the source cannot trigger a normal interrupt to the processor and is not seen by the priority handler.

If the interrupt source is programmed in level-sensitive mode and an active level is sampled, Fast Forcing results in the assertion of the nFIQ line to the core.

If the interrupt source is programmed in edge-triggered mode and an active edge is detected, Fast Forcing results in the assertion of the nFIQ line to the core.

The Fast Forcing feature does not affect the Source 0 pending bit in the AIC_IPR.

The AIC_FVR reads the contents of AIC_SVR0, whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the Fast Forcing feature is used and the interrupt source should be cleared by writing to the AIC_ICCR.

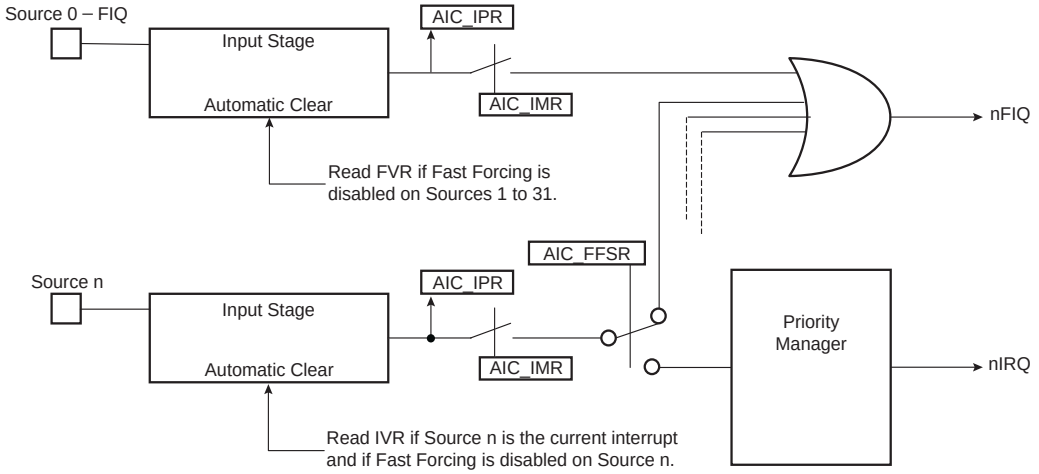
All enabled and pending interrupt sources that have the Fast Forcing feature enabled and that are programmed in edge-triggered mode must be cleared by writing to the AIC_ICCR. In doing so, they are cleared independently and thus lost interrupts are prevented.

The read of AIC_IVR does not clear the source that has the Fast Forcing feature enabled.

The source 0, reserved to the fast interrupt, continues operating normally and becomes one of the Fast Interrupt sources.

© 2017 Microchip Technology Inc.

Figure 10-10: Fast Forcing



10.8.5 Protect Mode

The Protect Mode permits reading the Interrupt Vector Register without performing the associated automatic operations. This is necessary when working with a debug system. When a debugger, working either with a Debug Monitor or the Arm processor's ICE, stops the applications and updates the opened windows, it might read the AIC User Interface and thus the AIC_IVR. This has undesirable consequences:

- If an enabled interrupt with a higher priority than the current one is pending, it is stacked.
- If there is no enabled pending interrupt, the spurious vector is returned.

In either case, an End of Interrupt command is necessary to acknowledge and to restore the context of the AIC. This operation is generally not performed by the debug system as the debug system would become strongly intrusive and cause the application to enter an undesired state.

This is avoided by using the Protect Mode. Writing a one to the PROT bit in the Debug Control Register (AIC_DCR) enables the Protect Mode.

When the Protect Mode is enabled, the AIC performs interrupt stacking only when a write access is performed on the AIC_IVR. Therefore, the Interrupt Service Routines must write (arbitrary data) to the AIC_IVR just after reading it. The new context of the AIC, including the value of the AIC_ISR, is updated with the current interrupt only when AIC_IVR is written.

An AIC_IVR read on its own (e.g., by a debugger), modifies neither the AIC context nor the AIC_ISR. Extra AIC_IVR reads perform the same operations. However, it is recommended to not stop the processor between the read and the write of AIC_IVR of the interrupt service routine to make sure the debugger does not modify the AIC context.

To summarize, in normal operating mode, the read of AIC_IVR performs the following operations within the AIC:

1. Calculates active interrupt (higher than current or spurious).
2. Determines and returns the vector of the active interrupt.
3. Memorizes the interrupt.
4. Pushes the current priority level onto the internal stack.
5. Acknowledges the interrupt.

However, while the Protect Mode is activated, only operations 1 to 3 are performed when AIC_IVR is read. Operations 4 and 5 are only performed by the AIC when AIC_IVR is written.

Software that has been written and debugged using the Protect Mode runs correctly in Normal Mode without modification. However, in Normal Mode the AIC_IVR write has no effect and can be removed to optimize the code.

10.8.6 Spurious Interrupt

The AIC features protection against spurious interrupts. A spurious interrupt is defined as being the assertion of an interrupt source long enough for the AIC to assert the nIRQ, but no longer present when AIC_IVR is read. This is most prone to occur when:

- An external interrupt source is programmed in level-sensitive mode and an active level occurs for only a short time.
- An internal interrupt source is programmed in level sensitive and the output signal of the corresponding embedded peripheral is activated for a short time (as in the case for the Watchdog).
- An interrupt occurs just a few cycles before the software begins to mask it, thus resulting in a pulse on the interrupt source.

The AIC detects a spurious interrupt at the time the AIC_IVR is read while no enabled interrupt source is pending. When this happens, the AIC returns the value stored by the programmer in the Spurious Vector Register (AIC_SPU). The programmer must store the address of a spurious interrupt handler in AIC_SPU as part of the application, to enable an as fast as possible return to the normal execution flow. This handler writes in AIC_EOICR and performs a return from interrupt.

10.8.7 General Interrupt Mask

The AIC features a General Interrupt Mask bit (GMSK in AIC_DCR) to prevent interrupts from reaching the processor. Both the nIRQ and the nFIQ lines are driven to their inactive state if GMSK is set. However, this mask does not prevent waking up the processor if it has entered Idle Mode. This function facilitates synchronizing the processor on a next event and, as soon as the event occurs, performs subsequent operations without having to handle an interrupt. It is strongly recommended to use this mask with caution.

10.8.8 Register Write Protection

To prevent any single software error from corrupting AIC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the AIC Write Protection Mode Register (AIC_WPMR).

If a write access to a write-protected register is detected, the WPVS bit in the AIC Write Protection Status Register (AIC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the AIC_WPSR.

The following registers can be write-protected:

- AIC Source Mode Register
- AIC Source Vector Register
- AIC Spurious Interrupt Vector Register
- AIC Debug Control Register

© 2017 Microchip Technology Inc.

10.9 Advanced Interrupt Controller (AIC) User Interface

The AIC is mapped at the address 0xFFFFF000. It has a total 4 Kbyte addressing space. This permits the vectoring feature, as the PC-relative load/store instructions of the Arm processor support only a ± 4 Kbyte offset.

Table 10-3: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Source Mode Register 0	AIC_SMR0	Read/Write	0x0
0x04	Source Mode Register 1	AIC_SMR1	Read/Write	0x0
...	...	...	...	...
0x7C	Source Mode Register 31	AIC_SMR31	Read/Write	0x0
0x80	Source Vector Register 0	AIC_SVR0	Read/Write	0x0
0x84	Source Vector Register 1	AIC_SVR1	Read/Write	0x0
...	...	...	...	...
0xFC	Source Vector Register 31	AIC_SVR31	Read/Write	0x0
0x100	Interrupt Vector Register	AIC_IVR	Read-only	0x0
0x104	FIQ Vector Register	AIC_FVR	Read-only	0x0
0x108	Interrupt Status Register	AIC_ISR	Read-only	0x0
0x10C	Interrupt Pending Register(2)	AIC_IPR	Read-only	0x0(1)
0x110	Interrupt Mask Register(2)	AIC_IMR	Read-only	0x0
0x114	Core Interrupt Status Register	AIC_CISR	Read-only	0x0
0x118–0x11C	Reserved	–	–	–
0x120	Interrupt Enable Command Register(2)	AIC_IECR	Write-only	–
0x124	Interrupt Disable Command Register(2)	AIC_IDCR	Write-only	–
0x128	Interrupt Clear Command Register(2)	AIC_ICCR	Write-only	–
0x12C	Interrupt Set Command Register(2)	AIC_ISCR	Write-only	–
0x130	End of Interrupt Command Register	AIC_EOICR	Write-only	–
0x134	Spurious Interrupt Vector Register	AIC_SPU	Read/Write	0x0
0x138	Debug Control Register	AIC_DCR	Read/Write	0x0
0x13C	Reserved	–	–	–
0x140	Fast Forcing Enable Register(2)	AIC_FFER	Write-only	–
0x144	Fast Forcing Disable Register(2)	AIC_FFDR	Write-only	–
0x148	Fast Forcing Status Register(2)	AIC_FFSR	Read-only	0x0
0x14C–0x1E0	Reserved	–	–	–
0x1E4	Write Protection Mode Register	AIC_WPMR	Read/Write	0x0
0x1E8	Write Protection Status Register	AIC_WPSR	Read-only	0x0
0x1EC–0x1FC	Reserved	–	–	–

Note 1: The reset value of this register depends on the level of the external interrupt source. All other sources are cleared at reset, thus not pending.

2: PID2...PID31 bit fields refer to the identifiers as defined in Section 7.2 “Peripheral Identifiers”.

10.9.1 AIC Source Mode Register

Name:AIC_SMR0..AIC_SMR31

Address:0xFFFFF000

AccessRead/Write

Reset:0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	SRCTYPE		–	–	PRIOR		

This register can only be written if the WPEN bit is cleared in the AIC Write Protection Mode Register.

PRIOR: Priority Level

The priority level is programmable from 0 (lowest priority) to 7 (highest priority).

The priority level is not used for the FIQ in AIC_SMR0.

SRCTYPE: Interrupt Source Type

The active level or edge is not programmable for the internal interrupt sources.

Value	Name	Description
0x0	INT_LEVEL_SENSITIVE	High level Sensitive for internal source Low level Sensitive for external source
0x1	INT_EDGE_TRIGGERED	Positive edge triggered for internal source Negative edge triggered for external source
0x2	EXT_HIGH_LEVEL	High level Sensitive for internal source High level Sensitive for external source
0x3	EXT_POSITIVE_EDGE	Positive edge triggered for internal source Positive edge triggered for external source

© 2017 Microchip Technology Inc.

10.9.2 AIC Source Vector Register

Name:AIC_SVR0..AIC_SVR31

Address:0xFFFFF080

Access: Read/Write

Reset:0x0

31	30	29	28	27	26	25	24
VECTOR							
23	22	21	20	19	18	17	16
VECTOR							
15	14	13	12	11	10	9	8
VECTOR							
7	6	5	4	3	2	1	0
VECTOR							

This register can only be written if the WPEN bit is cleared in the AIC Write Protection Mode Register.

VECTOR: Source Vector

The user may store in these registers the addresses of the corresponding handler for each interrupt source.

10.9.3 AIC Interrupt Vector Register

Name: AIC_IVR

Address:0xFFFFF100

Access: Read-only

Reset: 0x0

31	30	29	28	27	26	25	24
IRQV							
23	22	21	20	19	18	17	16
IRQV							
15	14	13	12	11	10	9	8
IRQV							
7	6	5	4	3	2	1	0
IRQV							

IRQV: Interrupt Vector Register

The Interrupt Vector Register contains the vector programmed by the user in the Source Vector Register corresponding to the current interrupt.

The Source Vector Register is indexed using the current interrupt number when the Interrupt Vector Register is read.

When there is no current interrupt, the Interrupt Vector Register reads the value stored in AIC_SPU.

© 2017 Microchip Technology Inc.

10.9.4 AIC FIQ Vector Register

Name: AIC_FVR

Address:0xFFFFF104

Access: Read-only

Reset:0x0

31	30	29	28	27	26	25	24
FIQV							

23	22	21	20	19	18	17	16
FIQV							

15	14	13	12	11	10	9	8
FIQV							

7	6	5	4	3	2	1	0
FIQV							

FIQV: FIQ Vector Register

The FIQ Vector Register contains the vector programmed by the user in the Source Vector Register 0. When there is no fast interrupt, the FIQ Vector Register reads the value stored in AIC_SPU.

10.9.5 AIC Interrupt Status Register

Name: AIC_ISR

Address:0xFFFFF108

Access: Read-only

Reset:0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	IRQID				

IRQID: Current Interrupt Identifier

The Interrupt Status Register returns the current interrupt source number.

© 2017 Microchip Technology Inc.

10.9.6 AIC Interrupt Pending Register

Name: AIC_IPR

Address:0xFFFFF10C

Access: Read-only

Reset: 0x0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

FIQ: Interrupt Pending

0: Corresponding interrupt is not pending.

1: Corresponding interrupt is pending.

SYS: Interrupt Pending

0: Corresponding interrupt is not pending.

1: Corresponding interrupt is pending.

PID2–PID31: Interrupt Pending

0: Corresponding interrupt is not pending.

1: Corresponding interrupt is pending.

10.9.7 AIC Interrupt Mask Register

Name:AIC_IMR

Address:0xFFFFF110

Access: Read-only

Reset:0x0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

FIQ: Interrupt Mask

0: Corresponding interrupt is disabled.

1: Corresponding interrupt is enabled.

SYS: Interrupt Mask

0: Corresponding interrupt is disabled.

1: Corresponding interrupt is enabled.

PID2–PID31: Interrupt Mask

0: Corresponding interrupt is disabled.

1: Corresponding interrupt is enabled.

© 2017 Microchip Technology Inc.

10.9.8 AIC Core Interrupt Status Register

Name: AIC_CISR

Address:0xFFFFF114

Access: Read-only

Reset:0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	NIRQ	NFIQ

NFIQ: NFIQ Status

0: nFIQ line is deactivated.

1: nFIQ line is active.

NIRQ: NIRQ Status

0: nIRQ line is deactivated.

1: nIRQ line is active.

10.9.9 AIC Interrupt Enable Command Register

Name: AIC_IECR

Address:0xFFFFF120

Access:Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

FIQ: Interrupt Enable

0: No effect.

1: Enables corresponding interrupt.

SYS: Interrupt Enable

0: No effect.

1: Enables corresponding interrupt.

PID2–PID31: Interrupt Enable

0: No effect.

1: Enables corresponding interrupt.

© 2017 Microchip Technology Inc.

10.9.10 AIC Interrupt Disable Command Register

Name: AIC_IDCR

Address:0xFFFFF124

Access:Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

FIQ: Interrupt Disable

0: No effect.

1: Disables corresponding interrupt.

SYS: Interrupt Disable

0: No effect.

1: Disables corresponding interrupt.

PID2–PID31: Interrupt Disable

0: No effect.

1: Disables corresponding interrupt.

10.9.11 AIC Interrupt Clear Command Register

Name:AIC_ICCR

Address:0xFFFFF128

Access:Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

FIQ: Interrupt Clear

0: No effect.

1: Clears corresponding interrupt.

SYS: Interrupt Clear

0: No effect.

1: Clears corresponding interrupt.

PID2–PID31: Interrupt Clear

0: No effect.

1: Clears corresponding interrupt.

© 2017 Microchip Technology Inc.

10.9.12 AIC Interrupt Set Command Register

Name: AIC_ISCR

Address:0xFFFFF12C

Access:Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

FIQ: Interrupt Set

0: No effect.

1: Sets corresponding interrupt.

SYS: Interrupt Set

0: No effect.

1: Sets corresponding interrupt.

PID2–PID31: Interrupt Set

0: No effect.

1: Sets corresponding interrupt.

10.9.13 AIC End of Interrupt Command Register

Name:AIC_EOICR

Address:0xFFFFF130

Access:Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	ENDIT

ENDIT: Interrupt Processing Complete Command

The End of Interrupt Command Register is used by the interrupt routine to indicate that the interrupt treatment is complete. Any value can be written because it is only necessary to make a write to this register location to signal the end of interrupt treatment.

© 2017 Microchip Technology Inc.

10.9.14 AIC Spurious Interrupt Vector Register

Name:AIC_SPU

Address:0xFFFFF134

Access:Read/Write

Reset: 0x0

31	30	29	28	27	26	25	24
SIVR							
23	22	21	20	19	18	17	16
SIVR							
15	14	13	12	11	10	9	8
SIVR							
7	6	5	4	3	2	1	0
SIVR							

This register can only be written if the WPEN bit is cleared in the AIC Write Protection Mode Register.

SIVR: Spurious Interrupt Vector Register

The user may store the address of a spurious interrupt handler in this register. The written value is returned in AIC_IVR in case of a spurious interrupt and in AIC_FVR in case of a spurious fast interrupt.

10.9.15 AIC Debug Control Register

Name:AIC_DCR

Address:0xFFFFF138

Access:Read/Write

Reset:0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	GMSK	PROT

This register can only be written if the WPEN bit is cleared in the AIC Write Protection Mode Register.

PROT: Protection Mode

0: The Protection Mode is disabled.

1: The Protection Mode is enabled.

GMSK: General Interrupt Mask

0: The nIRQ and nFIQ lines are normally controlled by the AIC.

1: The nIRQ and nFIQ lines are tied to their inactive state.

© 2017 Microchip Technology Inc.

10.9.16 AIC Fast Forcing Enable Register

Name:AIC_FFER

Address:0xFFFFF140

Access:Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

SYS: Fast Forcing Enable

0: No effect.

1: Enables the Fast Forcing feature on the corresponding interrupt.

PID2–PID31: Fast Forcing Enable

0: No effect.

1: Enables the Fast Forcing feature on the corresponding interrupt.

10.9.17 AIC Fast Forcing Disable Register

Name:AIC_FFDR

Address:0xFFFFF144

Access:Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

SYS: Fast Forcing Disable

0: No effect.

1: Disables the Fast Forcing feature on the corresponding interrupt.

PID2–PID31: Fast Forcing Disable

0: No effect.

1: Disables the Fast Forcing feature on the corresponding interrupt.

© 2017 Microchip Technology Inc.

10.9.18 AIC Fast Forcing Status Register

Name:AIC_FFSR

Address:0xFFFFF148

Access:Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

SYS: Fast Forcing Status

0: The Fast Forcing feature is disabled on the corresponding interrupt.

1: The Fast Forcing feature is enabled on the corresponding interrupt.

PID2–PID31: Fast Forcing Status

0: The Fast Forcing feature is disabled on the corresponding interrupt.

1: The Fast Forcing feature is enabled on the corresponding interrupt.

10.9.19 AIC Write Protection Mode Register

Name:AIC_WPMR

Address:0xFFFFF1E4

Access: Read/Write

Reset: See Table 10-3

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPEN

WPEN: Write Protection Enable

0: Disables write protection if WPKEY corresponds to 0x414943 (“AIC” in ASCII).

1: Enables write protection if WPKEY corresponds to 0x414943 (“AIC” in ASCII).

See Section 10.8.8 “Register Write Protection” for list of write-protected registers.

WPKEY: Write Protection Key

Value	Name	Description
0x414943	PASSWD	Writing any other value in this field aborts the write operation of bit WPEN. Always reads as 0.

© 2017 Microchip Technology Inc.

10.9.20 AIC Write Protection Status Register

Name:AIC_WPSR

Address:0xFFFFF1E8

Access: Read-only

Reset: See Table 10-3

31	30	29	28	27	26	25	24
—	—	—	—	—	—	—	—

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPVS

WPVS: Write Protection Violation Status

0: No write protection violation has occurred since the last read of the AIC_WPSR.

1: A write protection violation has occurred since the last read of the AIC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

WPVSRC: Write Protection Violation Source

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

11. Boot Strategies

11.1 SAM9CN12 Only

The SAM9CN12 embeds a Secure Boot allowing firmware stored in external Non-Volatile Memory to be protected.

The content of the external NVM is encrypted and signed based using a 256-bit AES algorithm. Prior to booting from the externally stored firmware, the secure boot will authenticate the firmware, decrypt it and store it in on-chip memory. Access to the on-chip memory is prevented and the maximum size of the firmware should not exceed 24 Kbytes. The programming of the external memory can only be done by the SAM9CN12 using a unique key stored in the on-chip OTP memory.

Herewith the software is uniquely linked to each SAM9CN12 device. A direct copy of the NVM memory will not run on another SAM9CN12 device, improving the firmware protection even further.

For software development the user should use the SAM9CN11 without secure boot and full access to on-chip memory for debug. Once the firmware development has been completed, the SAM9CN11 should be replaced by the SAM9CN12 and programmed via USB with the secure SAM Boot Assistant (SAM-BA®).

Refer to the application note *"Understanding and Using SAM9CN12 Secure Bootloader"*, literature No. 11196, for more details (NDA required).

11.2 SAM9CN11 and SAM9N12 Only

The system always boots at address 0x0. To ensure maximum boot possibilities, the memory layout can be changed thanks to the BMS pin. This allows the user to layout the ROM or an external memory to 0x0. The sampling of the BMS pin is done at reset.

If BMS is detected at 0, the controller boots on the memory connected to Chip Select 0 of the External Bus Interface.

In this boot mode, the chip starts with its default parameters (all registers in their reset state), including as follows:

- the main clock is the on-chip 12 MHz RC oscillator
- the Static Memory Controller is configured with its default parameters

The user software in the external memory performs a complete configuration:

- Enable the 32768 Hz oscillator if best accuracy is needed
- Program the PMC (main oscillator enable or bypass mode)
- Program and start the PLL
- Reprogram the SMC setup, cycle, hold, mode timing registers for EBI CS0, to adapt them to the new clock
- Switch the system clock to the new value

If BMS is detected at 1, the boot memory is the embedded ROM and the Boot Program described below is executed (Section 11.2.1 "ROM Code").

11.2.1 ROM Code

The ROM Code is a boot program contained in the embedded ROM. It is also called "First level bootloader".

The ROM Code performs several steps:

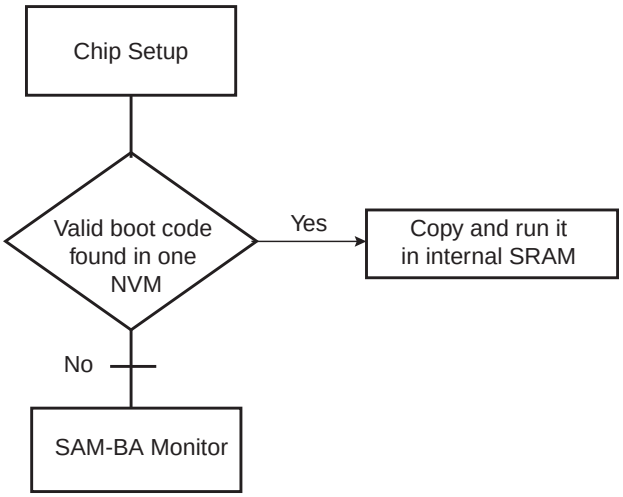
- basic chip initialization: XTAL or external clock frequency detection
- attempt to retrieve a valid code from external non-volatile memories (NVM)
- execution of a monitor called SAM-BA Monitor, in case no valid application has been found on any NVM

© 2017 Microchip Technology Inc.

11.2.2 Flow Diagram

The ROM Code implements the algorithm shown below in Figure 11-1.

Figure 11-1: ROM Code Algorithm Flow Diagram



11.2.3 Chip Setup

At boot start-up, the processor clock (PCK) and the master clock (MCK) source is the 12 MHz Fast RC Oscillator.

Initialization follows the steps described below:

1. **Stack setup** for Arm supervisor mode.
2. **Main Oscillator Detection:** The Main Clock is switched to the 32 kHz RC oscillator to allow external clock frequency to be measured. Then the Main Oscillator is enabled and set in bypass mode. If the MOSCSELS bit rises, an external clock is connected, and the next step is Main Clock Selection (3)**.** If not, the Bypass mode is cleared to attempt external quartz detection. This detection is successful when the MOSCXTS and MOSCSELS bits rise, else the 12 MHz Fast RC internal oscillator is used as the Main Clock.
3. **Main Clock Selection**: The Master Clock source is switched from the Slow Clock to the Main Oscillator without prescaler. The PMC Status Register is polled to wait for MCK Ready. PCK and MCK are now the Main Clock.
4. **C variable initialization:** Non zero-initialized data is initialized in the RAM (copy from ROM to RAM). Zero-initialized data is set to 0 in the RAM.
5. **PLLA initialization**: PLLA is configured to get a PCK at 48 MHz and an MCK at 48 MHz. If an external clock or crystal frequency running at 12 MHz is found, then the PLLA is configured to allow communication on the USB link for the SAM-BA Monitor; else the Main Clock is switched to the internal 12 MHz Fast RC, but USB will not be activated.

Table 11-1: External Clock and Crystal frequencies allowed for Boot Sequence (in MHz)

Boot Sequence	**≤ 4**	**12**	**≥ 28**
Boot on external memories	Yes	Yes	Yes
SAM-BA Monitor through DBGU	Yes	Yes	Yes
SAM-BA Monitor through USB	No	Yes	No

Providing a clock frequency not at 12 MHz but between 4 and 28 MHz will be considered by the ROM Code as 12 MHz, and PLL settings will be configured accordingly.

© 2017 Microchip Technology Inc.

11.2.4 NVM Boot

11.2.4.1 NVM Boot Sequence

The boot sequence on external memory devices can be controlled using the Boot Sequence Controller Configuration Register (BSC_CR). The three LSBs of the BSC_CR are available to control the sequence.

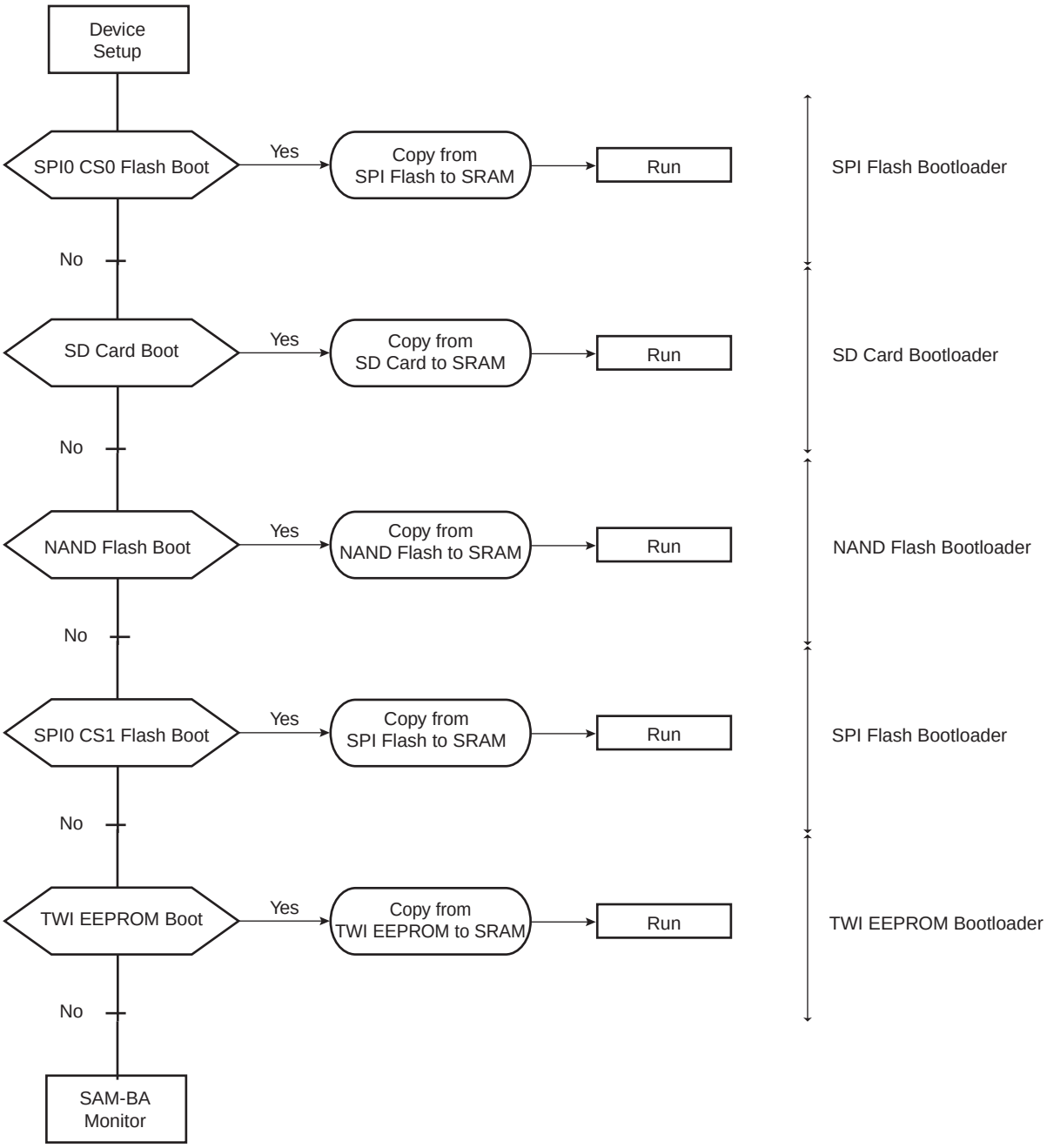
The user can then choose to bypass some steps shown in Figure 11-2 “NVM Bootloader Sequence Diagram” according to the value of the BOOT field in the BSC_CR.

Table 11-2: Boot Sequence Controller Configuration Register Values

BSC_CR.BOOT Value	SPI0 NPCS0	SDCard	NAND Flash	SPI0 NPCS1	TWI EEPROM	SAM-BA Monitor
0	Y	Y	Y	Y	Y	Y
1	Y	–	Y	Y	Y	Y
2	Y	–	–	Y	Y	Y
3	Y	–	–	Y	Y	Y
4	Y	–	–	–	Y	Y
5	–	–	–	–	–	Y
6	–	–	–	–	–	Y
7	–	–	–	–	–	Y

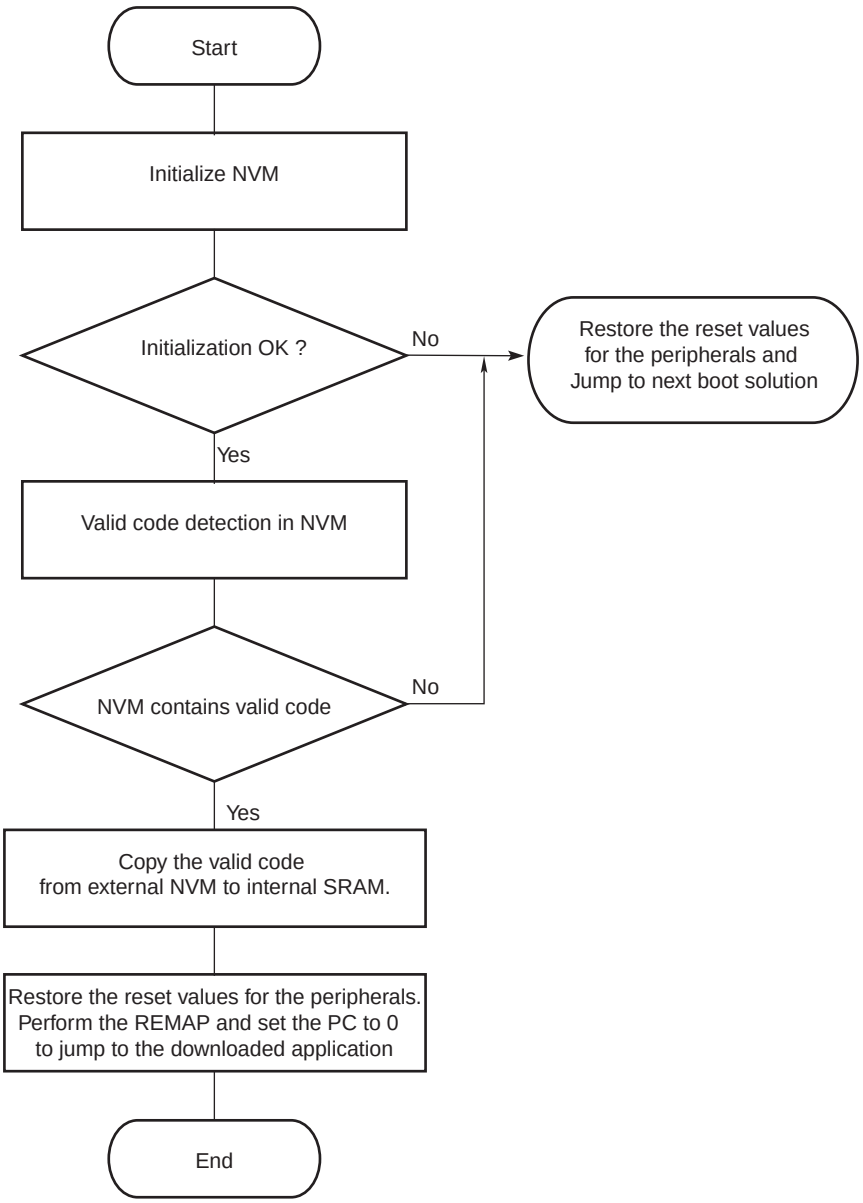
© 2017 Microchip Technology Inc.

Figure 11-2: NVM Bootloader Sequence Diagram



11.2.4.2 NVM Bootloader Program Description

Figure 11-3: NVM Bootloader Program Diagram



The NVM bootloader program first initializes the PIOs related to the NVM device. Then it configures the right peripheral depending on the NVM and tries to access this memory. If the initialization fails, it restores the reset values for the PIO and the peripheral and then tries the same operations on the next NVM of the sequence.

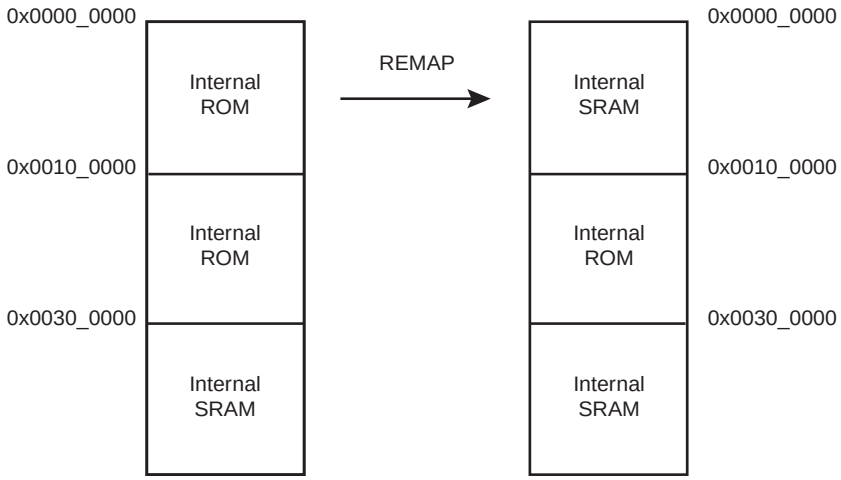
If the initialization is successful, the NVM bootloader program reads the beginning of the NVM and determines if the NVM contains valid code.

If the NVM does not contain valid code, the NVM bootloader program restores the reset value for the peripherals and then tries the same operations on the next NVM of the sequence.

© 2017 Microchip Technology Inc.

If a valid code is found, this code is loaded from NVM into internal SRAM and executed by branching at address 0x0000_0000 after remap. This code may be the application code or a second-level bootloader.

Figure 11-4: Remap Action after Download Completion



11.2.4.3 Valid Code Detection

There are two kinds of valid code detection.

- Arm Exception Vectors Check

The NVM bootloader program reads and analyzes the first 28 bytes corresponding to the first seven Arm exception vectors. Except for the sixth vector, these bytes must implement the Arm instructions for either branch or load PC with PC relative addressing.

Figure 11-5: LDR Opcode

31 28	27 24	23 20	19 16	15 12	11 0
1 1 1 0	0 1 I P	U 1 W 0	Rn	Rd	Offset

Figure 11-6: B Opcode

31 28	27 24	23 0
1 1 1 0	1 0 1 0	Offset (24 bits)

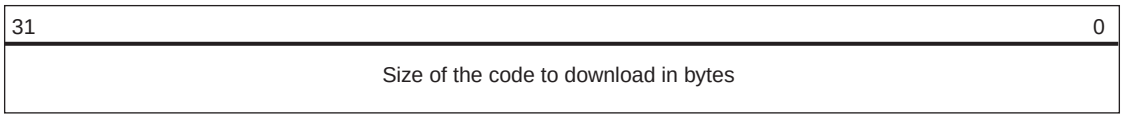
Unconditional instruction: 0xE for bits 31 to 28

Load PC with PC relative addressing instruction:

- Rn = Rd = PC = 0xF
- I==0 (12-bit immediate value)
- P==1 (pre-indexed)
- U offset added (U==1) or subtracted (U==0)
- W==1

The sixth vector, at offset 0x14, contains the size of the image to download. The user must replace this vector with the user's own vector. This information is described below.

Figure 11-7: Structure of the Arm Vector 6



The value has to be smaller than 24 Kbytes. This size is the internal SRAM size minus the stack size used by the ROM Code at the end of the internal SRAM.

Example

An example of valid vectors follows:

```
00          ea000006                B               0x20
04          eafffffe                B               0x04
08          ea00002f                B               _main
0c          eafffffe                B               0x0c
10          eafffffe                B               0x10
14          00001234                B               0x14            <- Code size = 4660 bytes
18          eafffffe                B               0x18
```

- boot.bin File Check

This method is the one used on FAT formatted SDCard. The boot program must be a file named "`boot.bin`" written in the root directory of the file system. Its size must not exceed the maximum size allowed: 24 kbytes (0x6000).

11.2.4.4 Detailed Memory Boot Procedures

- NAND Flash Boot: NAND Flash Detection

After NAND Flash interface configuration, a reset command is sent to the memory.

The Boot Program first tries to find valid software on a NAND Flash device connected to EBI CS3, with data lines connected to D0–D7, then on NAND Flash connected to D16–D23. Hardware ECC detection and correction are provided by the PMECC peripheral (refer to Section 27. "Programmable Multibit Error Correction Code Controller (PMECC)" for more information).

The Boot Program is able to retrieve NAND Flash parameters and ECC requirements using two methods as follows:

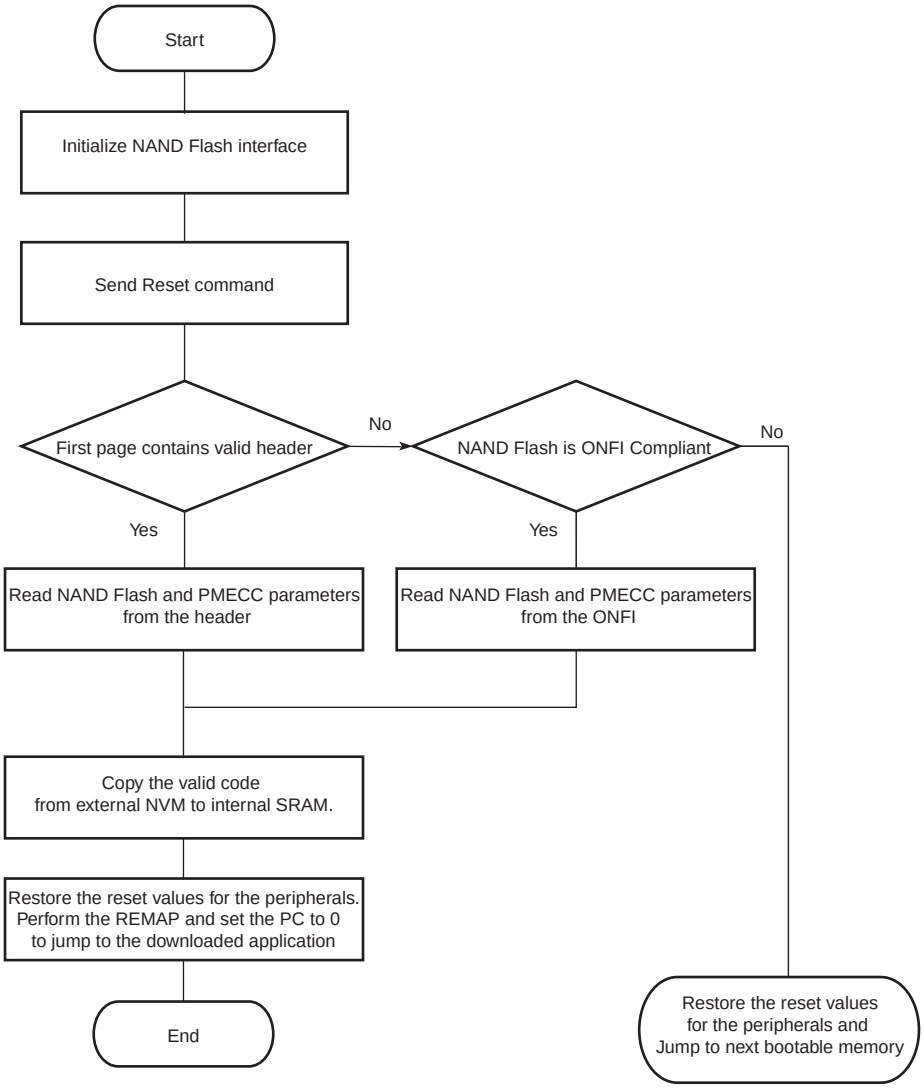
- the detection of a specific header written at the beginning of the first page of NAND Flash,

or

- through the ONFI parameters for ONFI compliant memories.

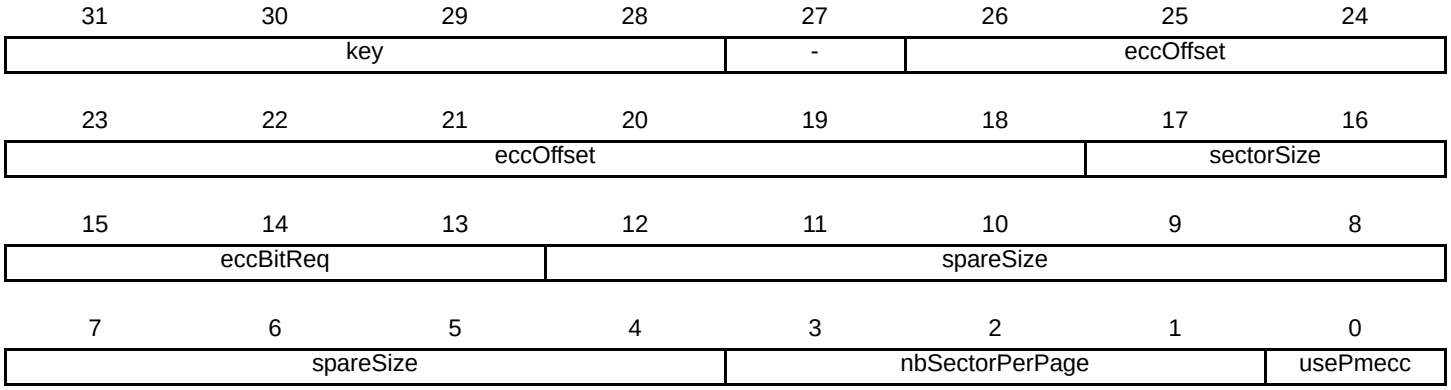
© 2017 Microchip Technology Inc.

Figure 11-8: Boot NAND Flash Download



NAND Flash Specific Header Detection

This is the first method used to determine NAND Flash parameters. After Initialization and Reset command, the Boot Program reads the first page without ECC check, to determine if the NAND parameter header is present. The header is made of 52 times the same 32-bit word (for redundancy reasons) which must contain NAND and PMECC parameters used to correctly perform the read of the rest of the data in the NAND. This 32-bit word is described below:



usePmecc: Use PMECC

0: Do not use PMECC to detect and correct the data.

1: Use PMECC to detect and correct the data.

nbSectorPerPage: Number of sectors per page

spareSize: Size of the spare zone in bytes

eccBitReq: Number of ECC bits required

sectorSize: Size of the ECC sector

0: for 512 bytes

1: for 1024 bytes per sector

Other value for future use.

eccOffset: Offset of the first ECC byte in the spare zone

A value below 2 is not allowed and will be considered as 2.

key: value 0xC must be written here to validate the content of the whole word.

If the header is valid, the Boot Program will continue with the detection of valid code.

© 2017 Microchip Technology Inc.

ONFI 2.2 Parameters

In case no valid header has been found, the Boot Program will check if the NAND Flash is ONFI compliant, sending a Read Id command (0x90) with 0x20 as parameter for the address. If the NAND Flash is ONFI compliant, the Boot Program retrieves the following parameters with the help of the Get Parameter Page command:

- Number of bytes per page (byte 80)
- Number of bytes in spare zone (byte 84)
- Number of ECC bit correction required (byte 112)
- ECC sector size: by default set to 512 bytes, or 1024 bytes if the ECC bit capability above is 0xFF

By default, ONFI NAND Flash detection will turn ON the usePmecc parameter, and ECC correction algorithm is automatically activated.

Once the Boot Program retrieves the parameter, using one of the two methods described above, it will read the first page again, with or without ECC, depending on the usePmecc parameter. Then it looks for a valid code programmed just after the header offset 0xD0. If the code is valid, the program is copied at the beginning of the internal SRAM.

Note: Booting on 16-bit NAND Flash is not possible, only 8-bit NAND Flash memories are supported.

- NAND Flash Boot: PMECC Error Detection and Correction

NAND Flash boot procedure uses PMECC to detect and correct errors during NAND Flash read operations in two cases:

- when the usePmecc flag is set in the specific NAND header. If the flag is not set, no ECC correction is performed during NAND Flash page read.
- when the NAND Flash has been detected using ONFI parameters.

The ROM code embeds the software used in the process of ECC detection/correction: the Galois Field tables, and the function PMECC_CorrectionAlgo(). The user does not need to embed it in other software.

This function can be called by user software when PMECC status returns errors after a read page command.

Its address can be retrieved by reading the third vector of the ROM Code interrupt vector table, at address 0x100008.

The API of this function is:

```
unsigned int PMECC_CorrectionAlgo(AT91PS_PMECC pPMECC,
                                  AT91PS_PMERRLOC pPMERRLOC,
                                  PMECC_paramDesc_struct *PMECC_desc,
                                  unsigned int PMECC_status,
                                  unsigned int pageBuffer)
```

pPMECC: pointer to the PMECC base address,

pPMERRLOC: pointer to the PMERRLOC base address,

PMECC_desc: pointer to the PMECC descriptor,

PMECC_status: the status returned by the read of PMECCISR register;

pageBuffer: address of the buffer containing the page to be corrected.

The PMECC descriptor structure is:

```
typedef struct _PMECC_paramDesc_struct {
    unsigned int pageSize;
    unsigned int spareSize;
    unsigned int sectorSize;  // 0 for 512, 1 for 1024 bytes
    unsigned int errBitNbrCapability;
    unsigned int eccSizeByte;
    unsigned int eccStartAddr;
    unsigned int eccEndAddr;

    unsigned int nandWR;
    unsigned int spareEna;
    unsigned int modeAuto;
    unsigned int clkCtrl;
    unsigned int interrupt;

    int tt;
    int mm;
    int nn;

    short *alpha_to;
    short *index_of;

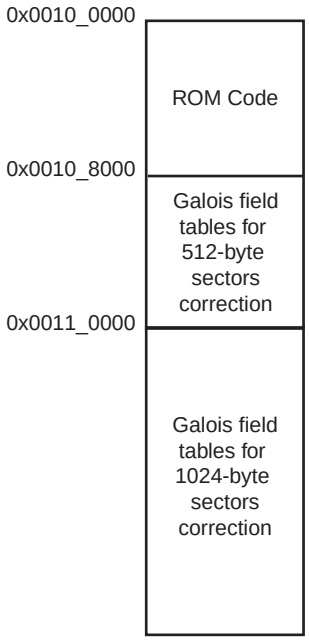
    short partialSyn[100];
    short si[100];

    /* sigma table */
    short smu[TT_MAX + 2][2 * TT_MAX + 1];
    /* polynom order */
    short lmu[TT_MAX + 1];

} PMECC_paramDesc_struct;
```

The Galois field tables are mapped in the ROM just after the ROM code, as described in Figure 11-9.

Figure 11-9: Galois Field Table Mapping



For a full description and an example of how to use the PMECC detection and correction feature, refer to the software package dedicated to this device on www.microchip.com.

© 2017 Microchip Technology Inc.

- SD Card Boot

The SD Card bootloader uses MCI0. It looks for a “`boot.bin`” file in the root directory of a FAT12/16/32 formatted SD Card.

Supported SD Card Devices

SD Card Boot supports all SD Card memories compliant with SD Memory Card Specification V2.0. This includes SDHC cards.

- SPI Flash Boot

Two kinds of SPI Flash are supported: SPI Serial Flash and SPI DataFlash.

The SPI Flash bootloader tries to boot on SPI0 Chip Select 0, first looking for SPI Serial Flash, and then for SPI DataFlash.

It uses only one valid code detection: analysis of Arm exception vectors.

The SPI Flash read is done by means of a Continuous Read command from address 0x0. This command is 0xE8 for DataFlash and 0x0B for Serial Flash devices.

Supported DataFlash Devices

The SPI Flash Boot program supports the DataFlash devices listed in Table 11-3.

Table 11-3: DataFlash Devices

Device	Density	Page Size (bytes)	Number of Pages
AT45DB011	1 Mbit	264	512
AT45DB021	2 Mbits	264	1024
AT45DB041	4 Mbits	264	2048
AT45DB081	8 Mbits	264	4096
AT45DB161	16 Mbits	528	4096
AT45DB321	32 Mbits	528	8192
AT45DB642	64 Mbits	1056	8192

Supported Serial Flash Devices

The SPI Flash Boot program supports all SPI Serial Flash devices responding correctly at both Get Status and Continuous Read commands.

- TWI EEPROM Boot

The TWI EEPROM Bootloader uses the TWI0. It uses only one valid code detection. It analyzes the Arm exception vectors.

Supported TWI EEPROM Devices

TWI EEPROM Boot supports all I^2C-compatible TWI EEPROM memories using 7-bit device address 0x50.

11.2.4.5 Hardware and Software Constraints

The NVM drivers use several PIOs in peripheral mode to communicate with external memory devices. Care must be taken when these PIOs are used by the application. The devices connected could be unintentionally driven at boot time, and electrical conflicts between output pins used by the NVM drivers and the connected devices may occur.

To assure correct functionality, it is recommended to plug in critical devices to other pins not used by NVM.

Table 11-4 contains a list of pins that are driven during the boot program execution. These pins are driven during the boot sequence for a period of less than 1 second if no correct boot program is found.

Before performing the jump to the application in internal SRAM, all the PIOs and peripherals used in the boot program are set to their reset state.

Table 11-4: PIO Driven during Boot Program Execution

NVM Bootloader	Peripheral	Pin	PIO Line
NAND	EBI CS3 SMC	NANDOE	PIOD0
	EBI CS3 SMC	NANDWE	PIOD1
	EBI CS3 SMC	NANDCS	PIOD4
	EBI CS3 SMC	NAND ALE	A21
	EBI CS3 SMC	NAND CLE	A22
	EBI CS3 SMC	Cmd/Addr/Data	D[16:0]
SD Card	MCI0	MCI0_CK	PIOA17
	MCI0	MCI0_D0	PIOA15
	MCI0	MCI0_D1	PIOA18
	MCI0	MCI0_D2	PIOA19
	MCI0	MCI0_D3	PIOA20
SPI Flash	SPI0	MOSI	PIOA10
	SPI0	MISO	PIOA11
	SPI0	SPCK	PIOA13
	SPI0	NPCS0	PIOA14
	SPI0	NPCS1	PIOA7
TWI0 EEPROM	TWI0	TWD0	PIOA30
	TWI0	TWCK0	PIOA31
SAM-BA Monitor	DBGU	DRXD	PIOA9
	DBGU	DTXD	PIOA10

11.2.5 SAM-BA Monitor

If no valid code has been found in NVM during the NVM bootloader sequence, the SAM-BA Monitor program is launched.

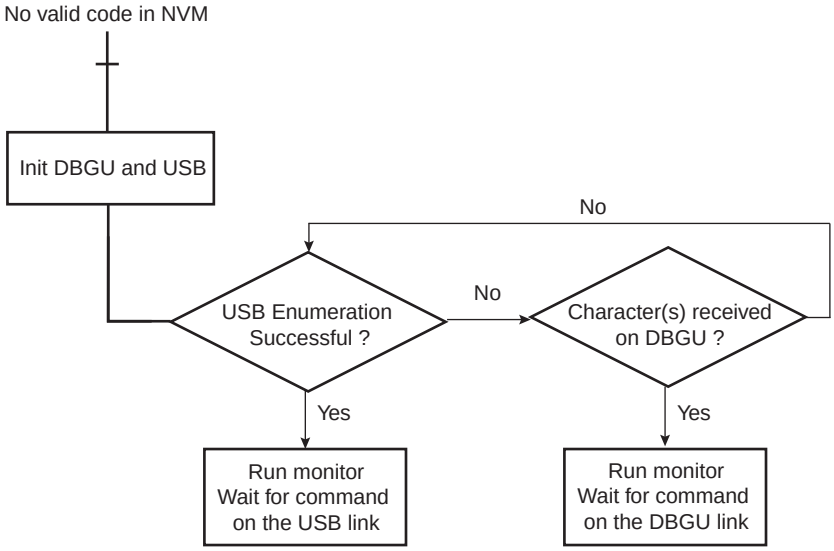
The SAM-BA Monitor principle is to:

- Initialize DBGU and USB
- Check if USB Device enumeration has occurred
- Check if characters have been received on the DBGU

Once the communication interface is identified, the application runs in an infinite loop waiting for different commands as listed in Table 11-5.

© 2017 Microchip Technology Inc.

Figure 11-10: SAM-BA Monitor Diagram



11.2.5.1 Command List

Table 11-5: Commands Available through the SAM-BA Monitor

Command	Action	Argument(s)	Example
N	set Normal mode	No argument	**N**#
T	set Terminal mode	No argument	**T**#
O	write a byte	Address, Value#	**O**200001,CA#
o	read a byte	Address,#	**o**200001,#
H	write a half word	Address, Value#	**H**200002,CAFE#
h	read a half word	Address,#	**h**200002,#
W	write a word	Address, Value#	**W**200000,CAFEDECA#
w	read a word	Address,#	**w**200000,#
S	send a file	Address,#	**S**200000,#
R	receive a file	Address, NbOfBytes#	**R**200000,1234#
G	go	Address#	**G**200200#
V	display version	No argument	**V**#

- Mode commands:
  - Normal mode configures SAM-BA Monitor to send / receive data in binary format,
  - Terminal mode configures SAM-BA Monitor to send / receive data in ascii format.
- Write commands: Write a byte (**O**), a halfword (**H**) or a word (**W**) to the target.
  - *Address*: Address in hexadecimal.
  - *Value*: Byte, halfword or word to write in hexadecimal.
  - *Output*: '>'
- Read commands: Read a byte (**o**), a halfword (**h**) or a word (**w**) from the target.
  - *Address*: Address in hexadecimal.
  - *Output*: The byte, halfword or word read in hexadecimal followed by '>'
- Send a file (**S**): Send a file to a specified address.
  - *Address*: Address in hexadecimal.
  - *Output*: '>'

Note: There is a time-out on this command which is reached when the prompt '>' appears before the end of the command execution.

- Receive a file (**R**): Receive data into a file from a specified address
  - *Address*: Address in hexadecimal.
  - *NbOfBytes*: Number of bytes in hexadecimal to receive.
  - *Output*: '>'
- Go (**G**): Jump to a specified address and execute the code.
  - *Address*: Address to jump in hexadecimal.
  - *Output*: '>'once returned from the program execution. If the executed program does not handle the link register at its entry and does not return, the prompt will not be displayed.
- Get Version (**V**): Return the Boot Program version.
  - *Output*: version, date and time of ROM code followed by '>'.

11.2.5.2 DBGU Serial Port

Communication is performed through the DBGU serial port initialized to 115,200 Baud, 8 bits of data, no parity, 1 stop bit.

- Supported External Crystal/External Clocks

The SAM-BA Monitor supports a frequency of 12 MHz to allow DBGU communication for both external crystal and external clock.

- Xmodem Protocol

The Send and Receive File commands use the Xmodem protocol to communicate. Any terminal performing this protocol can be used to send the application file to the target. The size of the binary file to send depends on the SRAM size embedded in the product. In all cases, the size of the binary file must be lower than the SRAM size because the Xmodem protocol requires some SRAM memory in order to work.

The Xmodem protocol supported is the 128-byte length block. This protocol uses a two-character CRC16 to guarantee detection of a maximum bit error.

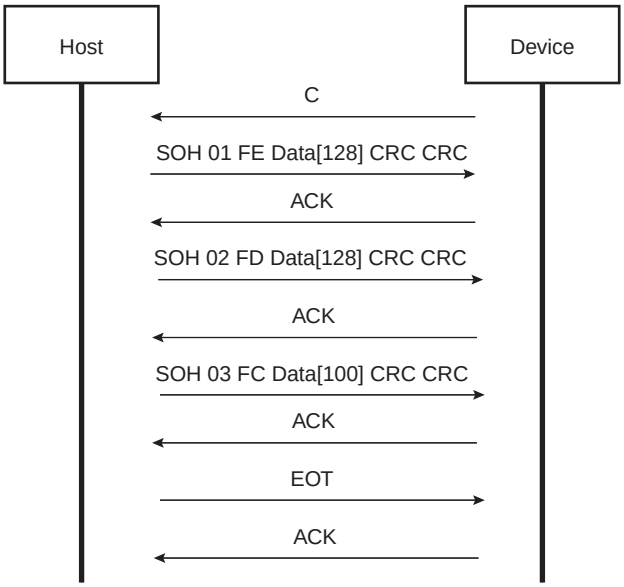
Xmodem protocol with CRC is accurate provided both sender and receiver report successful transmission. Each block of the transfer looks like:

<SOH><blk #><255-blk #><--128 data bytes--><checksum> in which:

- <SOH> = 01 hex
- <blk #> = binary number, starts at 01, increments by 1, and wraps 0FFH to 00H (not to 01)
- <255-blk #> = 1's complement of the blk#.
- <checksum> = 2 bytes CRC16

Figure 11-11 shows a transmission using this protocol.

Figure 11-11: Xmodem Transfer Example



© 2017 Microchip Technology Inc.

11.2.5.3 USB Device Port

- Supported External Crystal / External Clocks

The frequencies supported by SAM-BA Monitor to allow USB communication are 4, 8, 12 or 16 MHz crystal or external clock.

- USB Class

The device uses the USB Communication Device Class (CDC) drivers to take advantage of the installed PC RS-232 software to talk over the USB. The CDC class is implemented in all releases of Windows®, beginning with Windows 98SE. The CDC document, available at www.usb.org, describes how to implement devices such as ISDN modems and virtual COM ports.

The Vendor ID is Microchip's vendor ID 0x03EB. The product ID is 0x6124. These references are used by the host operating system to mount the correct driver. On Windows systems, the INF files contain the correspondence between vendor ID and product ID.

- Enumeration Process

The USB protocol is a master/slave protocol. The host starts the enumeration, sending requests to the device through the control endpoint. The device handles standard requests as defined in the USB Specification.

Table 11-6: Handled Standard Requests

Request	Definition
GET_DESCRIPTOR	Returns the current device configuration value.
SET_ADDRESS	Sets the device address for all future device access.
SET_CONFIGURATION	Sets the device configuration.
GET_CONFIGURATION	Returns the current device configuration value.
GET_STATUS	Returns status for the specified recipient.
SET_FEATURE	Used to set or enable a specific feature.
CLEAR_FEATURE	Used to clear or disable a specific feature.

The device also handles some class requests defined in the CDC class.

Table 11-7: Handled Class Requests

Request	Definition
SET_LINE_CODING	Configures DTE rate, stop bits, parity and number of character bits.
GET_LINE_CODING	Requests current DTE rate, stop bits, parity and number of character bits.
SET_CONTROL_LINE_STATE	RS-232 signal used to tell the DTE device is now present.

Unhandled requests are stalled.

- Communication Endpoints

There are two communication endpoints and endpoint 0 is used for the enumeration process. Endpoint 1 is a 64-byte Bulk OUT endpoint and endpoint 2 is a 64-byte Bulk IN endpoint. SAM-BA Boot commands are sent by the host through endpoint 1. If required, the message is split by the host into several data payloads by the host driver.

If the command requires a response, the host can send IN transactions to pick up the response.

© 2017 Microchip Technology Inc.

12. Boot Sequence Controller (BSC)

12.1 Description

The System Controller embeds a Boot Sequence Controller (BSC). The boot sequence is programmable through the Boot Sequence Controller Configuration Register (BSC_CR) to save timeout delays on boot.

The BSC_CR is powered by VDDBU. Any modification of the register value is stored and applied after the next reset. The register defaults to the factory value in case of battery removal.

The BSC_CR is programmable with user programs or SAM-BA and is key-protected.

12.2 Embedded Characteristics

- VDDBU powered register

12.3 Product Dependencies

- Product-dependent order

© 2017 Microchip Technology Inc.

12.4 Boot Sequence Controller (BSC) Registers User Interface

Table 12-1: Register Mapping

Offset	Register	Name	Access	Reset
0x0	Boot Sequence Controller Configuration Register	BSC_CR	Read/Write	–

12.4.1 Boot Sequence Controller Configuration Register

Name:BSC_CR

Address:0xFFFFFE54

Access: Read/Write

Factory Value: 0x0000_0000

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
BOOT							

BOOT: Boot Media Sequence

This value is defined in Section 11. "Boot Strategies". It is only written if WPKEY carries the valid value.

WPKEY: Write Protection Key (Write-only)

Value	Name	Description
0x6683	PASSWD	Writing any other value in this field aborts the write operation of the BOOT field. Always reads as 0.

© 2017 Microchip Technology Inc.

13. Reset Controller (RSTC)

13.1 Description

The Reset Controller (RSTC), based on power-on reset cells, handles all the resets of the system without any external components. It reports which reset occurred last.

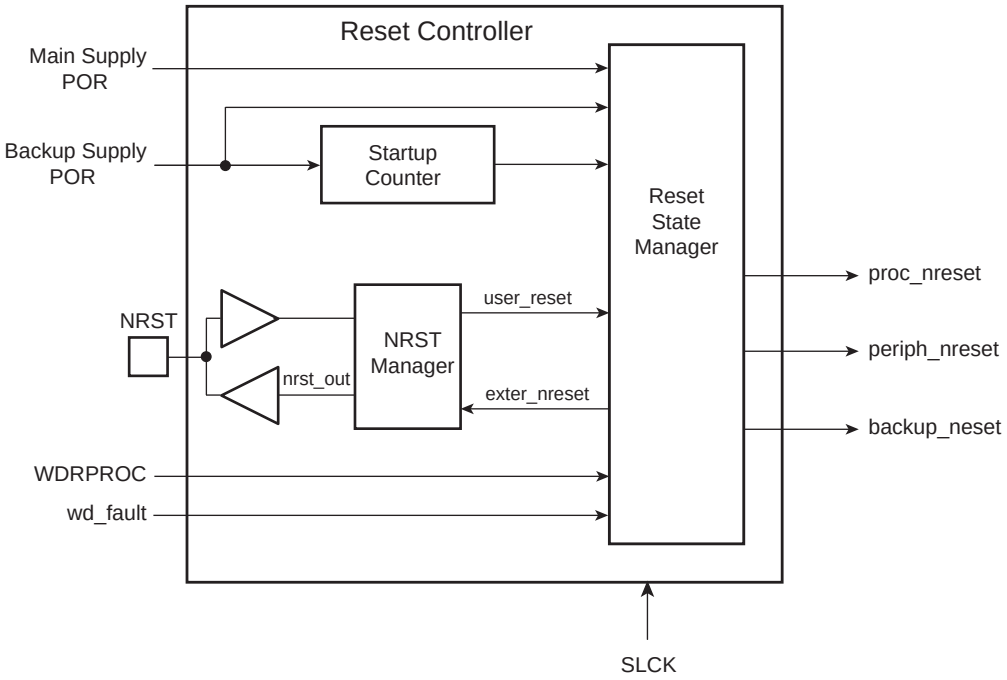
The Reset Controller also drives independently or simultaneously the external reset and the peripheral and processor resets.

13.2 Embedded Characteristics

- Manages All Resets of the System, Including
  - External Devices Through the NRST Pin
  - Processor Reset
  - Peripheral Set Reset
  - Backed-up Peripheral Reset
- Based on 2 Embedded Power-on Reset Cells
- Reset Source Status
  - Status of the Last Reset
  - Either General Reset, Wake-up Reset, Software Reset, User Reset, Watchdog Reset
- External Reset Signal Shaping

13.3 Block Diagram

Figure 13-1: Reset Controller Block Diagram



13.4 Functional Description

13.4.1 Reset Controller Overview

The Reset Controller is made up of an NRST Manager, a Startup Counter and a Reset State Manager. It runs at Slow Clock and generates the following reset signals:

- proc_nreset: Processor reset line. It also resets the Watchdog Timer.
- backup_nreset: Affects all the peripherals powered by VDDBU.
- periph_nreset: Affects the whole set of embedded peripherals.

- nrst_out: Drives the NRST pin.

These reset signals are asserted by the Reset Controller, either on external events or on software action. The Reset State Manager controls the generation of reset signals and provides a signal to the NRST Manager when an assertion of the NRST pin is required.

The NRST Manager shapes the NRST assertion during a programmable time, thus controlling external device resets.

The startup counter waits for the complete crystal oscillator startup. The wait delay is given by the crystal oscillator startup time maximum value that can be found in Section 47.4.4, Crystal Oscillator Characteristics.

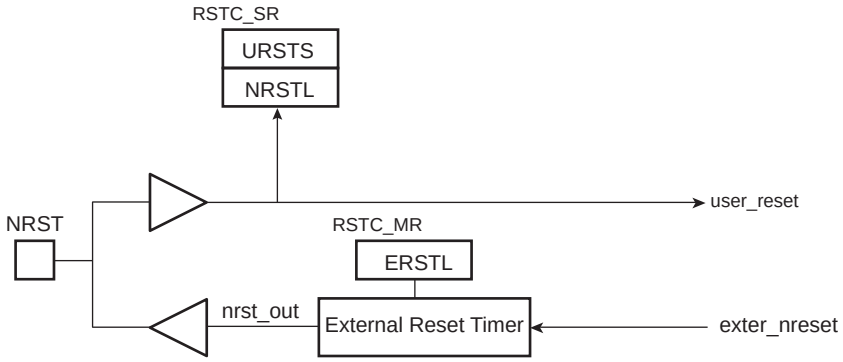
The Reset Controller Mode Register (RSTC_MR), used to configure the reset controller, is powered with VDDBU, so that its configuration is saved as long as VDDBU is on.

13.4.2 NRST Manager

After power-up, NRST is an output during the External Reset Length (ERSTL) time defined in the RSTC. When the ERSTL time has elapsed, the pin behaves as an input and all the system is held in reset if NRST is tied to GND by an external signal.

The NRST Manager samples the NRST input pin and drives this pin low when required by the Reset State Manager. Figure 13-2 shows the block diagram of the NRST Manager.

Figure 13-2: NRST Manager



13.4.2.1 NRST Signal

The NRST Manager handles the NRST input line asynchronously. When the line is low, a User Reset is immediately reported to the Reset State Manager. When the NRST goes from low to high, the internal reset is synchronized with the Slow Clock to provide a safe internal de-assertion of reset.

The level of the pin NRST can be read at any time in the bit NRSTL (NRST level) in the Reset Controller Status Register (RSTC_SR). As soon as the pin NRST is asserted, the bit URSTS in the RSTC_SR is set. This bit clears only when RSTC_SR is read.

13.4.2.2 NRST External Reset Control

The Reset State Manager asserts the signal ext_nreset to assert the NRST pin. When this occurs, the “nrst_out” signal is driven low by the NRST Manager for a time programmed by the field ERSTL in the RSTC_MR. This assertion duration, named EXTERNAL_RESET_LENGTH, lasts $2^{(ERSTL+1)}$ Slow Clock cycles. This gives the approximate duration of an assertion between 60 µs and 2 seconds. Note that ERSTL at 0 defines a two-cycle duration for the NRST pulse.

This feature allows the reset controller to shape the NRST pin level, and thus to guarantee that the NRST line is driven low for a time compliant with potential external devices connected on the system reset.

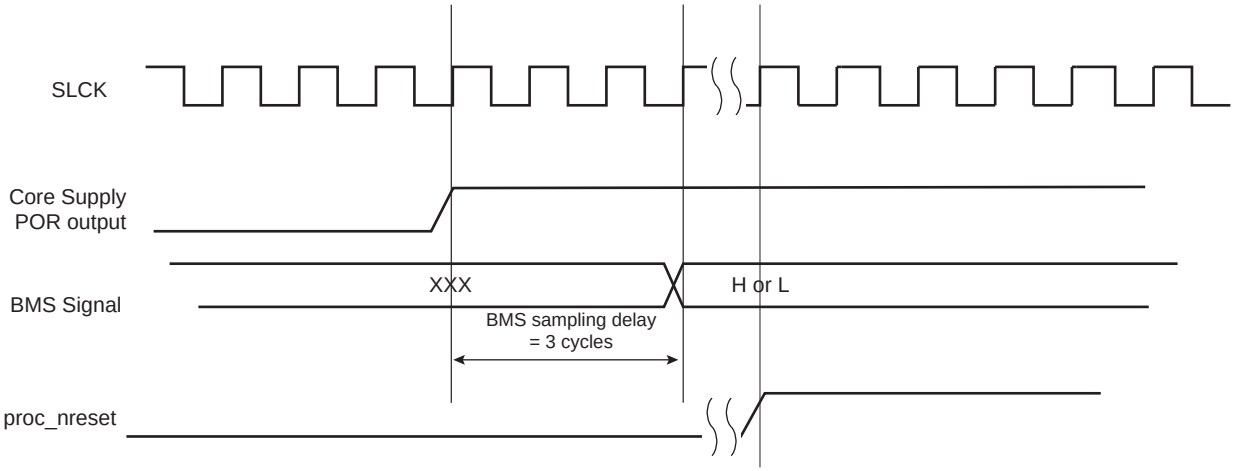
As the field is within RSTC_MR, which is backed-up, this field can be used to shape the system power-up reset for devices requiring a longer startup time than the Slow Clock Oscillator.

13.4.3 BMS Sampling

The product matrix manages a boot memory that depends on the level on the BMS pin at reset. The BMS signal is sampled three slow clock cycles after the Core Power-On-Reset output rising edge.

© 2017 Microchip Technology Inc.

Figure 13-3: BMS Sampling



13.4.4 Reset States

The Reset State Manager handles the different reset sources and generates the internal reset signals. It reports the reset status in the field RSTTYP of the RSTC_SR. The update of the field RSTTYP is performed when the processor reset is released.

13.4.4.1 General Reset

A general reset occurs when VDDBU and VDDCORE are powered on. The backup supply POR cell output rises and is filtered with a Startup Counter, which operates at Slow Clock. The purpose of this counter is to make sure the Slow Clock oscillator is stable before starting up the device. The length of startup time is hardcoded to comply with the Slow Clock Oscillator startup time.

After this time, the processor clock is released at Slow Clock and all the other signals remain valid for 3 cycles for proper processor and logic reset. Then, all the reset signals are released and the field RSTTYP in the RSTC_SR reports a General Reset. As the RSTC_MR is reset, the NRST line rises two cycles after the backup_nreset, as ERSTL defaults at value 0x0.

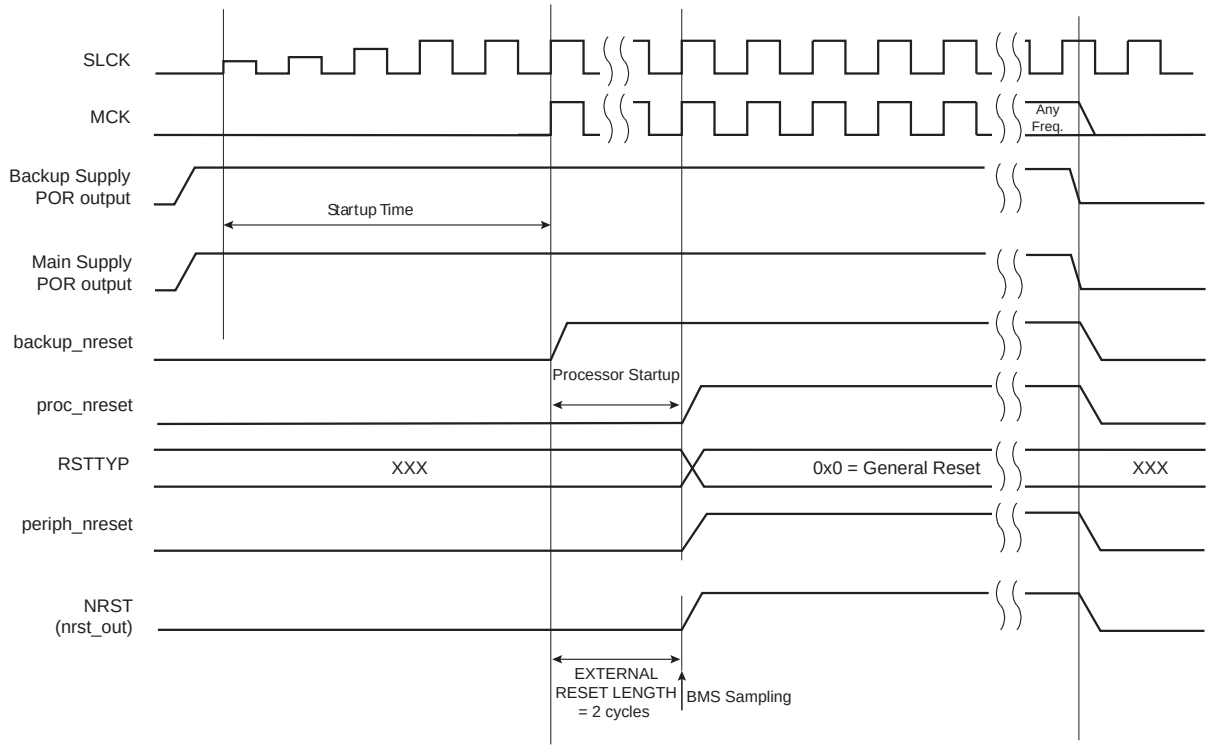
When VDDBU is detected low by the backup supply POR cell, all resets signals are immediately asserted, even if the main supply POR cell does not report a main supply shutdown.

VDDBU only activates the backup_nreset signal.

The backup_nreset must be released so that any other reset can be generated by VDDCORE (main supply POR output).

Figure 13-4 shows how the General Reset affects the reset signals.

Figure 13-4: General Reset State



13.4.4.2 Wake-up Reset

The wake-up reset occurs when the main supply is down. When the main supply POR output is active, all the reset signals are asserted except backup_nreset. When the main supply powers up, the POR output is resynchronized on Slow Clock. The processor clock is then re-enabled during 3 Slow Clock cycles, depending on the requirements of the Arm processor.

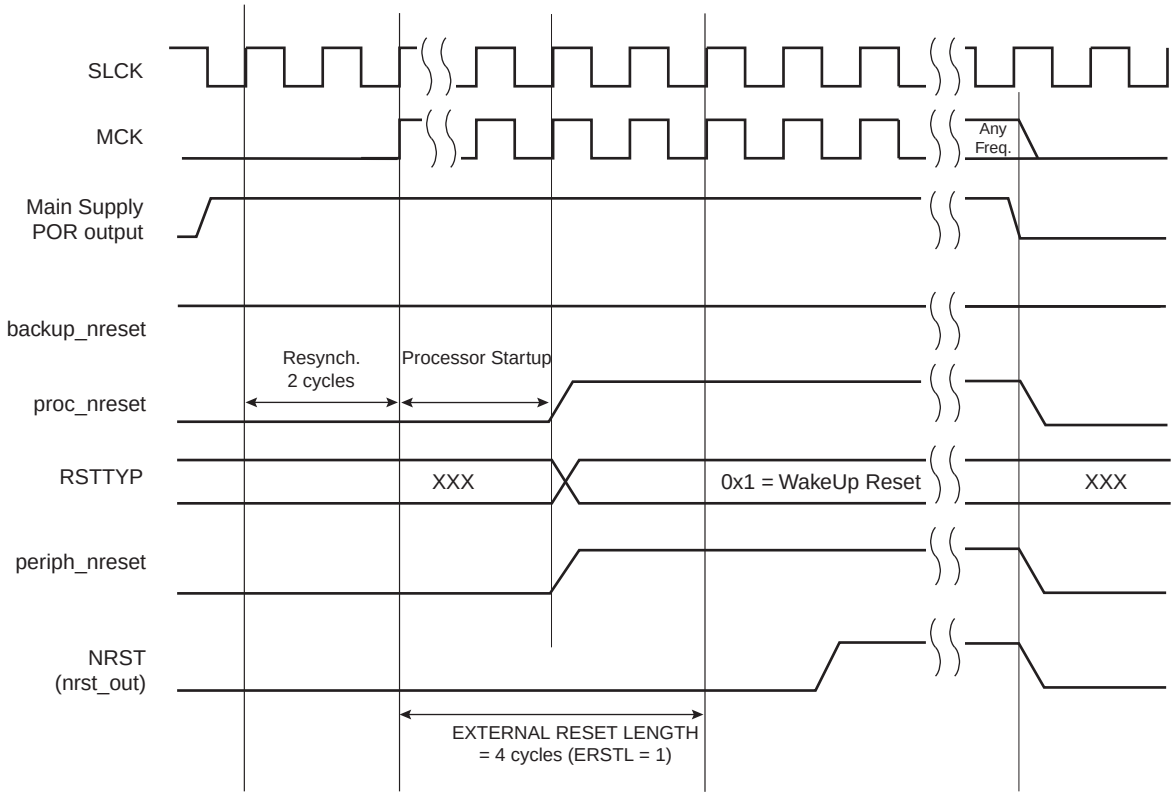
At the end of this delay, the processor and other reset signals rise. The field RSTTYP in the RSTC_SR is updated to report a wake-up reset.

The “nrst_out” remains asserted for EXTERNAL_RESET_LENGTH cycles. As RSTC_MR is backed-up, the programmed number of cycles is applicable.

When the main supply is detected falling, the reset signals are immediately asserted. This transition is synchronous with the output of the main supply POR.

© 2017 Microchip Technology Inc.

Figure 13-5: Wake-up Reset



13.4.4.3 User Reset

The User Reset is entered when a low level is detected on the NRST pin. When a falling edge occurs on NRST (reset activation), internal reset lines are immediately asserted.

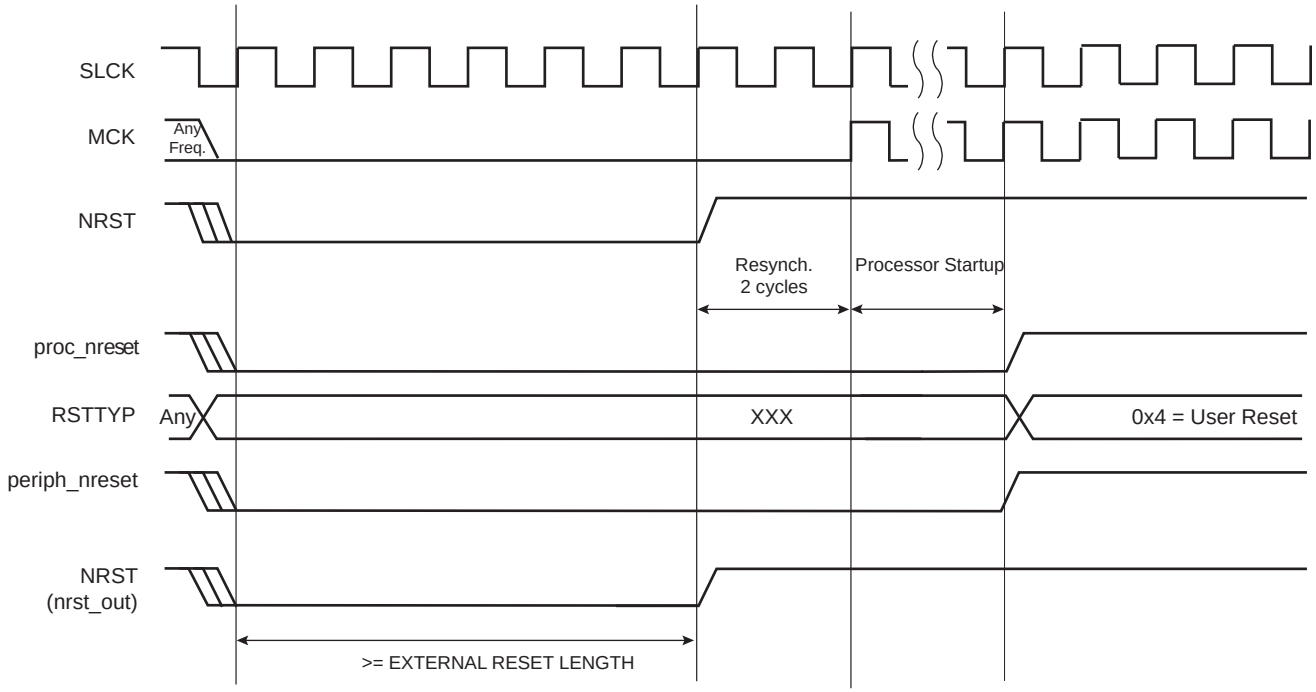
The Processor Reset and the Peripheral Reset are asserted.

The User Reset is left when NRST rises, after a two-cycle resynchronization time and a 3-cycle processor startup. The processor clock is re-enabled as soon as NRST is confirmed high.

When the processor reset signal is released, the RSTTYP field of the RSTC_SR is loaded with the value 0x4, indicating a User Reset.

The NRST Manager guarantees that the NRST line is asserted for EXTERNAL_RESET_LENGTH Slow Clock cycles, as programmed in the field ERSTL. However, if NRST does not rise after EXTERNAL_RESET_LENGTH because it is driven low externally, the internal reset lines remain asserted until NRST actually rises.

Figure 13-6: User Reset State



13.4.4.4 Software Reset

The Reset Controller offers several commands used to assert the different reset signals. These commands are performed by writing the Control Register (RSTC_CR) with the following bits at 1:

- PROCRST: Writing a 1 to PROCRST resets the processor and the watchdog timer.
- PERRST: Writing a 1 to PERRST resets all the embedded peripherals, including the memory system, and, in particular, the Remap Command. The Peripheral Reset is generally used for debug purposes.
  PERRST must always be used in conjunction with PROCRST (PERRST and PROCRST bot set to 1 simultaneously.)
- EXTRST: Writing a 1 to EXTRST asserts low the NRST pin during a time defined by the field ERSTL in the Mode Register (RSTC_MR).

The software reset is entered if at least one of these bits is set by the software. All these commands can be performed independently or simultaneously. The software reset lasts 3 Slow Clock cycles.

The internal reset signals are asserted as soon as the register write is performed. This is detected on the Master Clock (MCK). They are released when the software reset is left, i.e., synchronously to SLCK.

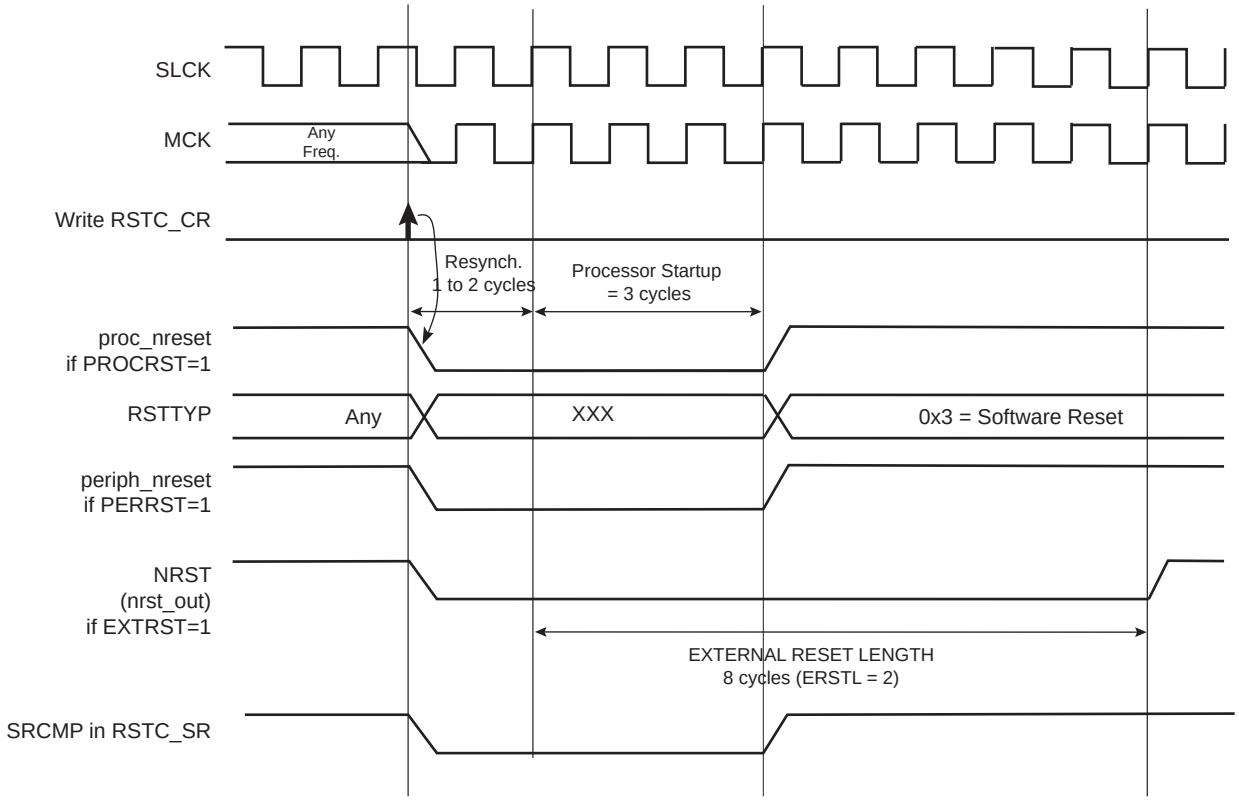
If EXTRST is set, the nrst_out signal is asserted depending on the programming of the field ERSTL. However, the resulting falling edge on NRST does not lead to a User Reset.

If and only if the PROCRST bit is set, the reset controller reports the software status in the field RSTTYP of the RSTC_SR. Other software resets are not reported in RSTTYP.

As soon as a software operation is detected, the bit SRCMP (Software Reset Command in Progress) is set in the RSTC_SR. It is cleared as soon as the software reset is left. No other software reset can be performed while the SRCMP bit is set, and writing any value in the RSTC_CR has no effect.

© 2017 Microchip Technology Inc.

Figure 13-7: Software Reset



13.4.4.5 Watchdog Reset

The Watchdog Reset is entered when a watchdog fault occurs. This state lasts 3 Slow Clock cycles.

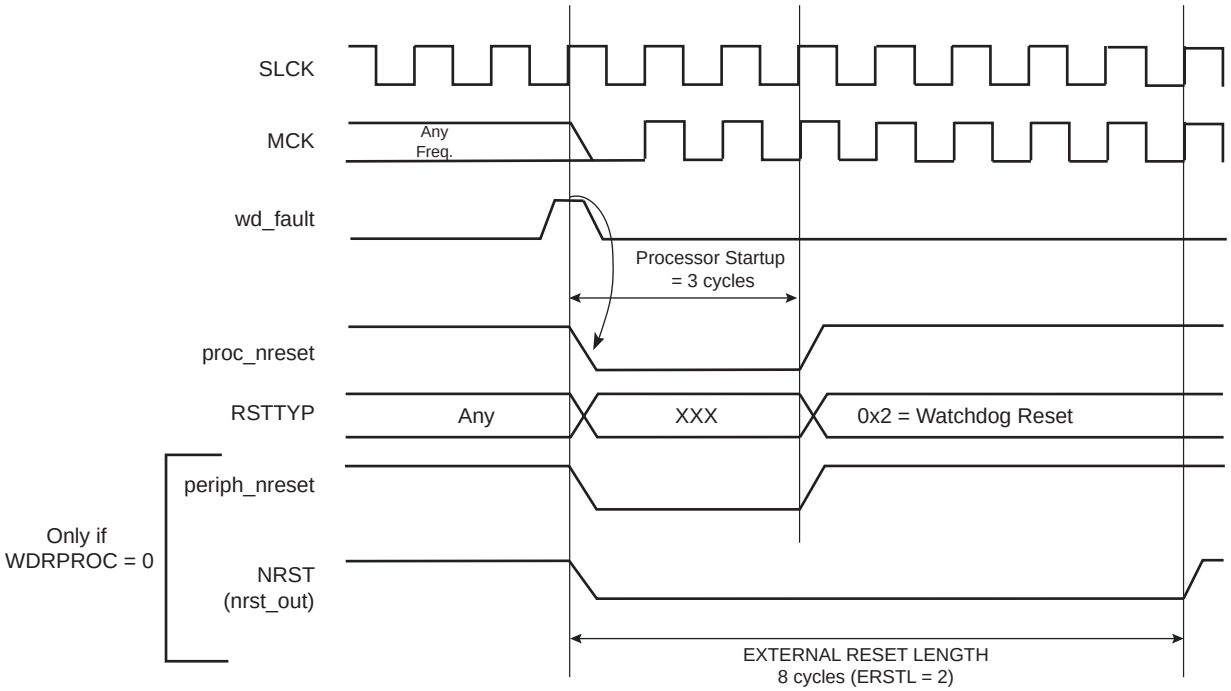
When in Watchdog Reset, assertion of the reset signals depends on the WDRPROC bit in WDT_MR:

- If WDRPROC = 0, the Processor Reset and the Peripheral Reset are asserted. The NRST line is also asserted, depending on how field RSTC_MR.ERSTL is programmed. However, the resulting low level on NRST does not result in a User Reset state.
- If WDRPROC = 1, only the processor reset is asserted.

The Watchdog Timer is reset by the proc_nreset signal. As the watchdog fault always causes a processor reset if WDRSTEN in the WDT_MR is set, the Watchdog Timer is always reset after a Watchdog Reset and the Watchdog is enabled by default and with a period set to a maximum.

When bit WDT_MR.WDRSTEN is reset, the watchdog fault has no impact on the reset controller.

Figure 13-8: Watchdog Reset



13.4.5 Reset State Priorities

The Reset State Manager manages the following priorities between the different reset sources, given in descending order:

- Backup Reset
- Wake-up Reset
- User Reset
- Watchdog Reset
- Software Reset

Particular cases are listed below:

- When in User Reset:
  - A watchdog event is impossible because the Watchdog Timer is being reset by the proc_nreset signal.
  - A software reset is impossible, since the processor reset is being activated.
- When in Software Reset:
  - A watchdog event has priority over the current state.
  - The NRST has no effect.
- When in Watchdog Reset:
  - The processor reset is active and so a Software Reset cannot be programmed.
  - A User Reset cannot be entered.

© 2017 Microchip Technology Inc.

13.5 Reset Controller (RSTC) User Interface

Table 13-1: Register Mapping

Offset	Register	Name	Access	Reset	Back-up Reset
0x00	Control Register	RSTC_CR	Write-only	–	–
0x04	Status Register	RSTC_SR	Read-only	0x0000_0100 (1)	0x0000_0000 (2)
0x08	Mode Register	RSTC_MR	Read/Write	–	0x0000_0000

Note 1: Only power supply VDDCORE rising

2: Both power supplies VDDCORE and VDDBU rising

13.5.1 Reset Controller Control Register

Name:RSTC_CR

Address:0xFFFFFE00

Access Type: Write-only

31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–		–
7	6	5	4	3	2	1	0
–	–	–	–	EXTRST	PERRST	–	PROCRST

PROCRST: Processor Reset

0: No effect

1: If KEY value = 0xA5, resets the processor

PERRST: Peripheral Reset

0: No effect

1: If KEY value = 0xA5, resets the peripherals

EXTRST: External Reset

0: No effect

1: If KEY value = 0xA5, asserts the NRST pin and resets the processor and the peripherals

KEY: Write Access Password

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation. Always reads as 0.

© 2017 Microchip Technology Inc.

13.5.2 Reset Controller Status Register

Name:RSTC_SR

Address:0xFFFFFE04

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	SRCMP	NRSTL
15	14	13	12	11	10	9	8
–	–	–	–	–	RSTTYP		
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	URSTS

URSTS: User Reset Status

A high-to-low transition of the NRST pin sets the URSTS bit. This transition is also detected on the Master Clock (MCK) rising edge. Reading the RSTC_SR resets the URSTS bit.

0: No high-to-low edge on NRST happened since the last read of RSTC_SR.

1: At least one high-to-low transition of NRST has been detected since the last read of RSTC_SR.

RSTTYP: Reset Type

This field reports the cause of the last processor reset. Reading this RSTC_SR does not reset this field.

Value	Name	Description
0	GENERAL_RST	Both VDDCORE and VDDBU rising
1	WKUP_RST	VDDCORE rising
2	WDT_RST	Watchdog fault occurred
3	SOFT_RST	Processor reset required by the software
4	USER_RST	NRST pin detected low

NRSTL: NRST Pin Level

This bit registers the NRST pin level sampled on each Master Clock (MCK) rising edge.

SRCMP: Software Reset Command in Progress

When set, this bit indicates that a Software Reset Command is in progress and that no further software reset should be performed until the end of the current one. This bit is automatically cleared at the end of the current software reset.

0: No software command is being performed by the reset controller. The reset controller is ready for a software command.

1: A software reset command is being performed by the reset controller. The reset controller is busy.

13.5.3 Reset Controller Mode Register

Name:RSTC_MR

Address:0xFFFFFE08

Access Type: Read/Write

31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	ERSTL			
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

ERSTL: External Reset Length

This field defines the external reset length. The external reset is asserted during a time of $2^{(ERSTL+1)}$ Slow Clock cycles. This allows the assertion duration to be programmed between 60 µs and 2 seconds.

KEY: Write Access Password

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation. Always reads as 0.

© 2017 Microchip Technology Inc.

14. Real-time Clock (RTC)

14.1 Description

The Real-time Clock (RTC) peripheral is designed for very low power consumption. For optimal functionality, the RTC requires an accurate external 32.768 kHz clock, which can be provided by a crystal oscillator.

It combines a complete time-of-day clock with alarm and a Gregorian calendar, complemented by a programmable periodic interrupt. The alarm and calendar registers are accessed by a 32-bit data bus.

The time and calendar values are coded in binary-coded decimal (BCD) format. The time format can be 24-hour mode or 12-hour mode with an AM/PM indicator.

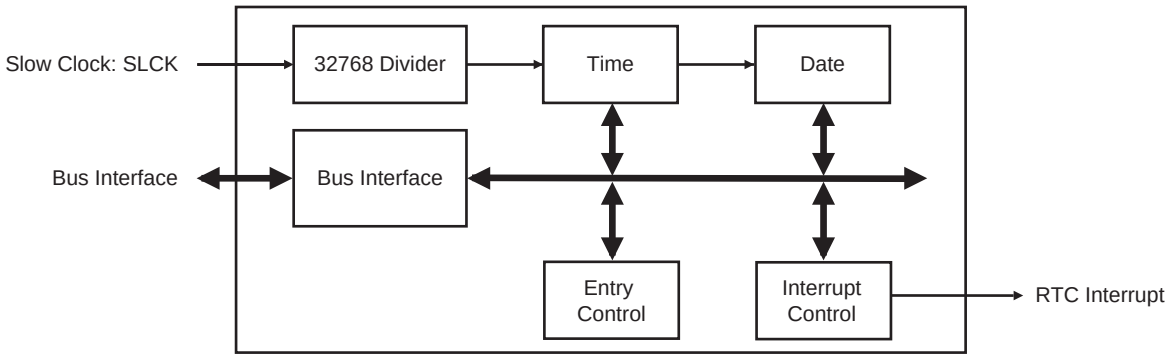
Updating time and calendar fields and configuring the alarm fields are performed by a parallel capture on the 32-bit data bus. An entry control is performed to avoid loading registers with incompatible BCD format data or with an incompatible date according to the current month/year/century.

14.2 Embedded Characteristics

- Full Asynchronous Design for Ultra Low Power Consumption
- Gregorian Mode Supported
- Programmable Periodic Interrupt
- Safety/security Features:
  - Valid Time and Date Programmation Check
- Register Write Protection

14.3 Block Diagram

Figure 14-1: Real-time Clock Block Diagram



14.4 Product Dependencies

14.4.1 Power Management

The Real-time Clock is continuously clocked at 32.768 kHz. The Power Management Controller has no effect on RTC behavior.

14.4.2 Interrupt

Within the System Controller, the RTC interrupt is OR-wired with all the other module interrupts.

Only one System Controller interrupt line is connected on one of the internal sources of the interrupt controller.

RTC interrupt requires the interrupt controller to be programmed first.

When a System Controller interrupt occurs, the service routine must first determine the cause of the interrupt. This is done by reading each status register of the System Controller peripherals successively.

14.5 Functional Description

The RTC provides a full binary-coded decimal (BCD) clock that includes century (19/20), year (with leap years), month, date, day, hours, minutes and seconds reported in RTC Time Register (RTC_TIMR) and RTC Calendar Register (RTC_CALR).

The valid year range is up to 2099 in Gregorian mode .

The RTC can operate in 24-hour mode or in 12-hour mode with an AM/PM indicator.

Corrections for leap years are included (all years divisible by 4 being leap years except 1900). This is correct up to the year 2099.

14.5.1 Reference Clock

The reference clock is the Slow Clock (SLCK). It can be driven internally or by an external 32.768 kHz crystal.

During low power modes of the processor, the oscillator runs and power consumption is critical. The crystal selection has to take into account the current consumption for power saving and the frequency drift due to temperature effect on the circuit for time accuracy.

14.5.2 Timing

The RTC is updated in real time at one-second intervals in Normal mode for the counters of seconds, at one-minute intervals for the counter of minutes and so on.

Due to the asynchronous operation of the RTC with respect to the rest of the chip, to be certain that the value read in the RTC registers (century, year, month, date, day, hours, minutes, seconds) are valid and stable, it is necessary to read these registers twice. If the data is the same both times, then it is valid. Therefore, a minimum of two and a maximum of three accesses are required.

14.5.3 Alarm

The RTC has five programmable fields: month, date, hours, minutes and seconds.

Each of these fields can be enabled or disabled to match the alarm condition:

- If all the fields are enabled, an alarm flag is generated (the corresponding flag is asserted and an interrupt generated if enabled) at a given month, date, hour/minute/second.
- If only the “seconds” field is enabled, then an alarm is generated every minute.

Depending on the combination of fields enabled, a large number of possibilities are available to the user ranging from minutes to 365/366 days.

Hour, minute and second matching alarm (SECEN, MINEN, HOUREN) can be enabled independently of SEC, MIN, HOUR fields.

Note: To change one of the SEC, MIN, HOUR, DATE, MONTH fields, it is recommended to disable the field before changing the value and then re-enable it after the change has been made. This requires up to three accesses to the RTC_TIMALR or RTC_CALALR. The first access clears the enable corresponding to the field to change (SECEN, MINEN, HOUREN, DATEEN, MTHEN). If the field is already cleared, this access is not required. The second access performs the change of the value (SEC, MIN, HOUR, DATE, MONTH). The third access is required to re-enable the field by writing 1 in SECEN, MINEN, HOUREn, DATEEN, MTHEN fields.

14.5.4 Error Checking when Programming

Verification on user interface data is performed when accessing the century, year, month, date, day, hours, minutes, seconds and alarms. A check is performed on illegal BCD entries such as illegal date of the month with regard to the year and century configured.

If one of the time fields is not correct, the data is not loaded into the register/counter and a flag is set in the validity register. The user can not reset this flag. It is reset as soon as an acceptable value is programmed. This avoids any further side effects in the hardware. The same procedure is followed for the alarm.

The following checks are performed:

1. Century (check if it is in range 19–20)
2. Year (BCD entry check)
3. Date (check range 01–31)
4. Month (check if it is in BCD range 01–12, check validity regarding “date”)
5. Day (check range 1–7)
6. Hour (BCD checks: in 24-hour mode, check range 00–23 and check that AM/PM flag is not set if RTC is set in 24-hour mode; in 12-hour mode check range 01–12)
7. Minute (check BCD and range 00–59)
8. Second (check BCD and range 00–59)

© 2017 Microchip Technology Inc.

Note: If the 12-hour mode is selected by means of the RTC Mode Register (RTC_MR), a 12-hour value can be programmed and the returned value on RTC_TIMR will be the corresponding 24-hour value. The entry control checks the value of the AM/PM indicator (bit 22 of RTC_TIMR) to determine the range to be checked.

14.5.5 Updating Time/Calendar

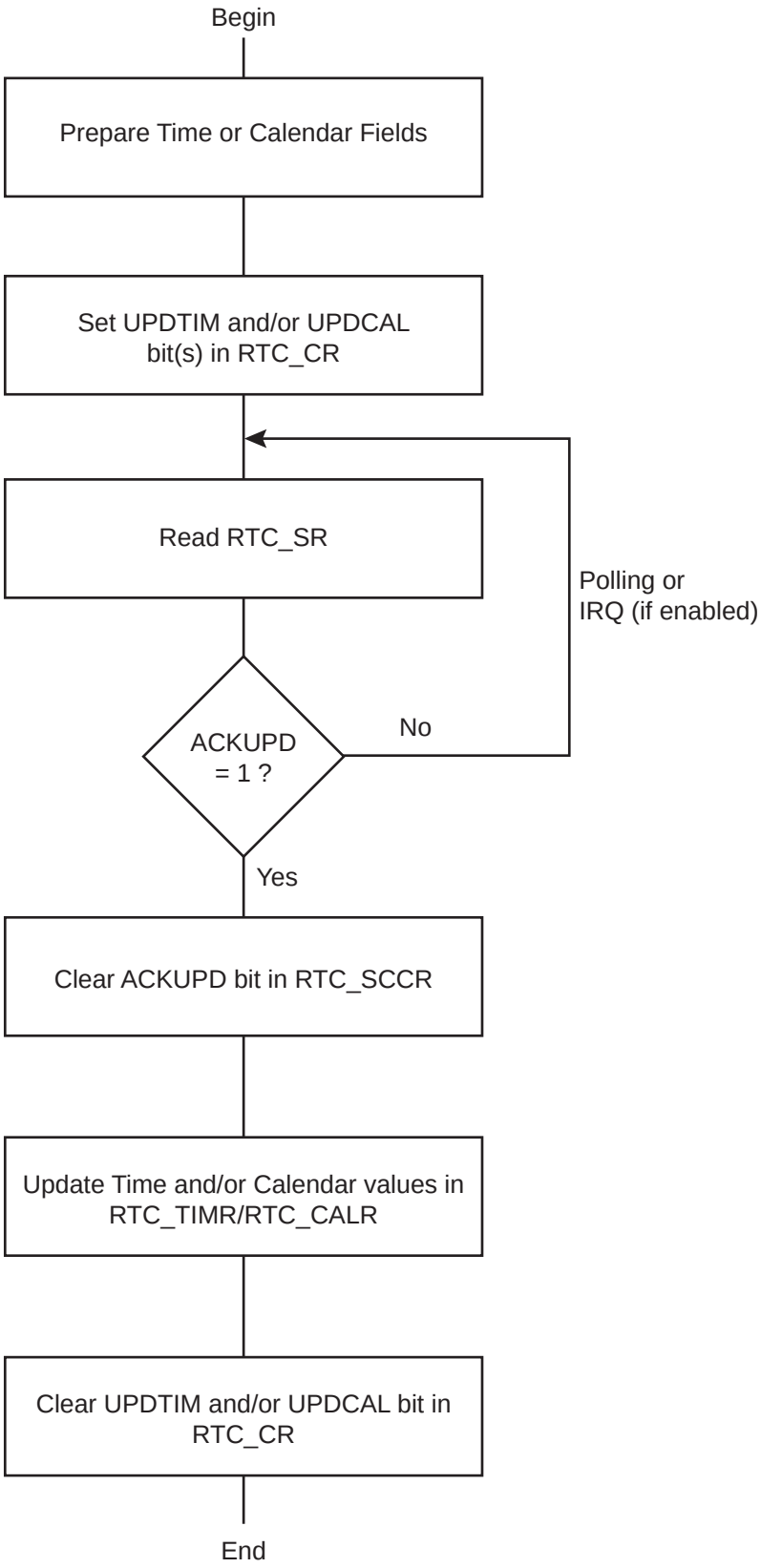
To update any of the time/calendar fields, the user must first stop the RTC by setting the corresponding field in the Control Register (RTC_CR). Bit UPDTIM must be set to update time fields (hour, minute, second) and bit UPDCAL must be set to update calendar fields (century, year, month, date, day).

The ACKUPD bit is automatically set within a second after setting the UPDTIM and/or UPDCAL bit (meaning one second is the maximum duration of the polling or wait for interrupt period). Once ACKUPD is set, it is mandatory to clear this flag by writing the corresponding bit in the RTC_SCCR, after which the user can write to the Time Register, the Calendar Register, or both.

Once the update is finished, the user must clear UPDTIM and/or UPDCAL in the RTC_CR.

When entering the programming mode of the calendar fields, the time fields remain enabled. When entering the programming mode of the time fields, both time and calendar fields are stopped. This is due to the location of the calendar logic circuity (downstream for low-power considerations). It is highly recommended to prepare all the fields to be updated before entering programming mode. In successive update operations, the user must wait at least one second after resetting the UPDTIM/UPDCAL bit in the RTC_CR before setting these bits again. This is done by waiting for the SEC flag in the RTC_SR before setting UPDTIM/UPDCAL bit. After clearing UPDTIM/UPDCAL, the SEC flag must also be cleared.

Figure 14-2: Update Sequence



© 2017 Microchip Technology Inc.

14.6 Real-time Clock (RTC) User Interface

Table 14-1: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	RTC_CR	Read/Write	0x00000000
0x04	Mode Register	RTC_MR	Read/Write	0x00000000
0x08	Time Register	RTC_TIMR	Read/Write	0x00000000
0x0C	Calendar Register	RTC_CALR	Read/Write	0x01210720
0x10	Time Alarm Register	RTC_TIMALR	Read/Write	0x00000000
0x14	Calendar Alarm Register	RTC_CALALR	Read/Write	0x01010000
0x18	Status Register	RTC_SR	Read-only	0x00000000
0x1C	Status Clear Command Register	RTC_SCCR	Write-only	–
0x20	Interrupt Enable Register	RTC_IER	Write-only	–
0x24	Interrupt Disable Register	RTC_IDR	Write-only	–
0x28	Interrupt Mask Register	RTC_IMR	Read-only	0x00000000
0x2C	Valid Entry Register	RTC_VER	Read-only	0x00000000
0x30–0xC8	Reserved	–	–	–
0xCC	Reserved	–	–	–
0xD0	Reserved	–	–	–
0xD4–0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

Note: If an offset is not listed in the table it must be considered as reserved.

14.6.1 RTC Control Register

Name: RTC_CR

Address:0xFFFFFEB0

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CALEVSEL	
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TIMEVSEL	
7	6	5	4	3	2	1	0
–	–	–	–	–	–	UPDCAL	UPDTIM

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_WPMR).

UPDTIM: Update Request Time Register

0: No effect or, if UPDTIM has been previously written to 1, stops the update procedure.

1: Stops the RTC time counting.

Time counting consists of second, minute and hour counters. Time counters can be programmed once this bit is set and acknowledged by the bit ACKUPD of the RTC_SR.

UPDCAL: Update Request Calendar Register

0: No effect or, if UPDCAL has been previously written to 1, stops the update procedure.

1: Stops the RTC calendar counting.

Calendar counting consists of day, date, month, year and century counters. Calendar counters can be programmed once this bit is set and acknowledged by the bit ACKUPD of the RTC_SR.

TIMEVSEL: Time Event Selection

The event that generates the flag TIMEV in RTC_SR depends on the value of TIMEVSEL.

Value	Name	Description
0	MINUTE	Minute change
1	HOUR	Hour change
2	MIDNIGHT	Every day at midnight
3	NOON	Every day at noon

CALEVSEL: Calendar Event Selection

The event that generates the flag CALEV in RTC_SR depends on the value of CALEVSEL

Value	Name	Description
0	WEEK	Week change (every Monday at time 00:00:00)
1	MONTH	Month change (every 01 of each month at time 00:00:00)
2	YEAR	Year change (every January 1 at time 00:00:00)

© 2017 Microchip Technology Inc.

14.6.2 RTC Mode Register

Name: RTC_MR

Address:0xFFFFFEB4

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	HRMOD

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_WPMR).

HRMOD: 12-/24-hour Mode

0: 24-hour mode is selected.

1: 12-hour mode is selected.

14.6.3 RTC Time Register

Name: RTC_TIMR

Address:0xFFFFFEB8

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	AMPM	HOUR					

15	14	13	12	11	10	9	8
–	MIN						

7	6	5	4	3	2	1	0
–	SEC						

SEC: Current Second

The range that can be set is 0–59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

MIN: Current Minute

The range that can be set is 0–59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

HOUR: Current Hour

The range that can be set is 1–12 (BCD) in 12-hour mode or 0–23 (BCD) in 24-hour mode.

AMPM: Ante Meridiem Post Meridiem Indicator

This bit is the AM/PM indicator in 12-hour mode.

0: AM.

1: PM.

© 2017 Microchip Technology Inc.

14.6.4 RTC Calendar Register

Name: RTC_CALR

Address:0xFFFFFEBC

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	DATE					

23	22	21	20	19	18	17	16
DAY			MONTH				

15	14	13	12	11	10	9	8
YEAR							

7	6	5	4	3	2	1	0
–	CENT						

CENT: Current Century

Only the BCD value 20 can be configured.

The lowest four bits encode the units. The higher bits encode the tens.

YEAR: Current Year

The range that can be set is 00–99 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

MONTH: Current Month

The range that can be set is 01–12 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

DAY: Current Day in Current Week

The range that can be set is 1–7 (BCD).

The coding of the number (which number represents which day) is user-defined as it has no effect on the date counter.

DATE: Current Day in Current Month

The range that can be set is 01–31 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

© 2017 Microchip Technology Inc.

14.6.5 RTC Time Alarm Register

Name: RTC_TIMALR

Address:0xFFFFFEC0

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
HOUREN	AMPM	HOUR					
15	14	13	12	11	10	9	8
MINEN	MIN						
7	6	5	4	3	2	1	0
SECEN	SEC						

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_WPMR).

Note: To change one of the SEC, MIN, HOUR fields, it is recommended to disable the field before changing the value and then re-enable it after the change has been made. This requires up to three accesses to the RTC_TIMALR. The first access clears the enable corresponding to the field to change (SECEN, MINEN, HOUREN). If the field is already cleared, this access is not required. The second access performs the change of the value (SEC, MIN, HOUR). The third access is required to re-enable the field by writing 1 in SECEN, MINEN, HOUREN fields.

SEC: Second Alarm

This field is the alarm field corresponding to the BCD-coded second counter.

SECEN: Second Alarm Enable

0: The second-matching alarm is disabled.

1: The second-matching alarm is enabled.

MIN: Minute Alarm

This field is the alarm field corresponding to the BCD-coded minute counter.

MINEN: Minute Alarm Enable

0: The minute-matching alarm is disabled.

1: The minute-matching alarm is enabled.

HOUR: Hour Alarm

This field is the alarm field corresponding to the BCD-coded hour counter.

AMPM: AM/PM Indicator

This field is the alarm field corresponding to the BCD-coded hour counter.

HOUREN: Hour Alarm Enable

0: The hour-matching alarm is disabled.

1: The hour-matching alarm is enabled.

© 2017 Microchip Technology Inc.

14.6.6 RTC Calendar Alarm Register

Name: RTC_CALALR

Address:0xFFFFFEC4

Access: Read/Write

31	30	29	28	27	26	25	24
DATEEN	–	DATE					
23	22	21	20	19	18	17	16
MTHEN	–	–	MONTH				
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_WPMR).

Note: To change one of the DATE, MONTH fields, it is recommended to disable the field before changing the value and then re-enable it after the change has been made. This requires up to three accesses to the RTC_CALALR. The first access clears the enable corresponding to the field to change (DATEEN, MTHEN). If the field is already cleared, this access is not required. The second access performs the change of the value (DATE, MONTH). The third access is required to re-enable the field by writing 1 in DATEEN, MTHEN fields.

MONTH: Month Alarm

This field is the alarm field corresponding to the BCD-coded month counter.

MTHEN: Month Alarm Enable

0: The month-matching alarm is disabled.

1: The month-matching alarm is enabled.

DATE: Date Alarm

This field is the alarm field corresponding to the BCD-coded date counter.

DATEEN: Date Alarm Enable

0: The date-matching alarm is disabled.

1: The date-matching alarm is enabled.

14.6.7 RTC Status Register

Name: RTC_SR

Address:0xFFFFFEC8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	CALEV	TIMEV	SEC	ALARM	ACKUPD

ACKUPD: Acknowledge for Update

Value	Name	Description
0	FREERUN	Time and calendar registers cannot be updated.
1	UPDATE	Time and calendar registers can be updated.

ALARM: Alarm Flag

Value	Name	Description
0	NO_ALARMEVENT	No alarm matching condition occurred.
1	ALARMEVENT	An alarm matching condition has occurred.

SEC: Second Event

Value	Name	Description
0	NO_SECEVENT	No second event has occurred since the last clear.
1	SECEVENT	At least one second event has occurred since the last clear.

TIMEV: Time Event

Value	Name	Description
0	NO_TIMEVENT	No time event has occurred since the last clear.
1	TIMEVENT	At least one time event has occurred since the last clear.

Note: The time event is selected in the TIMEVSEL field in the Control Register (RTC_CR) and can be any one of the following events: minute change, hour change, noon, midnight (day change).

CALEV: Calendar Event

Value	Name	Description
0	NO_CALEVENT	No calendar event has occurred since the last clear.
1	CALEVENT	At least one calendar event has occurred since the last clear.

Note: The calendar event is selected in the CALEVSEL field in the Control Register (RTC_CR) and can be any one of the following events: week change, month change and year change.

© 2017 Microchip Technology Inc.

14.6.8 RTC Status Clear Command Register

Name: RTC_SCCR

Address:0xFFFFFECC

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	CALCLR	TIMCLR	SECCLR	ALRCLR	ACKCLR

ACKCLR: Acknowledge Clear

0: No effect.

1: Clears corresponding status flag in the Status Register (RTC_SR).

ALRCLR: Alarm Clear

0: No effect.

1: Clears corresponding status flag in the Status Register (RTC_SR).

SECCLR: Second Clear

0: No effect.

1: Clears corresponding status flag in the Status Register (RTC_SR).

TIMCLR: Time Clear

0: No effect.

1: Clears corresponding status flag in the Status Register (RTC_SR).

CALCLR: Calendar Clear

0: No effect.

1: Clears corresponding status flag in the Status Register (RTC_SR).

14.6.9 RTC Interrupt Enable Register

Name: RTC_IER

Address:0xFFFFFED0

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	CALEN	TIMEN	SECEN	ALREN	ACKEN

ACKEN: Acknowledge Update Interrupt Enable

0: No effect.

1: The acknowledge for update interrupt is enabled.

ALREN: Alarm Interrupt Enable

0: No effect.

1: The alarm interrupt is enabled.

SECEN: Second Event Interrupt Enable

0: No effect.

1: The second periodic interrupt is enabled.

TIMEN: Time Event Interrupt Enable

0: No effect.

1: The selected time event interrupt is enabled.

CALEN: Calendar Event Interrupt Enable

0: No effect.

1: The selected calendar event interrupt is enabled.

© 2017 Microchip Technology Inc.

14.6.10 RTC Interrupt Disable Register

Name: RTC_IDR

Address:0xFFFFFED4

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	CALDIS	TIMDIS	SECDIS	ALRDIS	ACKDIS

ACKDIS: Acknowledge Update Interrupt Disable

0: No effect.

1: The acknowledge for update interrupt is disabled.

ALRDIS: Alarm Interrupt Disable

0: No effect.

1: The alarm interrupt is disabled.

SECDIS: Second Event Interrupt Disable

0: No effect.

1: The second periodic interrupt is disabled.

TIMDIS: Time Event Interrupt Disable

0: No effect.

1: The selected time event interrupt is disabled.

CALDIS: Calendar Event Interrupt Disable

0: No effect.

1: The selected calendar event interrupt is disabled.

14.6.11 RTC Interrupt Mask Register

Name: RTC_IMR

Address:0xFFFFFED8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	CAL	TIM	SEC	ALR	ACK

ACK: Acknowledge Update Interrupt Mask

0: The acknowledge for update interrupt is disabled.

1: The acknowledge for update interrupt is enabled.

ALR: Alarm Interrupt Mask

0: The alarm interrupt is disabled.

1: The alarm interrupt is enabled.

SEC: Second Event Interrupt Mask

0: The second periodic interrupt is disabled.

1: The second periodic interrupt is enabled.

TIM: Time Event Interrupt Mask

0: The selected time event interrupt is disabled.

1: The selected time event interrupt is enabled.

CAL: Calendar Event Interrupt Mask

0: The selected calendar event interrupt is disabled.

1: The selected calendar event interrupt is enabled.

© 2017 Microchip Technology Inc.

14.6.12 RTC Valid Entry Register

Name: RTC_VER

Address:0xFFFFFEDC

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	NVCALALR	NVTIMALR	NVCAL	NVTIM

NVTIM: Non-valid Time

0: No invalid data has been detected in RTC_TIMR (Time Register).

1: RTC_TIMR has contained invalid data since it was last programmed.

NVCAL: Non-valid Calendar

0: No invalid data has been detected in RTC_CALR (Calendar Register).

1: RTC_CALR has contained invalid data since it was last programmed.

NVTIMALR: Non-valid Time Alarm

0: No invalid data has been detected in RTC_TIMALR (Time Alarm Register).

1: RTC_TIMALR has contained invalid data since it was last programmed.

NVCALALR: Non-valid Calendar Alarm

0: No invalid data has been detected in RTC_CALALR (Calendar Alarm Register).

1: RTC_CALALR has contained invalid data since it was last programmed.

© 2017 Microchip Technology Inc.

15. Periodic Interval Timer (PIT)

15.1 Description

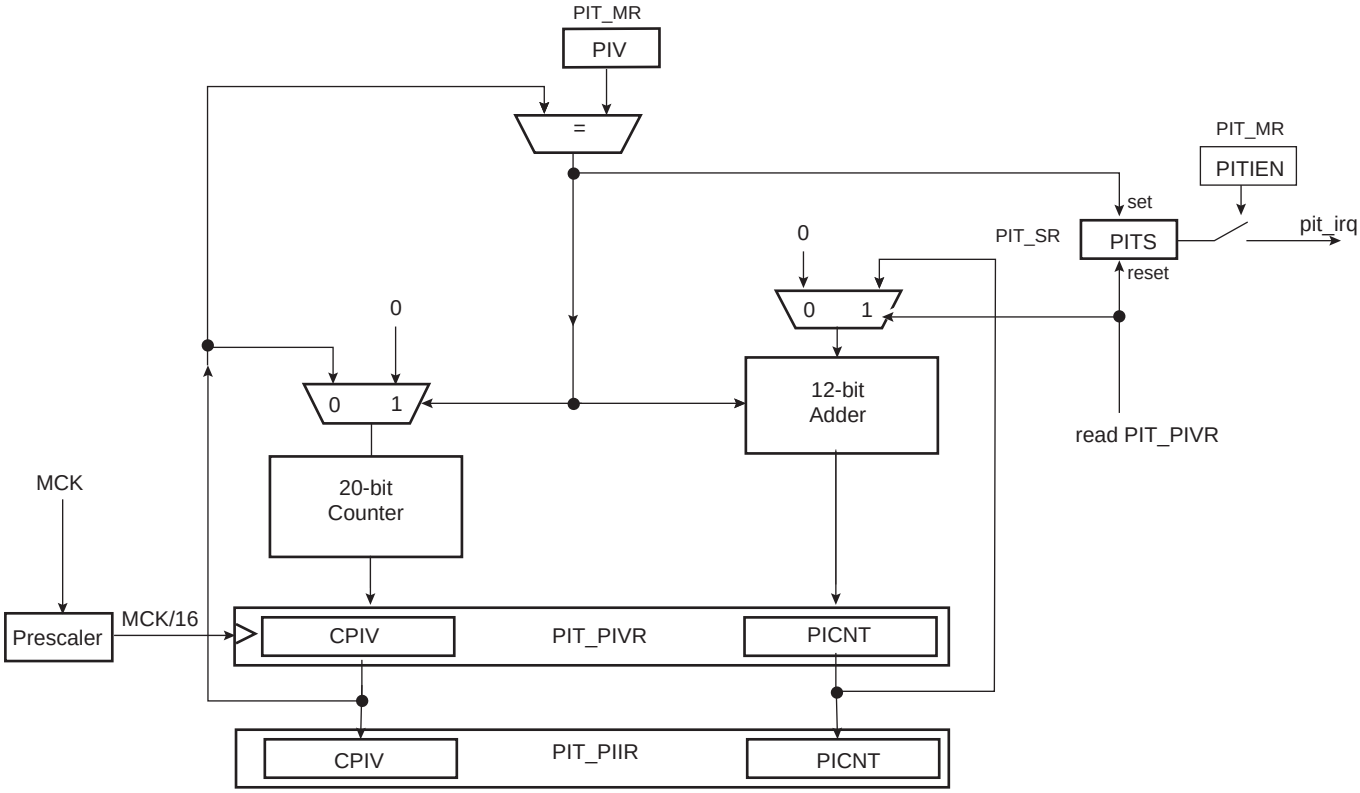
The Periodic Interval Timer (PIT) provides the operating system's scheduler interrupt. It is designed to offer maximum accuracy and efficient management, even for systems with long response time.

15.2 Embedded Characteristics

- 20-bit Programmable Counter plus 12-bit Interval Counter
- Reset-on-read Feature
- Both Counters Work on Master Clock/16

15.3 Block Diagram

Figure 15-1: Periodic Interval Timer



15.4 Functional Description

The Periodic Interval Timer aims at providing periodic interrupts for use by operating systems.

The PIT provides a programmable overflow counter and a reset-on-read feature. It is built around two counters: a 20-bit CPIV counter and a 12-bit PICNT counter. Both counters work at Master Clock /16.

The first 20-bit CPIV counter increments from 0 up to a programmable overflow value set in the field PIV of the Mode Register (PIT_MR). When the counter CPIV reaches this value, it resets to 0 and increments the Periodic Interval Counter, PICNT. The status bit PITS in the Status Register (PIT_SR) rises and triggers an interrupt, provided the interrupt is enabled (PITIEN in PIT_MR).

Writing a new PIV value in PIT_MR does not reset/restart the counters.

When CPIV and PICNT values are obtained by reading the Periodic Interval Value Register (PIT_PIVR), the overflow counter (PICNT) is reset and the PITS bit is cleared, thus acknowledging the interrupt. The value of PICNT gives the number of periodic intervals elapsed since the last read of PIT_PIVR.

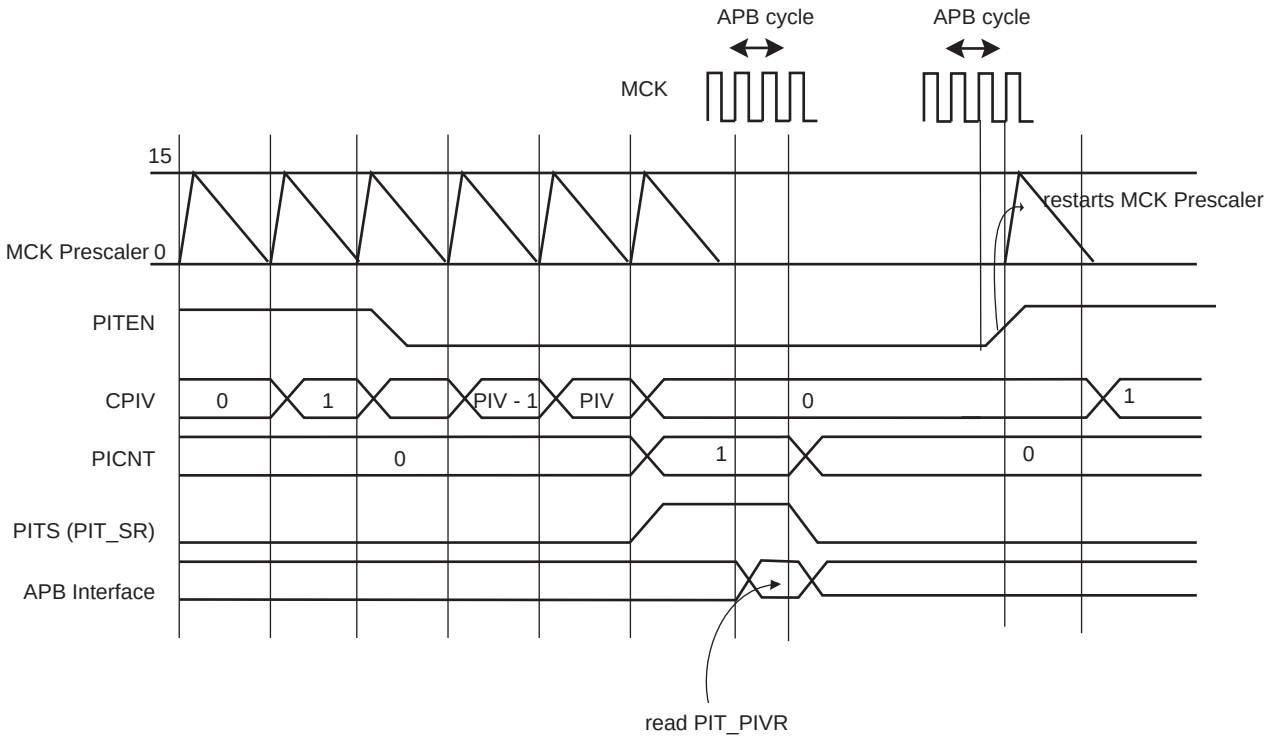
© 2017 Microchip Technology Inc.

When CPIV and PICNT values are obtained by reading the Periodic Interval Image Register (PIT_PIIR), there is no effect on the counters CPIV and PICNT, nor on the bit PITS. For example, a profiler can read PIT_PIIR without clearing any pending interrupt, whereas a timer interrupt clears the interrupt by reading PIT_PIVR.

The PIT may be enabled/disabled using the PITEN bit in the PIT_MR register (disabled on reset). The PITEN bit only becomes effective when the CPIV value is 0. Figure 15-2 illustrates the PIT counting. After the PIT Enable bit is reset (PITEN = 0), the CPIV goes on counting until the PIV value is reached, and is then reset. PIT restarts counting, only if the PITEN is set again.

The PIT is stopped when the core enters debug state.

Figure 15-2: Enabling/Disabling PIT with PITEN



15.5 Periodic Interval Timer (PIT) User Interface

Table 15-1: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Mode Register	PIT_MR	Read/Write	0x000F_FFFF
0x04	Status Register	PIT_SR	Read-only	0x0000_0000
0x08	Periodic Interval Value Register	PIT_PIVR	Read-only	0x0000_0000
0x0C	Periodic Interval Image Register	PIT_PIIR	Read-only	0x0000_0000

© 2017 Microchip Technology Inc.

15.5.1 Periodic Interval Timer Mode Register

Name:PIT_MR

Address:0xFFFFFE30

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	PITIEN	PITEN

23	22	21	20	19	18	17	16
–	–	–	–	PIV			

15	14	13	12	11	10	9	8
PIV							

7	6	5	4	3	2	1	0
PIV							

PIV: Periodic Interval Value

Defines the value compared with the primary 20-bit counter of the Periodic Interval Timer (CPIV). The period is equal to (PIV + 1).

PITEN: Period Interval Timer Enabled

0: The Periodic Interval Timer is disabled when the PIV value is reached.

1: The Periodic Interval Timer is enabled.

PITIEN: Periodic Interval Timer Interrupt Enable

0: The bit PITS in PIT_SR has no effect on interrupt.

1: The bit PITS in PIT_SR asserts interrupt.

© 2017 Microchip Technology Inc.

15.5.2 Periodic Interval Timer Status Register

Name:PIT_SR

Address:0xFFFFFE34

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	PITS

PITS: Periodic Interval Timer Status

0: The Periodic Interval timer has not reached PIV since the last read of PIT_PIVR.

1: The Periodic Interval timer has reached PIV since the last read of PIT_PIVR.

© 2017 Microchip Technology Inc.

15.5.3 Periodic Interval Timer Value Register

Name:PIT_PIVR

Address:0xFFFFFE38

Access: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

Reading this register clears PITS in PIT_SR.

CPIV: Current Periodic Interval Value

Returns the current value of the periodic interval timer.

PICNT: Periodic Interval Counter

Returns the number of occurrences of periodic intervals since the last read of PIT_PIVR.

15.5.4 Periodic Interval Timer Image Register

Name:PIT_PIIR

Address:0xFFFFFE3C

Access: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

CPIV: Current Periodic Interval Value

Returns the current value of the periodic interval timer.

PICNT: Periodic Interval Counter

Returns the number of occurrences of periodic intervals since the last read of PIT_PIVR.

© 2017 Microchip Technology Inc.

16. Watchdog Timer (WDT)

16.1 Description

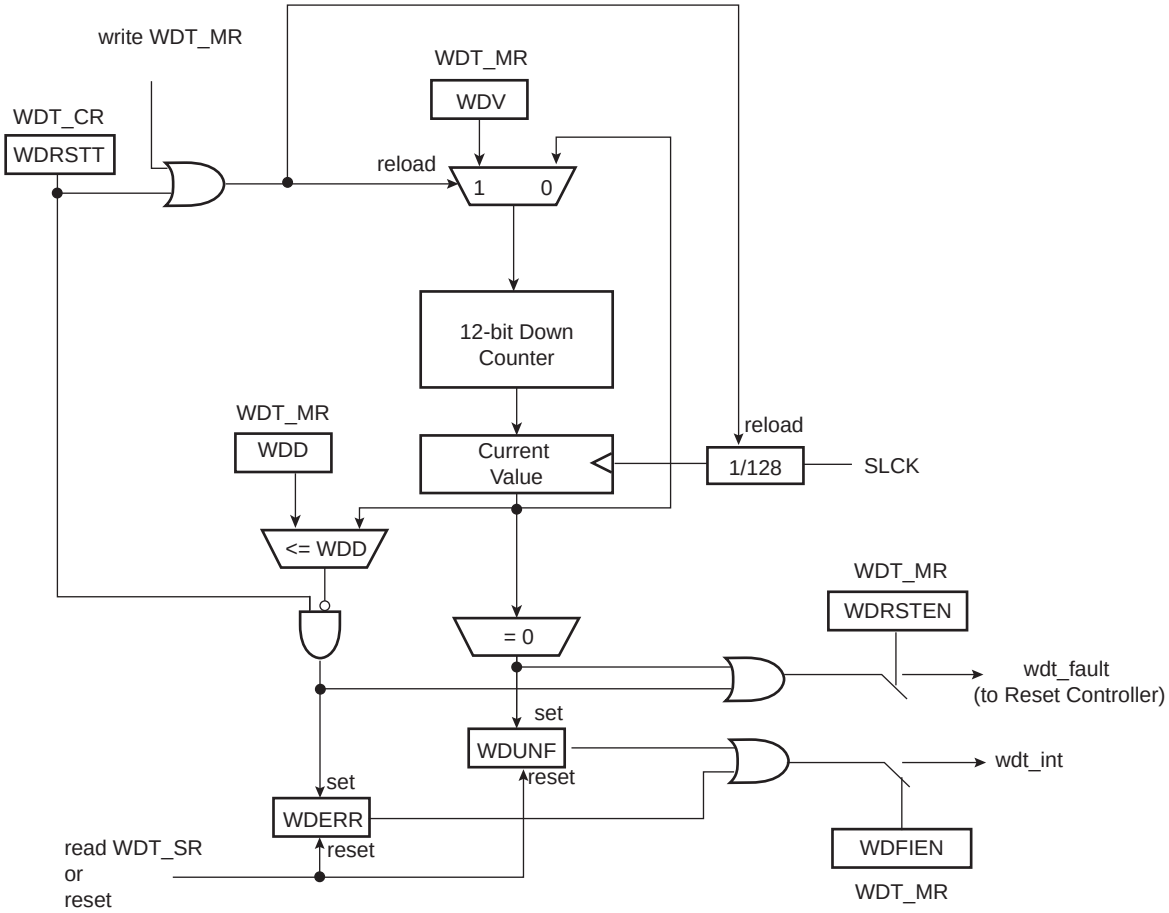
The Watchdog Timer (WDT) is used to prevent system lock-up if the software becomes trapped in a deadlock. It features a 12-bit down counter that allows a watchdog period of up to 16 seconds (slow clock around 32 kHz). It can generate a general reset or a processor reset only. In addition, it can be stopped while the processor is in Debug mode or Idle mode.

16.2 Embedded Characteristics

- 12-bit Key-protected Programmable Counter
- Watchdog Clock is Independent from Processor Clock
- Provides Reset or Interrupt Signals to the System
- Counter May Be Stopped while the Processor is in Debug State or in Idle Mode

16.3 Block Diagram

Figure 16-1: Watchdog Timer Block Diagram



16.4 Functional Description

The Watchdog Timer is used to prevent system lock-up if the software becomes trapped in a deadlock. It is supplied with VDDCORE. It restarts with initial values on processor reset.

The watchdog is built around a 12-bit down counter, which is loaded with the value defined in the field WDV of the Mode Register (WDT_MR). The Watchdog Timer uses the slow clock divided by 128 to establish the maximum watchdog period to be 16 seconds (with a typical slow clock of 32.768 kHz).

After a processor reset, the value of WDV is 0xFFF, corresponding to the maximum value of the counter with the external reset generation enabled (field WDRSTEN at 1 after a backup reset). This means that a default watchdog is running at reset, i.e., at power-up. The user can either disable the WDT by setting bit WDT_MR.WDDIS or reprogram the WDT to meet the maximum watchdog period the application requires.

When setting the WDDIS bit, and while it is set, the fields WDV and WDD must not be modified.

If the watchdog is restarted by writing into the Control Register (WDT_CR), WDT_MR must not be programmed during a period of time of three slow clock periods following the WDT_CR write access. In any case, programming a new value in WDT_MR automatically initiates a restart instruction.

WDT_MR can be written only once. Only a processor reset resets it. Writing WDT_MR reloads the timer with the newly programmed mode parameters.

In normal operation, the user reloads the watchdog at regular intervals before the timer underflow occurs, by setting bit WDT_CR.WDRSTT. The watchdog counter is then immediately reloaded from WDT_MR and restarted, and the slow clock 128 divider is reset and restarted. WDT_CR is write-protected. As a result, writing WDT_CR without the correct hard-coded key has no effect. If an underflow does occur, the “wdt_fault” signal to the Reset Controller is asserted if bit WDT_MR.WDRSTEN is set. Moreover, the bit WDUNF is set in the Status Register (WDT_SR).

To prevent a software deadlock that continuously triggers the watchdog, the reload of the watchdog must occur while the watchdog counter is within a window between 0 and WDD. WDD is defined in WDT_MR.

Any attempt to restart the watchdog while the watchdog counter is between WDV and WDD results in a watchdog error, even if the watchdog is disabled. The bit WDT_SR.WDERR is updated and the “wdt_fault” signal to the Reset Controller is asserted.

Note that this feature can be disabled by programming a WDD value greater than or equal to the WDV value. In such a configuration, restarting the Watchdog Timer is permitted in the whole range [0; WDV] and does not generate an error. This is the default configuration on reset (the WDD and WDV values are equal).

The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDT_MR.WDFIEN is set. The signal “wdt_fault” to the Reset Controller causes a watchdog reset if the WDRSTEN bit is set as already explained in the Reset Controller documentation. In this case, the processor and the Watchdog Timer are reset, and the WDERR and WDUNF flags are reset.

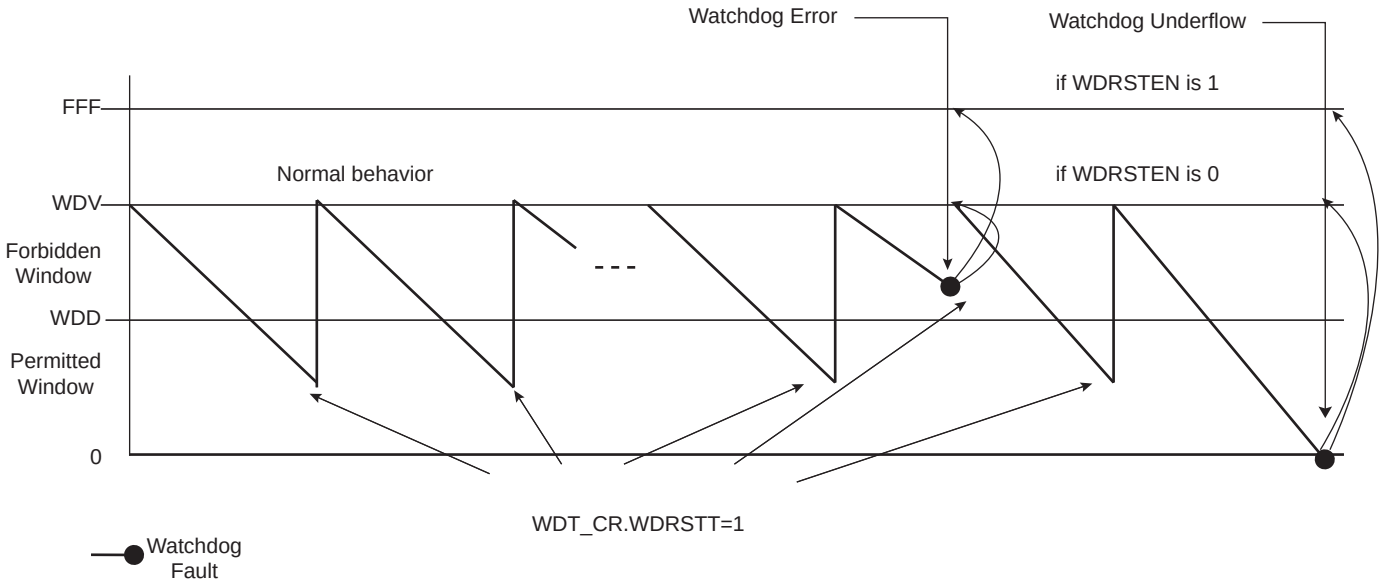
If a reset is generated or if WDT_SR is read, the status bits are reset, the interrupt is cleared, and the “wdt_fault” signal to the reset controller is deasserted.

Writing WDT_MR reloads and restarts the down counter.

While the processor is in debug state or in idle mode, the counter may be stopped depending on the value programmed for the bits WDIDLEHLT and WDDBGHLT in WDT_MR.

© 2017 Microchip Technology Inc.

Figure 16-2: Watchdog Behavior



16.5 Watchdog Timer (WDT) User Interface

Table 16-1: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	WDT_CR	Write-only	–
0x04	Mode Register	WDT_MR	Read/Write Once	0x3FFF_2FFF
0x08	Status Register	WDT_SR	Read-only	0x0000_0000

© 2017 Microchip Technology Inc.

16.5.1 Watchdog Timer Control Register

Name:WDT_CR

Address:0xFFFFFE40

Access: Write-only

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WDRSTT

Note: The WDT_CR register values must not be modified within three slow clock periods following a restart of the watchdog performed by a write access in WDT_CR. Any modification will cause the watchdog to trigger an end of period earlier than expected.

WDRSTT: Watchdog Restart

0: No effect.

1: Restarts the watchdog if KEY is written to 0xA5.

KEY: Password

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation.

16.5.2 Watchdog Timer Mode Register

Name:WDT_MR

Address:0xFFFFFE44

Access: Read/Write Once

31	30	29	28	27	26	25	24
–	–	WDIDLEHLT	WDDBGHLT	WDD			
23	22	21	20	19	18	17	16
WDD							
15	14	13	12	11	10	9	8
WDDIS	WDRPROC	WDRSTEN	WDFIEN	WDV			
7	6	5	4	3	2	1	0
WDV							

Note 1: The first write access prevents any further modification of the value of this register. Read accesses remain possible.

2: The WDT_MR register values must not be modified within three slow clock periods following a restart of the watchdog performed by a write access in WDT_CR. Any modification will cause the watchdog to trigger an end of period earlier than expected.

WDV: Watchdog Counter Value

Defines the value loaded in the 12-bit watchdog counter.

WDFIEN: Watchdog Fault Interrupt Enable

0: A watchdog fault (underflow or error) has no effect on interrupt.

1: A watchdog fault (underflow or error) asserts interrupt.

WDRSTEN: Watchdog Reset Enable

0: A watchdog fault (underflow or error) has no effect on the resets.

1: A watchdog fault (underflow or error) triggers a watchdog reset.

WDRPROC: Watchdog Reset Processor

0: If WDRSTEN is 1, a watchdog fault (underflow or error) activates all resets.

1: If WDRSTEN is 1, a watchdog fault (underflow or error) activates the processor reset.

WDDIS: Watchdog Disable

0: Enables the Watchdog Timer.

1: Disables the Watchdog Timer.

Note: When setting the WDDIS bit, and while it is set, the fields WDV and WDD must not be modified.

WDD: Watchdog Delta Value

Defines the permitted range for reloading the Watchdog Timer.

If the Watchdog Timer value is less than or equal to WDD, setting bit WDT_CR.WDRSTT restarts the timer.

If the Watchdog Timer value is greater than WDD, setting bit WDT_CR.WDRSTT causes a watchdog error.

WDDBGHLT: Watchdog Debug Halt

0: The watchdog runs when the processor is in debug state.

1: The watchdog stops when the processor is in debug state.

© 2017 Microchip Technology Inc.

WDIDLEHLT: Watchdog Idle Halt

0: The watchdog runs when the system is in idle state.

1: The watchdog stops when the system is in idle state.

16.5.3 Watchdog Timer Status Register

Name:WDT_SR

Address:0xFFFFFE48

AccessRead-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDERR	WDUNF

WDUNF: Watchdog Underflow (cleared on read)

0: No watchdog underflow occurred since the last read of WDT_SR.

1: At least one watchdog underflow occurred since the last read of WDT_SR.

WDERR: Watchdog Error (cleared on read)

0: No watchdog error occurred since the last read of WDT_SR.

1: At least one watchdog error occurred since the last read of WDT_SR.

© 2017 Microchip Technology Inc.

17. Shutdown Controller (SHDWC)

17.1 Description

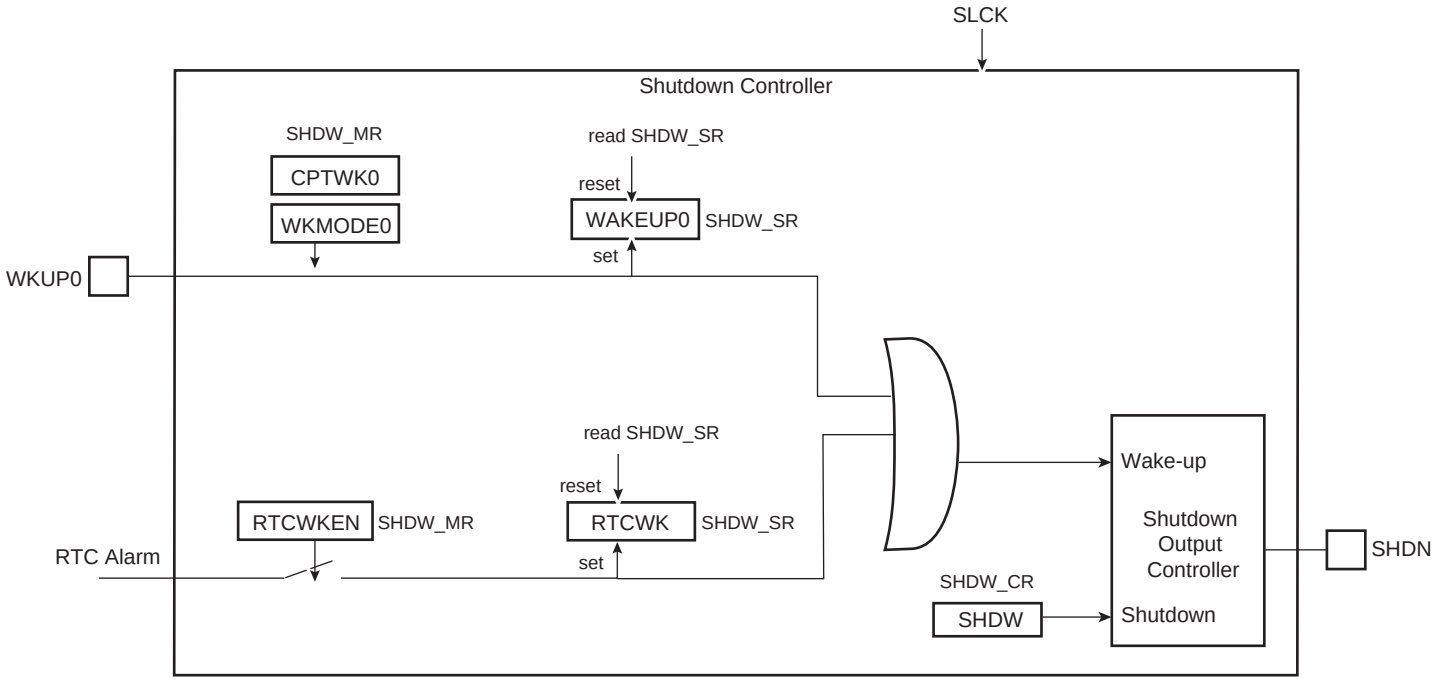
The Shutdown Controller (SHDWC) controls the power supplies VDDIO and VDDCORE and the wake-up detection on debounced input lines.

17.2 Embedded Characteristics

- Shutdown Logic
  - Software Assertion of the Shutdown Output Pin (SHDN)
  - Programmable De-assertion from the WKUP Input Pins
- Wake-up Logic
  - Programmable Assertion from the WKUP Input Pins, and Internal Wake-up Event from RTC

17.3 Block Diagram

Figure 17-1: Shutdown Controller Block Diagram



17.4 I/O Lines Description

Table 17-1: I/O Lines Description

Name	Description	Type
WKUP0	Wake-up 0 input	Input
SHDN	Shutdown output	Output

17.5 Product Dependencies

17.5.1 Power Management

The Shutdown Controller is continuously clocked by the Slow Clock (SLCK). The Power Management Controller has no effect on the behavior of the Shutdown Controller.

17.6 Functional Description

The Shutdown Controller manages the main power supply. To do so, it is supplied with VDDBU and manages wake-up input pins and one output pin, SHDN.

A typical application connects the pin SHDN to the shutdown input of the DC/DC Converter providing the main power supplies of the system, and especially VDDCORE and/or VDDIO. The wake-up inputs (WKUP0) connect to any push-buttons or signal that wake up the system.

The software is able to control the pin SHDN by writing the Shutdown Control Register (SHDW_CR) with the bit SHDW at 1. The shutdown is taken into account only two slow clock cycles after the write of SHDW_CR. This register is password-protected and so the value written should contain the correct key for the command to be taken into account. As a result, the system should be powered down.

17.6.1 Wake-up Inputs

A level change on WKUP0 can trigger a wake-up. Wake-up is configured in the Shutdown Mode Register (SHDW_MR). The transition detector can be programmed to detect either a positive or negative transition or any level change on WKUP0. The detection can also be disabled. Programming is performed by defining WKMODE0

Moreover, a debouncing circuit can be programmed for WKUP0. The debouncing circuit filters pulses on WKUP0 shorter than the programmed number of 16 SLCK cycles in CPTWK0 of the SHDW_MR. If the programmed level change is detected on a pin, a counter starts. When the counter reaches the value programmed in the corresponding field, CPTWK0, the SHDN pin is released. If a new input change is detected before the counter reaches the corresponding value, the counter is stopped and cleared. WAKEUP0 of the Status Register (SHDW_SR) reports the detection of the programmed events on WKUP0 with a reset after the read of SHDW_SR.

The Shutdown Controller can be programmed so as to activate the wake-up using the RTC alarm(detection of the rising edge event is synchronized with SLCK). This is done by writing the SHDW_MR using the RTCWKEN bit. When enabled, the detection of RTC alarm is reported in the RTCWK bit of SHDW_SR. They are cleared after reading SHDW_SR. When using the RTC alarm to wake up the system, the user must ensure that RTC alarmstatus flag is cleared before shutting down the system. Otherwise, no rising edge of the status flags may be detected and the wake-up will fail.

© 2017 Microchip Technology Inc.

17.7 Shutdown Controller (SHDWC) User Interface

Table 17-2: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Shutdown Control Register	SHDW_CR	Write-only	–
0x04	Shutdown Mode Register	SHDW_MR	Read/Write	0x0000_0003
0x08	Shutdown Status Register	SHDW_SR	Read-only	0x0000_0000

17.7.1 Shutdown Control Register

Name:SHDW_CR

Address:0xFFFFFE10

Access: Write-only

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SHDW

SHDW: Shutdown Command

0: No effect.

1: If KEY value is correct, asserts the SHDN pin.

KEY: Password

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation.

© 2017 Microchip Technology Inc.

17.7.2 Shutdown Mode Register

Name:SHDW_MR

Address:0xFFFFFE14

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	RTCWKEN	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
CPTWK0				–	–	WKMODE0	

WKMODE0: Wake-up Mode 0

Value	Name	Description
0	NO_DETECTION	No detection is performed on the wake-up input
1	RISING_EDGE	Low to high transition triggers the detection process
2	FALLING_EDGE	High to low level transition triggers the detection process
3	ANY_EDGE	Any edge on the wake-up input triggers the detection process

CPTWK0: Debounce Counter on Wake-up 0

Defines the minimum duration of the WKUP1 pin after the occurrence of the selected triggering edge (WKMODE0).

The SHDN pin is released if the WKUP0 holds the selected level for (CPTWK $\times$ 16 + 1) consecutive Slow Clock cycles after the occurrence of the selected triggering edge on WKUP0.

RTCWKEN: Real-time Clock Wake-up Enable

0: The RTC Alarm signal has no effect on the Shutdown Controller.

1: The RTC Alarm signal forces the de-assertion of the SHDN pin.

17.7.3 Shutdown Status Register

Name:SHDW_SR

Address:0xFFFFFE18

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RTCWK	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WAKEUP0

WAKEUP0: Wake-up 0 Status

0: No wake-up event occurred on WKUP0 input since the last read of SHDW_SR.

1: At least one wake-up event occurred on WKUP0 input since the last read of SHDW_SR.

RTCWK: Real-time Clock Wake-up

0: No wake-up alarm from the RTC occurred since the last read of SHDW_SR.

1: At least one wake-up alarm from the RTC occurred since the last read of SHDW_SR.

© 2017 Microchip Technology Inc.

18. General Purpose Backup Registers (GPBR)

18.1 Description

The System Controller embeds 128 bits of General Purpose Backup registers organized as four 32-bit registers.

18.2 Embedded Characteristics

- 128 bits of General Purpose Backup Registers

18.3 General Purpose Backup Registers (GPBR) User Interface

Table 18-1: Register Mapping

Offset	Register	Name	Access	Reset
0x0	General Purpose Backup Register 0	SYS_GPBR0	Read/Write	0x00000000
...	...	...	...	...
0xC	General Purpose Backup Register 3	SYS_GPBR3	Read/Write	0x00000000

© 2017 Microchip Technology Inc.

18.3.1 General Purpose Backup Register x

Name:SYS_GPBRx

Address:0xFFFFFE60

Access: Read/Write

31	30	29	28	27	26	25	24
GPBR_VALUE							
23	22	21	20	19	18	17	16
GPBR_VALUE							
15	14	13	12	11	10	9	8
GPBR_VALUE							
7	6	5	4	3	2	1	0
GPBR_VALUE							

These registers are reset at first power-up and on each loss of VDDBU.

GPBR_VALUE: Value of GPBR x

19. Slow Clock Controller (SCKC)

19.1 Description

The System Controller embeds a Slow Clock Controller (SCKC). The SCKC selects the slow clock from one of two sources:

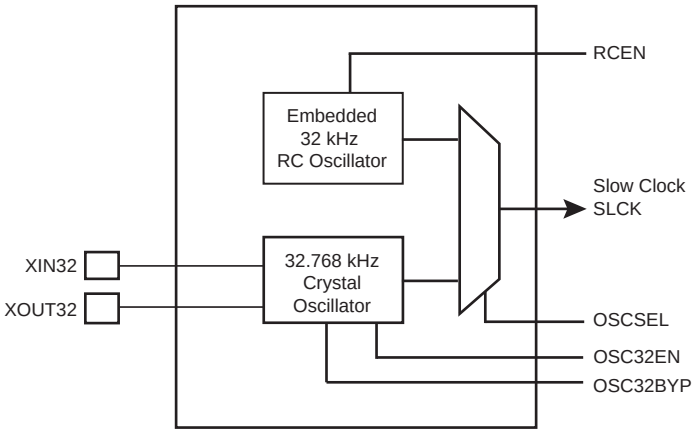
- External 32.768 kHz crystal oscillator
- Embedded 32 kHz (typical) RC oscillator

19.2 Embedded Characteristics

- 32 kHz (typical) RC Oscillator or 32.768 kHz Crystal Oscillator Selector
- VDDBU Powered

19.3 Block Diagram

Figure 19-1: Block Diagram



19.4 Functional Description

The bits RCEN, OSC32EN, OSCSEL, and OSC32BYP are located in the Slow Clock Controller Configuration Register (SCKC_CR) located at the address 0xFFFFFE50 in the backed-up part of the System Controller and, thus, they are preserved while VDDBU is present.

The embedded 32 kHz (typical) RC oscillator and the 32.768 kHz crystal oscillator can be enabled by setting to 1, respectively, the RCEN and OSC32EN bits. The Slow Clock Selector command (OSCSEL bit) selects the slow clock source.

The 32.768 kHz crystal oscillator can be bypassed by setting the OSC32BYP bit to accept an external slow clock on XIN32.

After the VDDBU power-on reset, the default configuration is RCEN = 1, OSC32EN = 0 and OSCSEL = 0, allowing the system to start on the embedded 32 kHz (typical) RC oscillator.

The programmer controls the slow clock switching by software and so must take precautions during the switching phase.

19.4.1 Switching from Embedded 32 kHz RC Oscillator to 32.768 kHz Crystal Oscillator

The sequence to switch from the embedded 32 kHz (typical) RC oscillator to the 32.768 kHz crystal oscillator is the following:

1. Switch the master clock to a source different from slow clock (PLL or Main Oscillator) through the Power Management Controller.
2. Enable the 32.768 kHz crystal oscillator by writing a 1 to the OSC32EN bit.
3. Wait for the 32.768 kHz crystal oscillator to stabilize (software loop).
4. Switch from the embedded 32 kHz (typical) RC oscillator to the 32.768 kHz crystal oscillator by writing a 1 to the OSCSEL bit.
5. Wait 5 slow clock cycles for internal resynchronization.
6. Disable the 32 kHz (typical) RC oscillator by writing a 0 to the RCEN bit.

© 2017 Microchip Technology Inc.

19.4.2 Bypassing the 32.768 kHz Crystal Oscillator

The sequence to bypass the 32.768 kHz crystal oscillator is the following:

1. An external clock must be connected on XIN32.
2. Enable the bypass path by writing a 1 to the OSC32BYP bit.
3. Disable the 32.768 kHz crystal oscillator by writing a 0 to the OSC32EN bit.

19.4.3 Switching from 32.768 kHz Crystal Oscillator to Embedded 32 kHz RC Oscillator

The sequence to switch from the 32.768 kHz crystal oscillator to the embedded 32 kHz (typical) RC oscillator is the following:

1. Switch the master clock to a source different from slow clock (PLL or Main Oscillator).
2. Enable the embedded 32 kHz (typical) RC oscillator for low power by writing a 1 to the RCEN bit.
3. Wait for the embedded 32 kHz (typical) RC oscillator to stabilize (software loop).
4. Switch from the 32.768 kHz crystal oscillator to the embedded RC oscillator by writing a 0 to the OSCSEL bit.
5. Wait 5 slow clock cycles for internal resynchronization.
6. Disable the 32.768 kHz crystal oscillator by writing a 0 to the OSC32EN bit.

© 2017 Microchip Technology Inc.

19.5 Slow Clock Controller (SCKC) User Interface

Table 19-1: Register Mapping

Offset	Register	Name	Access	Reset
0x0	Slow Clock Controller Configuration Register	SCKC_CR	Read/Write	0x0000_0001

© 2017 Microchip Technology Inc.

19.5.1 Slow Clock Controller Configuration Register

Name:SCKC_CR

Address:0xFFFFFE50

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSCSEL	OSC32BYP	OSC32EN	RCEN

RCEN: Embedded 32 kHz (typical) RC Oscillator

0: 32 kHz (typical) RC oscillator is disabled.

1: 32 kHz (typical) RC oscillator is enabled.

OSC32EN: 32.768 kHz Crystal Oscillator

0: 32.768 kHz crystal oscillator is disabled.

1: 32.768 kHz crystal oscillator is enabled.

OSC32BYP: 32.768 kHz Crystal Oscillator Bypass

0: 32.768 kHz crystal oscillator is not bypassed.

1: 32.768 kHz crystal oscillator is bypassed and accepts an external slow clock on XIN32.

OSCSEL: Slow Clock Selector

0 (RC): Slow clock is the embedded 32 kHz (typical) RC oscillator.

1 (XTAL): Slow clock is the 32.768 kHz crystal oscillator.

20. Clock Generator

20.1 Description

The Clock Generator User Interface is embedded within the Power Management Controller and is described in Section 21.13 “Power Management Controller (PMC) User Interface”. However, the Clock Generator registers are named CKGR_.

20.2 Embedded Characteristics

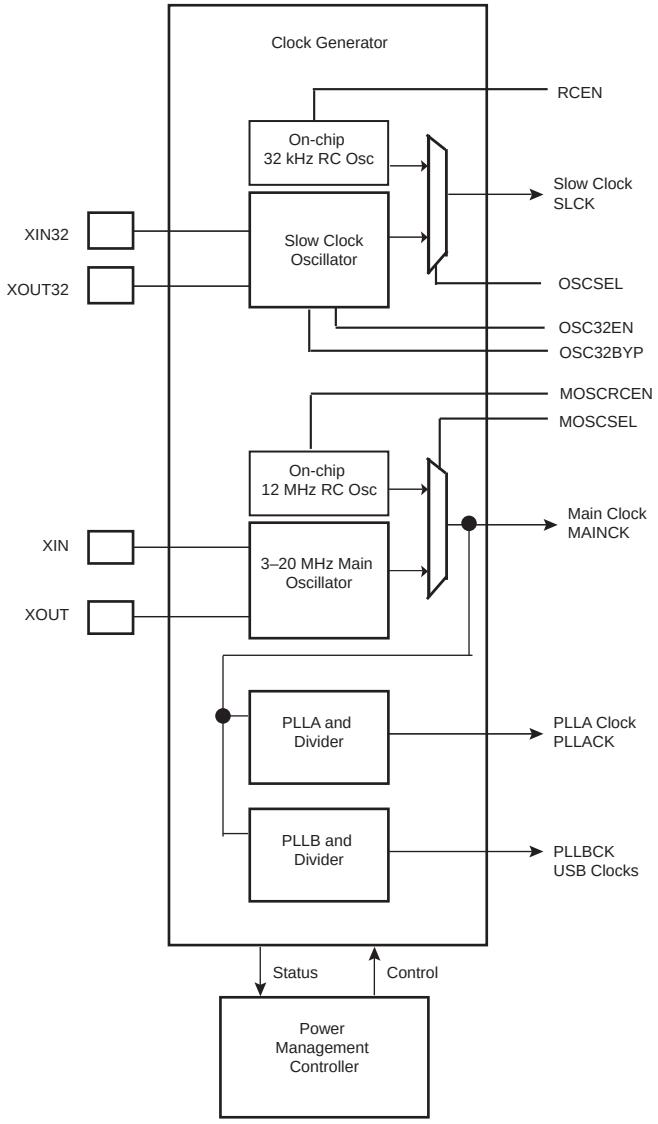
The Clock Generator is made up of:

- Low Power 32768 Hz Slow Clock Oscillator with bypass mode
- Low-Power 32 kHz RC oscillator
- Low-Power 12 MHz RC oscillator
- 3 to 20 MHz Main Oscillator, which can be bypassed
- 400 to 800 MHz programmable PLLA, capable to provide the clock MCK to the processor, and to the peripherals. This PLL has an input divider to offer a wider range of output frequencies from the 16 MHz input, the only limitation being the lowest input frequency shall be higher or equal to 2 MHz.
- 100 MHz programmable PLLB dedicated to USB Full Speed operations. This PLL has an input divider to offer a wider range of output frequencies from the 16 MHz input, the only limitation being the lowest input frequency shall be higher or equal to 2 MHz.

© 2017 Microchip Technology Inc.

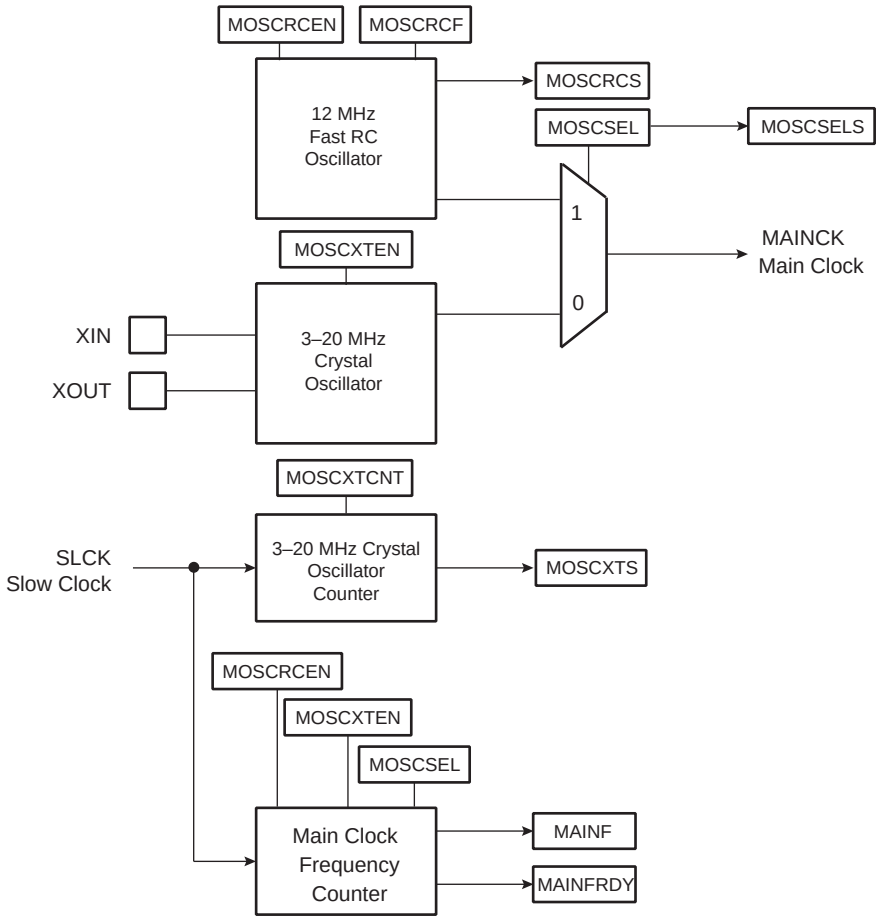
20.3 Block Diagram

Figure 20-1: Clock Generator Block Diagram



20.4 Main Clock

Figure 20-2: Main Clock Block Diagram



The Main Clock has two sources:

- 12 MHz Fast RC Oscillator which starts very quickly and is used at startup
- 3 to 20 MHz Crystal Oscillator (can be bypassed; refer to Section 20.4.4 “Bypass the Main Crystal Oscillator”)

20.4.1 12 MHz Fast RC Oscillator

After reset, the 12 MHz Fast RC Oscillator is enabled and it is selected as the source of MCK. MCK is the default clock selected to start up the system.

Refer to Section 47.2 “DC Characteristics””.

The software can disable or enable the 12 MHz Fast RC Oscillator with the MOSCRCEN bit in the Clock Generator Main Oscillator Register (CKGR_MOR).

When disabling the Main Clock by clearing the MOSCRCEN bit in CKGR_MOR, the MOSCRCS bit in the Power Management Controller Status Register (PMC_SR) is automatically cleared, indicating the Main Clock is off.

Setting the MOSCRCS bit in the Power Management Controller Interrupt Enable Register (PMC_IER) can trigger an interrupt to the processor.

20.4.2 3 to 20 MHz Crystal Oscillator

After reset, the 3 to 20 MHz Crystal Oscillator is disabled and it is not selected as the source of MAINCK.

The user can select the 3 to 20 MHz crystal oscillator to be the source of MAINCK, as it provides a more accurate frequency. The software enables or disables the main oscillator so as to reduce power consumption by clearing the MOSCXTEN bit in the Main Oscillator Register (CKGR_MOR).

© 2017 Microchip Technology Inc.

When disabling the main oscillator by clearing the MOSCXTEN bit in CKGR_MOR, the MOSCXTS bit in PMC_SR is automatically cleared, indicating the Main Clock is off.

When enabling the main oscillator, the user must initiate the main oscillator counter with a value corresponding to the startup time of the oscillator. This startup time depends on the crystal frequency connected to the oscillator.

When the MOSCXTEN bit and the MOSCXTCNT are written in CKGR_MOR to enable the main oscillator, the MOSCXTS bit in the Power Management Controller Status Register (PMC_SR) is cleared and the counter starts counting down on the slow clock divided by 8 from the MOSCXTCNT value. Since the MOSCXTCNT value is coded with 8 bits, the maximum startup time is about 62 ms.

When the counter reaches 0, the MOSCXTS bit is set, indicating that the main clock is valid. Setting the MOSCXTS bit in PMC_IMR can trigger an interrupt to the processor.

20.4.3 Main Clock Oscillator Selection

The user can select either the 12 MHz Fast RC Oscillator or the 3 to 20 MHz Crystal Oscillator to be the source of Main Clock.

The advantage of the 12 MHz Fast RC Oscillator is to have fast startup time, this is why it is selected by default (to start up the system) and when entering in Wait Mode.

The advantage of the 3 to 20 MHz Crystal Oscillator is that it is very accurate.

The selection is made by writing the MOSCSEL bit in the Main Oscillator Register (CKGR_MOR). The switch of the Main Clock source is glitch free, so there is no need to run out of SLCK, PLLACK or UPLLCK in order to change the selection. The MOSCSELS bit of the Power Management Controller Status Register (PMC_SR) allows knowing when the switch sequence is done.

Setting the MOSCSELS bit in PMC_IMR can trigger an interrupt to the processor.

20.4.4 Bypass the Main Crystal Oscillator

Prior to bypassing the 3 to 20 MHz crystal oscillator, the external clock frequency provided on the XIN pin must be stable and within the values specified in Section 47.4.5 “XIN Clock Characteristics”.

The sequence is as follows:

1. An external clock must be connected on XIN.
2. Enable the bypass by writing a 1 to CKGR_MOR.MOSCXTBY.
3. Disable the 3 to 20 MHz oscillator by writing a 0 to bit CKGR_MOR.MOSCXTEN.

20.4.5 Main Clock Frequency Counter

The device features a Main Clock frequency counter that provides the frequency of the Main Clock.

The Main Clock frequency counter is reset and starts incrementing at the Main Clock speed after the next rising edge of the Slow Clock in the following cases:

- when the 12 MHz Fast RC Oscillator clock is selected as the source of Main Clock and when this oscillator becomes stable (i.e., when the MOSCRCS bit is set)
- when the 3 to 20 MHz Crystal Oscillator is selected as the source of Main Clock and when this oscillator becomes stable (i.e., when the MOSCXTS bit is set)
- when the Main Clock Oscillator selection is modified

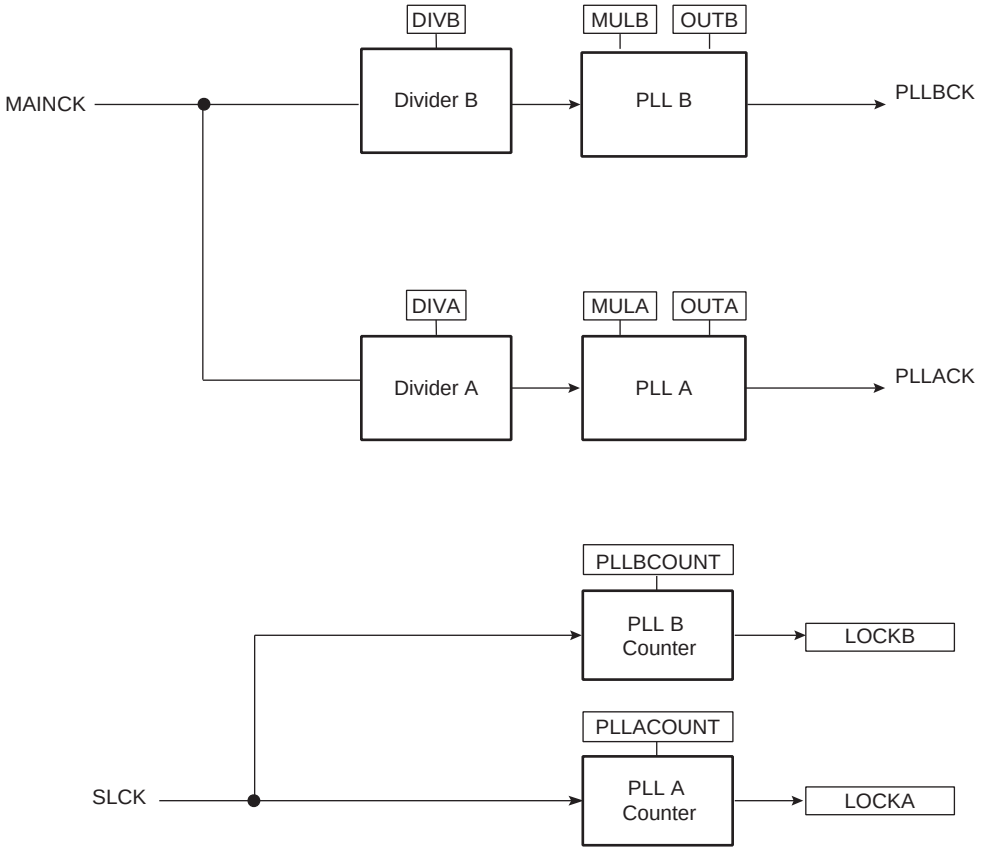
Then, at the 16th falling edge of Slow Clock, the MAINFRDY bit in the Clock Generator Main Clock Frequency Register (CKGR_MCFR) is set and the counter stops counting. Its value can be read in the MAINF field of CKGR_MCFR and gives the number of Main Clock cycles during 16 periods of Slow Clock, so that the frequency of the 12 MHz Fast RC Oscillator or 3 to 20 MHz Crystal Oscillator can be determined.

20.5 Divider and PLL Block

The PLL embeds an input divider to increase the accuracy of the resulting clock signals. However, the user must respect the PLL minimum input frequency when programming the divider.

Figure 20-3 shows the block diagram of the divider and PLL block.

Figure 20-3: Divider and PLL Block Diagram



20.5.1 Divider and Phase Lock Loop Programming

The divider can be set between 1 and 255 in steps of 1. When a divider field (DIV) is cleared, the output of the corresponding divider and the PLL output is a continuous signal at level 0. On reset, each DIV field is cleared, thus the corresponding PLL input clock is stuck at 0.

The PLL allows multiplication of the divider's outputs. The PLL clock signal has a frequency that depends on the respective source signal frequency and on the parameters DIV and MUL. The factor applied to the source signal frequency is (MUL + 1)/DIV. When MUL is written to 0, the corresponding PLL is disabled and its power consumption is saved. Re-enabling the PLL can be performed by writing a value higher than 0 in the MUL field.

Whenever the PLL is re-enabled or one of its parameters is changed, the LOCK bit (LOCKA or LOCKB) in PMC_SR is automatically cleared. The values written in the PLLCOUNT field (PLLACOUNT or PLLBCOUNT) in CKGR_PLLR (CKGR_PLLAR or CKGR_PLLBR), are loaded in the PLL counter. The PLL counter then decrements at the speed of the Slow Clock until it reaches 0. At this time, the LOCK bit is set in PMC_SR and can trigger an interrupt to the processor. The user has to load the number of Slow Clock cycles required to cover the PLL transient time into the PLLCOUNT field.

During the PLLA or PLLB initialization, the PMC_PLLICPR must be programmed correctly.

The PLLA clock can be divided by 2 by writing the PLLADIV2 bit in PMC_MCKR.

© 2017 Microchip Technology Inc.

21. Power Management Controller (PMC)

21.1 Description

The Power Management Controller (PMC) optimizes power consumption by controlling all system and user peripheral clocks. The PMC enables/disables the clock inputs to many of the peripherals and the Core.

21.2 Embedded Characteristics

The Power Management Controller provides all the clock signals to the system.

PMC input clocks:

- PLLACK: From PLLA
- PLLBCK: From PLLB and dedicated to USB clock generation.
- SLCK: slow clock from external 32 kHz oscillator or internal 32 kHz RC
- MAINCK: Main Clock from external 3 to 20 MHz oscillator or internal 12 MHz RC

PMC output clocks:

- Processor Clock PCK
- Master Clock MCK, in particular to the Matrix, the memory interfaces, the peripheral bridge. The divider can be 2, 3 or 4.
- Each peripheral embeds its own divider, programmable in the PMC User Interface.
- 266 MHz DDR system clock

  Note: DDR system clock is not available when Master Clock (MCK) equals Processor Clock (PCK).
- LCD pixel clock that can use DDR system clock or MCK, the choice is done in the LCD user interface
- UHP Clock (UHPCK), required by USB Host Port operations
- UDP Clock (UDPCK), required by USB Device Port operations
- Two programmable clock outputs: PCK0 and PCK1

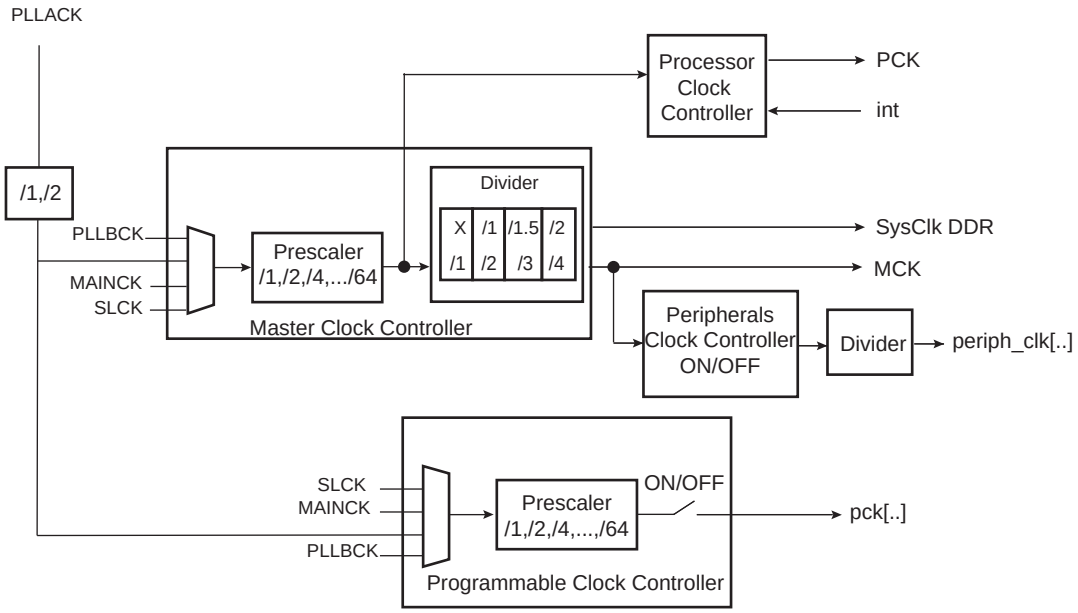
This allows the software control of five flexible operating modes:

- Normal Mode, processor and peripherals running at a programmable frequency
- Idle Mode, processor stopped waiting for an interrupt
- Slow Clock Mode, processor and peripherals running at low frequency
- Standby Mode, mix of Idle and Backup Mode, peripheral running at low frequency, processor stopped waiting for an interrupt
- Backup Mode, Main Power Supplies off, VDDBU powered by a battery

© 2017 Microchip Technology Inc.

21.3 Block Diagram

Figure 21-1: General Clock Block Diagram



21.4 Master Clock Controller

The Master Clock Controller provides selection and division of the Master Clock (MCK). MCK is the clock provided to all the peripherals and the memory controller.

The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLLs.

The Master Clock Controller is made up of a clock selector and a prescaler. It also contains a Master Clock divider which allows the processor clock to be faster than the Master Clock.

The Master Clock selection is made by writing the CSS field (Clock Source Selection) in PMC_MCKR (Master Clock Register). The prescaler supports the division by a power of 2 of the selected clock between 1 and 64. The PRES field in PMC_MCKR programs the prescaler.

Each time PMC_MCKR is written to define a new Master Clock, the MCKRDY bit is cleared in PMC_SR. It reads 0 until the Master Clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one to inform the software when the change is actually done.

21.5 Processor Clock Controller

The PMC features a Processor Clock Controller (PCK) that implements the Processor Idle Mode. The Processor Clock can be disabled by writing the System Clock Disable Register (PMC_SCDR). The status of this clock (at least for debug purposes) can be read in the System Clock Status Register (PMC_SCSR).

The Processor Clock (PCK) is enabled after a reset and is automatically re-enabled by any enabled interrupt. The Processor Idle Mode is achieved by disabling the Processor Clock and entering Wait for Interrupt Mode. The Processor Clock is automatically re-enabled by any enabled fast or normal interrupt, or by the reset of the product. Note: The Arm Wait for Interrupt mode is entered by a CP15 coprocessor operation. Refer to the Microchip application note “Optimizing Power Consumption of AT91SAM9261-based Systems”, literature No. 6217 (available on www.microchip.com). When the Processor Clock is disabled, the current instruction is finished before the clock is stopped, but this does not prevent data transfers from other masters of the system bus.

21.6 USB Device and Host Clocks

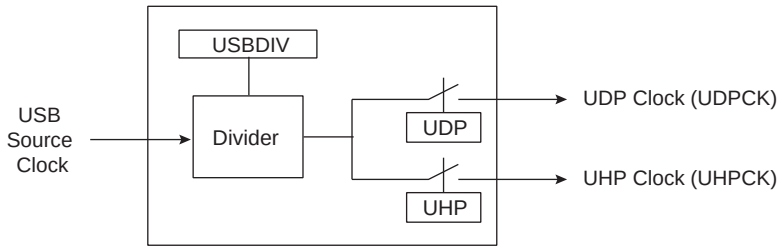
The USB Source Clock is always generated from the PLL B output. If using the USB, the user must program the PLL to generate a 48 MHz, a 96 MHz or a 192 MHz signal with an accuracy of ± 0.25%.

When the PLL B output is stable, i.e., the LOCKB is set:

© 2017 Microchip Technology Inc.

- The USB host clock can be enabled by setting the UHP bit in PMC_SCER. To save power on this peripheral when it is not used, the user can set the UHP bit in PMC_SCDR. The UHP bit in PMC_SCSR gives the activity of this clock. The USB host port require both the 12/48 MHz signal and the Master Clock. The Master Clock may be controlled via the Master Clock Controller.
- The USB device clock can be enabled by setting the UDP bit in PMC_SCER. To save power on this peripheral when it is not used, the user can set the UDP bit in PMC_SCDR. The UDP bit in PMC_SCSR gives the activity of this clock. The USB device port require both the 48 MHz signal and the Master Clock. The Master Clock may be controlled via the Master Clock Controller.

Figure 21-2: USB Clock Controller



21.7 LP-DDR/DDR2 Clock

The Power Management Controller controls the clocks of the DDR memory. It provides SysClk DDR internal clock. That clock is used by the DDR Controller to provide DDR control, data and DDR clock signals.

The DDR clock can be enabled and disabled with DDRCK bit respectively in PMC_SCER and PMC_SDER. At reset DDR clock is disabled to save power consumption.

The Input clock is the same as Master Clock. The Output SysClk DDR Clock is 2 × MCK.

In the case MDIV = ‘00’, (PCK = MCK) SysClk DDR and DDRCK clocks are not available.

If Input clock is PLLACK/PLLADIV2 the DDR Controller can drive DDR2 and LP-DDR at up to 133 MHz with MDIV = ‘11’.

21.8 Peripheral Clock Controller

The Power Management Controller controls the clocks of each embedded peripheral by means of the Peripheral Clock Controller. The user can individually enable and disable the clock on the peripherals and select a division factor from MCK. This is done through the Peripheral Control Register (PMC_PCR).

In order to save power consumption, the division factor can be 1, 2, 4 or 8. The PMC_PCR features a command and acts like a mailbox. To write the division factor on a particular peripheral, the user needs to write a WRITE command, the peripheral ID and the chosen division factor. To read the current division factor on a particular peripheral, the user just needs to write the READ command and the peripheral ID.

Code Example to select divider 8 for peripheral 2 and enable its clock:

```
write_register(PMC_PCR,0x10030102)
```

Code Example to read the divider of peripheral 4:

```
write_register(PMC_PCR,0x00000004)
```

When a peripheral clock is disabled, the clock is immediately stopped. The peripheral clocks are automatically disabled after a reset.

In order to stop a peripheral, it is recommended that the system software wait until the peripheral has executed its last programmed operation before disabling the clock. This is to avoid data corruption or erroneous behavior of the system.

The bit number within the Peripheral Control registers is the Peripheral Identifier defined at the product level. Generally, the bit number corresponds to the interrupt source number assigned to the peripheral.

21.9 Programmable Clock Output Controller

The PMC controls two signals to be output on external pins PCKx. Each signal can be independently programmed via the PMC_PCKx registers.

PCKx can be independently selected between the Slow clock, the Master Clock, the PLLACK/PLLADIV2, the UTMI PLL output and the main clock by writing the CSS field in PMC_PCKx. Each output signal can also be divided by a power of 2 between 1 and 64 by writing the PRES (Prescaler) field in PMC_PCKx.

© 2017 Microchip Technology Inc.

Each output signal can be enabled and disabled by writing a 1 in the corresponding bit, PCKx of PMC_SCER and PMC_SCDR, respectively. Status of the active programmable output clocks are given in the PCKx bits of PMC_SCSR (System Clock Status Register).

Moreover, like the PCK, a status bit in PMC_SR indicates that the Programmable Clock is actually what has been programmed in the Programmable Clock registers.

As the Programmable Clock Controller does not manage with glitch prevention when switching clocks, it is strongly recommended to disable the Programmable Clock before any configuration change and to re-enable it after the change is actually performed.

21.10 Programming Sequence

1. Enabling the 3 to 20 MHz Main Oscillator:

   The main oscillator is enabled by setting the MOSCXTEN field in the CKGR_MOR. In some cases it may be advantageous to define a start-up time. This can be achieved by writing a value in the MOSCXTST field in the CKGR_MOR.

   Once this register has been correctly configured, the user must wait for MOSCXTS field in the PMC_SR to be set. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to MOSCXTS has been enabled in the PMC_IER.

2. Setting PLLA and divider:

   All parameters needed to configure PLLA and the divider are located in the CKGR_PLLAR.

   The DIVA field is used to control the divider itself. A value between 0 and 255 can be programmed. Divider output is divider input divided by DIVA parameter. By default, the DIVA field is cleared, which means that divider A is turned off.

   The OUTA field is used to select the PLLA output frequency range.

   The MULA field is the PLLA multiplier factor. This parameter can be programmed between 0 and 254. If MULA is cleared, PLLA will be turned off, otherwise the PLLA output frequency is PLLA input frequency multiplied by (MULA + 1).

   The PLLACOUNT field specifies the number of slow clock cycles before LOCKA bit is set in the PMC_SR after CKGR_PLLAR has been written.

   Once the PMC_PLLAR has been written, the user must wait for the LOCKA bit to be set in the PMC_SR. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKA has been enabled in the PMC_IER. All parameters in CKGR_PLLAR can be programmed in a single write operation. If at some stage one of the following parameters, MULA, DIVA is modified, LOCKA bit will go low to indicate that PLLA is not ready yet. When PLLA is locked, LOCKA will be set again. The user is constrained to wait for LOCKA bit to be set before using the PLLA output clock.

   Code Example:

   ```
   write_register(CKGR_PLLAR,0x00040805)
   ```

   If PLLA and divider are enabled, the PLLA input clock is the main clock. PLLA output clock is PLLA input clock multiplied by 5. Once CKGR_PLLAR has been written, LOCKA bit will be set after eight slow clock cycles.

3. Setting PLL B and divider B:

   All parameters needed to configure PLL B and divider B are located in the CKGR_PLLBR.

   The DIVB field is used to control divider B itself. A value between 0 and 255 can be programmed. Divider B output is divider B input divided by DIVB parameter. By default, the DIVB field is cleared, which means that divider B is turned off.

   The OUTB field is used to select the PLL B output frequency range.

   The MULB field is the PLL B multiplier factor. This parameter can be programmed between 0 and 2047. If MULB is cleared, PLL B will be turned off, otherwise the PLL B output frequency is PLL B input frequency multiplied by (MULB + 1).

   The PLLBCOUNT field specifies the number of slow clock cycles before LOCKB bit is set in the PMC_SR after CKGR_PLLBR has been written.

   Once the PMC_PLLB register has been written, the user must wait for the LOCKB bit to be set in the PMC_SR. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKB has been enabled in the PMC_IER. All parameters in CKGR_PLLBR can be programmed in a single write operation. If at some stage one of the following parameters, MULB, DIVB is modified, LOCKB bit will go low to indicate that PLL B is not ready yet. When PLL B is locked, LOCKB will be set again. The user is constrained to wait for LOCKB bit to be set before using the PLL A output clock.

   The USBDIV field is used to control the additional divider which generates the USB clock(s).

   Code Example:

   ```
   write_register(CKGR_PLLBR,0x00040805)
   ```


© 2017 Microchip Technology Inc.

If PLL B and divider B are enabled, the PLL B input clock is the main clock. PLL B output clock is PLL B input clock multiplied by 5. Once CKGR_PLLBR has been written, LOCKB bit will be set after eight slow clock cycles.

4. Selection of Master Clock and Processor Clock

   The Master Clock and the Processor Clock are configurable via the PMC_MCKR.

   The CSS field is used to select the clock source of the Master Clock and Processor Clock dividers. By default, the selected clock source is main clock.

   The PRES field is used to control the Master/Processor Clock prescaler. The user can choose between different values (1, 2, 4, 8, 16, 32, 64). Prescaler output is the selected clock source divided by PRES parameter. By default, PRES parameter is set to 1 which means that the input clock of the Master Clock and Processor Clock dividers is equal to slow clock.

   The MDIV field is used to control the Master Clock divider. It is possible to choose between different values (0, 1, 2, 3). The Master Clock output is Master/Processor Clock Prescaler output divided by 1, 2, 4 or 3, depending on the value programmed in MDIV.

   The PLLADIV2 field is used to control the PLLA Clock divider. It is possible to choose between different values (0, 1). The PMC PLLA Clock input is divided by 1 or 2, depending on the value programmed in PLLADIV2.

   By default, MDIV and PLLADIV2 are cleared, which indicates that Processor Clock is equal to the Master Clock.

   Once the PMC_MCKR has been written, the user must wait for the MCKRDY bit to be set in the PMC_SR. This can be done either by polling the status register or by waiting for the interrupt line to be raised if the associated interrupt to MCKRDY has been enabled in the PMC_IER.

   The PMC_MCKR must not be programmed in a single write operation. The preferred programming sequence for the PMC_MCKR is as follows:

- If a new value for CSS field corresponds to PLLA Clock,
  - Program the PRES field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
  - Program the CSS field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
- If a new value for CSS field corresponds to Main Clock or Slow Clock,
  - Program the CSS field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
  - Program the PRES field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.

   If at some stage one of the following parameters, CSS or PRES, is modified, the MCKRDY bit will go low to indicate that the Master Clock and the Processor Clock are not ready yet. The user must wait for MCKRDY bit to be set again before using the Master and Processor Clocks.

   Note: IF PLLA clock was selected as the Master Clock and the user decides to modify it by writing in CKGR_PLLAR, the MCKRDY flag will go low while PLLA is unlocked. Once PLLA is locked again, LOCK goes high and MCKRDY is set.
   While PLLA is unlocked, the Master Clock selection is automatically changed to Main Clock. For further information, see “Clock Switching Waveforms”.

   Code Example:

```
write_register(PMC_MCKR,0x00000001)
        wait (MCKRDY=1)
write_register(PMC_MCKR,0x00000011)
wait (MCKRDY=1)
```

   The Master Clock is main clock divided by 16.

   The Processor Clock is the Master Clock.

5. Selection of Programmable clocks

   Programmable clocks are controlled via registers PMC_SCER, PMC_SCDR and PMC_SCSR.

   Programmable clocks can be enabled and/or disabled via the PMC_SCER and PMC_SCDR. Depending on the system used, 2 programmable clocks can be enabled or disabled. The PMC_SCSR provides a clear indication as to which Programmable clock is enabled. By default all Programmable clocks are disabled.

   PMC_PCKx registers are used to configure programmable clocks.

© 2017 Microchip Technology Inc.

The CSS and CSSMCK fields are used to select the programmable clock divider source. Five clock options are available: main clock, slow clock, master clock, PLLACK, UPLLCK. By default, the clock source selected is slow clock.

The PRES field is used to control the programmable clock prescaler. It is possible to choose between different values (1, 2, 4, 8, 16, 32, 64). Programmable clock output is prescaler input divided by PRES parameter. By default, the PRES parameter is set to 1 which means that master clock is equal to slow clock.

Once the PMC_PCKx register has been programmed, The corresponding programmable clock must be enabled and the user is constrained to wait for the PCKRDYx bit to be set in the PMC_SR. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to PCKRDYx has been enabled in the PMC_IER. All parameters in PMC_PCKx can be programmed in a single write operation.

If the CSS and PRES parameters are to be modified, the corresponding programmable clock must be disabled first. The parameters can then be modified. Once this has been done, the user must re-enable the programmable clock and wait for the PCKRDYx bit to be set.

Code Example:

```
write_register(PMC_PCK0,0x00000015)
```

Programmable clock 0 is main clock divided by 32.

6. Enabling Peripheral Clocks

Once all of the previous steps have been completed, the peripheral clocks can be enabled and/or disabled via registers PMC_PCER and PMC_PCDR.

Depending on the system used, 19 peripheral clocks can be enabled or disabled. The PMC_PCR provides a clear view as to which peripheral clock is enabled.

Note: Each enabled peripheral clock corresponds to Master Clock.

Code Examples:

```
write_register(PMC_PCER,0x00000110)
```

Peripheral clocks 4 and 8 are enabled.

```
write_register(PMC_PCDR,0x00000010)
```

Peripheral clock 4 is disabled.

21.11 Clock Switching Details

21.11.1 Master Clock Switching Timings

Table 21-1 and Table 21-2 give the worst case timings required for the Master Clock to switch from one selected clock to another one. This is in the event that the prescaler is de-activated. When the prescaler is activated, an additional time of 64 clock cycles of the new selected clock has to be added.

Table 21-1: Clock Switching Timings (Worst Case)

	From		
To	**Main Clock**	**SLCK**	**PLL Clock**
Main Clock	–	4 x SLCK + 2.5 x Main Clock	3 x PLL Clock + 4 x SLCK + 1 x Main Clock
SLCK	0.5 x Main Clock + 4.5 x SLCK	–	3 x PLL Clock + 5 x SLCK
PLL Clock	0.5 x Main Clock + 4 x SLCK + PLLCOUNT x SLCK + 2.5 x PLLx Clock	2.5 x PLL Clock + 5 x SLCK + PLLCOUNT x SLCK	2.5 x PLL Clock + 4 x SLCK + PLLCOUNT x SLCK

Note 1: PLL designates either the PLLA or the UPLL Clock.

2: PLLCOUNT designates either PLLACOUNT or UPLLCOUNT.

© 2017 Microchip Technology Inc.

Table 21-2: Clock Switching Timings between Two PLLs (Worst Case)

To	From	
	PLLA Clock	**PLLB Clock**
PLLA Clock	2.5 x PLLA Clock + 4 x SLCK + PLLACOUNT x SLCK	3 x PLLA Clock + 4 x SLCK + 1.5 x PLLA Clock
PLLB Clock	3 x PLLB Clock + 4 x SLCK + 1.5 x PLLB Clock	2.5 x PLLB Clock + 4 x SLCK + PLLBCOUNT x SLCK

21.11.2 Clock Switching Waveforms

Figure 21-3: Switch Master Clock from Slow Clock to PLL Clock

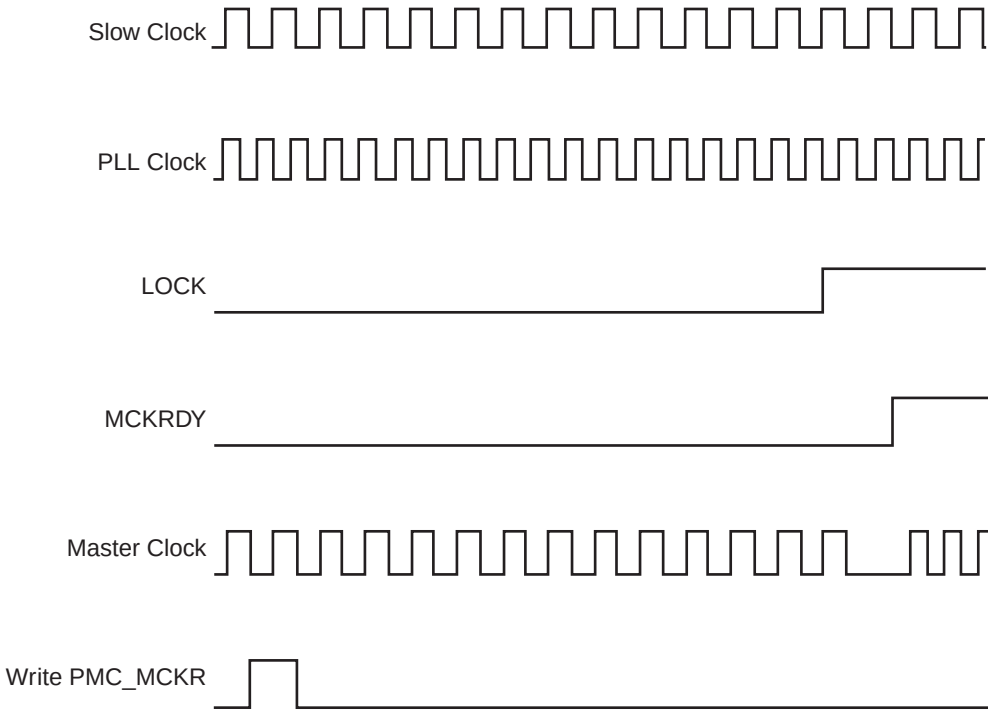


Figure 21-4: Switch Master Clock from Main Clock to Slow Clock

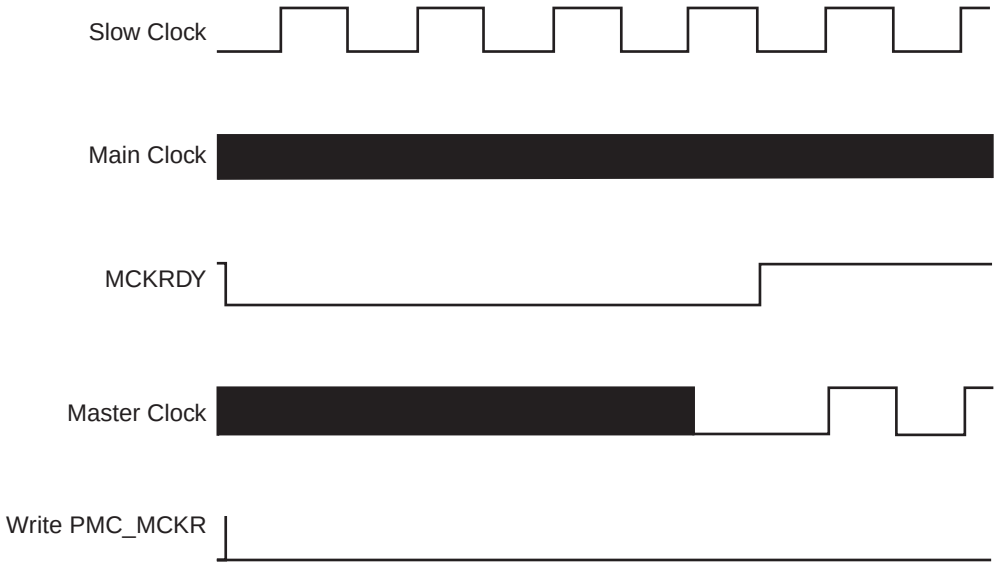
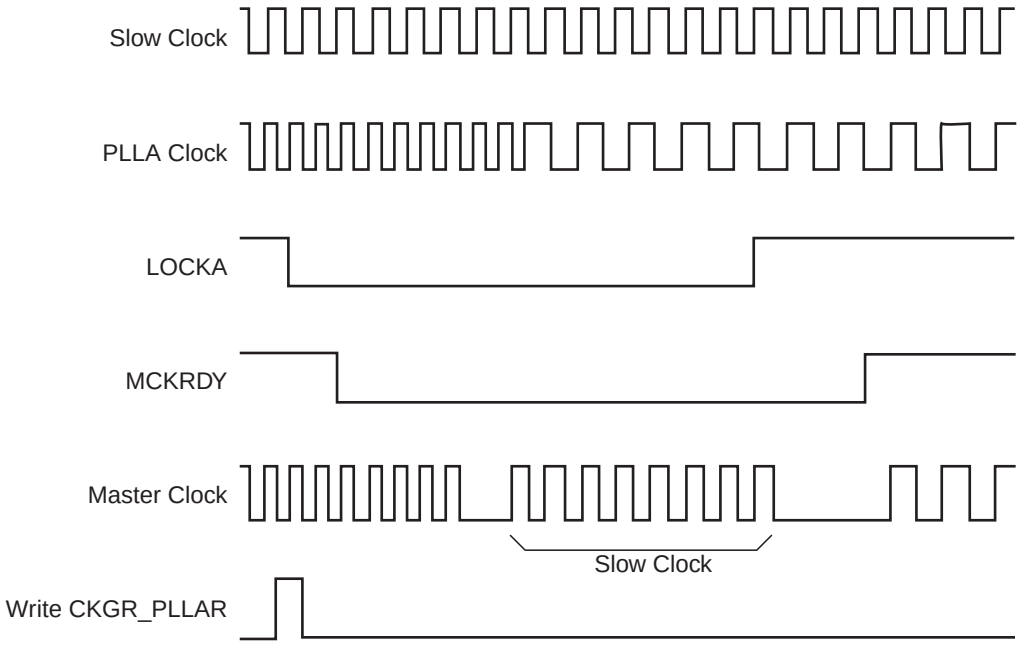
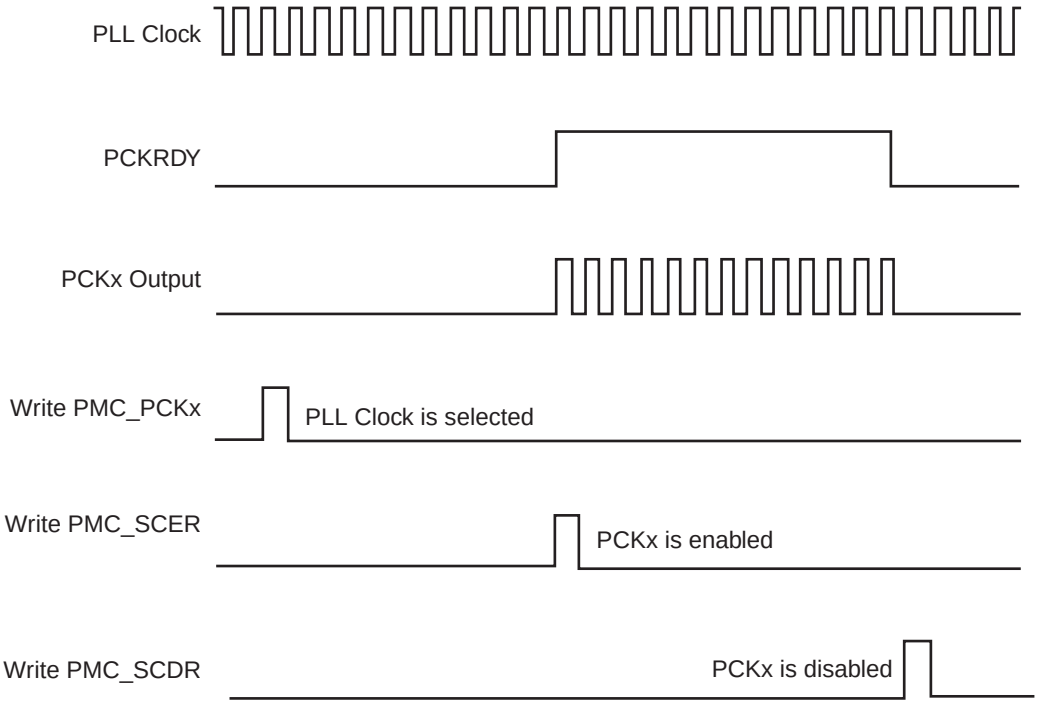


Figure 21-5: Change PLLA Programming



© 2017 Microchip Technology Inc.

Figure 21-6: Programmable Clock Output Programming



21.12 Register Write Protection

To prevent any single software error from corrupting PMC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the PMC Write Protection Mode Register (PMC_WPMR).

If a write access to a write-protected register is detected, the WPVS bit in the PMC Write Protection Status Register (PMC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the PMC_WPSR.

The following registers can be write-protected:

- PMC System Clock Enable Register
- PMC System Clock Disable Register
- PMC Clock Generator Main Clock Frequency Register
- PMC Clock Generator PLLA Register
- PMC Master Clock Register
- PMC Programmable Clock Register
- PLL Charge Pump Current Register

21.13 Power Management Controller (PMC) User Interface

Table 21-3: Register Mapping

Offset	Register	Name	Access	Reset
0x0000	System Clock Enable Register	PMC_SCER	Write-only	–
0x0004	System Clock Disable Register	PMC_SCDR	Write-only	–
0x0008	System Clock Status Register	PMC_SCSR	Read-only	0x0000_0005
0x0010	Peripheral Clock Enable Register	PMC _PCER	Write-only	–
0x0014	Peripheral Clock Disable Register	PMC_PCDR	Write-only	–
0x0018	Peripheral Clock Status Register	PMC_PCSR	Read-only	0x0000_0000
0x000C–0x0018	Reserved	–	–	–
0x001C	Reserved	–	–	–
0x0020	Main Oscillator Register	CKGR_MOR	Read/Write	0x0100_0001
0x0024	Main Clock Frequency Register	CKGR_MCFR	Read/Write	0x0000_0000
0x0028	PLLA Register	CKGR_PLLAR	Read/Write	0x0000_3F00
0x002C	PLLB Register	CKGR_PLLBR	Read/Write	0x0000_3F00
0x0030	Master Clock Register	PMC_MCKR	Read/Write	0x0000_0001
0x0034	Reserved	–	–	–
0x0038	USB Clock Register	PMC_USB	Read/Write	0x0000_0000
0x003C	Reserved	–	–	–
0x0040	Programmable Clock 0 Register	PMC_PCK0	Read/Write	0x0000_0000
0x0044	Programmable Clock 1 Register	PMC_PCK1	Read/Write	0x0000_0000
0x0048–0x005C	Reserved	–	–	–
0x0060	Interrupt Enable Register	PMC_IER	Write-only	–
0x0064	Interrupt Disable Register	PMC_IDR	Write-only	–
0x0068	Status Register	PMC_SR	Read-only	0x0001_0008
0x006C	Interrupt Mask Register	PMC_IMR	Read-only	0x0000_0000
0x0070–0x0078	Reserved	–	–	–
0x0080	PLL Charge Pump Current Register	PMC_PLLICPR	Read/Write	0x0100_0100
0x0084–0x00E0	Reserved	–	–	–
0x00E4	Write Protection Mode Register	PMC_WPMR	Read/Write	0x0000_0000
0x00E8	Write Protection Status Register	PMC_WPSR	Read-only	0x0000_0000
0x00EC–0x0108	Reserved	–	–	–
0x010C	Peripheral Control Register	PMC_PCR	Read/Write	0x0000_0000
0x007C–0x00FC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

21.13.1 PMC System Clock Enable Register

Name:PMC_SCER

Address:0xFFFFFC00

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	–	LCDCK	DDRCK	–	–

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

DDRCK: DDR Clock Enable

0: No effect.

1: Enables the DDR clock.

LCDCK: LCD Clock Enable

0: No effect.

1: Enables the LCD clock.

UHP: USB Host OHCI Clocks Enable

0: No effect.

1: Enables the UHP48M and UHP12M OHCI clocks.

UDP: USB Device Clock Enable

0: No effect.

1: Enables the USB Device clock.

PCKx: Programmable Clock x Output Enable

0: No effect.

1: Enables the corresponding Programmable Clock output.

21.13.2 PMC System Clock Disable Register

Name:PMC_SCDR

Address:0xFFFFFC04

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	–	LCDCK	DDRCK	–	PCK

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

PCK: Processor Clock Disable

0: No effect.

1: Disables the Processor clock. This is used to enter the processor in Idle Mode.

DDRCK: DDR Clock Disable

0: No effect.

1: Disables the DDR clock.

LCDCK: LCD Clock Disable

0: No effect.

1: Disables the LCD clock.

UHP: USB Host OHCI Clock Disable

0: No effect.

1: Disables the UHP48M and UHP12M OHCI clocks.

UDP: USB Device Clock Enable

0: No effect.

1: Disables the USB Device clock.

PCKx: Programmable Clock x Output Disable

0: No effect.

1: Disables the corresponding Programmable Clock output.

© 2017 Microchip Technology Inc.

21.13.3 PMC System Clock Status Register

Name:PMC_SCSR

Address:0xFFFFFC08

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	–	LCDCK	DDRCK	–	PCK

PCK: Processor Clock Status

0: The Processor clock is disabled.

1: The Processor clock is enabled.

DDRCK: DDR Clock Status

0: The DDR clock is disabled.

1: The DDR clock is enabled.

LCDCK: LCD Clock Status

0: The LCD clock is disabled.

1: The LCD clock is enabled.

UHP: USB Host Port Clock Status

0: The UHP48M and UHP12M OHCI clocks are disabled.

1: The UHP48M and UHP12M OHCI clocks are enabled.

UDP: USB Device Port Clock Status

0: The USB Device clock is disabled.

1: The USB Device clock is enabled.

PCKx: Programmable Clock x Output Status

0: The corresponding Programmable Clock output is disabled.

1: The corresponding Programmable Clock output is enabled.

21.13.4 PMC Peripheral Clock Enable Register

Name:PMC_PCER

Address:0xFFFFFC10

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

PIDx: Peripheral Clock x Enable

0: No effect.

1: Enables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in Section 7.2 “Peripheral Identifiers”.

Note: Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

© 2017 Microchip Technology Inc.

21.13.5 PMC Peripheral Clock Disable Register

Name:PMC_PCDR

Address:0xFFFFFC14

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

PIDx: Peripheral Clock x Disable

0: No effect.

1: Disables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in Section 7.2 “Peripheral Identifiers”.

21.13.6 PMC Peripheral Clock Status Register

Name:PMC_PCSR

Address:0xFFFFFC18

Access: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

PIDx: Peripheral Clock x Status

0: The corresponding peripheral clock is disabled.

1: The corresponding peripheral clock is enabled.

Note: PID2 to PID31 refer to identifiers as defined in Section 7.2 “Peripheral Identifiers”.

© 2017 Microchip Technology Inc.

21.13.7 PMC Clock Generator Main Oscillator Register

Name:CKGR_MOR

Address:0xFFFFFC20

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	CFDEN	MOSCSEL

23	22	21	20	19	18	17	16
KEY							

15	14	13	12	11	10	9	8
MOSCXTST							

7	6	5	4	3	2	1	0
–	–	–	–	MOSCRCEN	–	MOSCXTBY	MOSCXTEN

KEY: Password

Should be written at value 0x37. Writing any other value in this field aborts the write operation.

MOSCXTEN: Main Crystal Oscillator Enable

A crystal must be connected between XIN and XOUT.

0: The Main Crystal Oscillator is disabled.

1: The Main Crystal Oscillator is enabled. MOSCXTBY must be cleared.

When MOSCXTEN is set, the MOSCXTS flag is set once the Main Crystal Oscillator startup time is achieved.

MOSCXTBY: Main Crystal Oscillator Bypass

0: No effect.

1: The Main Crystal Oscillator is bypassed. MOSCXTEN must be cleared. An external clock must be connected on XIN.

When MOSCXTBY is set, the MOSCXTS flag in PMC_SR is automatically set.

Clearing MOSCXTEN and MOSCXTBY bits allows resetting the MOSCXTS flag.

MOSCRCEN: Main On-chip RC Oscillator Enable

0: The Main On-chip RC Oscillator is disabled.

1: The Main On-chip RC Oscillator is enabled.

When MOSCRCEN is set, the MOSCRCS flag is set once the Main On-chip RC Oscillator startup time is achieved.

MOSCXTST: Main Crystal Oscillator Start-up Time

Specifies the number of Slow Clock cycles multiplied by 8 for the Main Crystal Oscillator start-up time.

MOSCSEL: Main Oscillator Selection

0: The Main On-chip RC Oscillator is selected.

1: The Main Crystal Oscillator is selected.

CFDEN: Clock Failure Detector Enable

0: The Clock Failure Detector is disabled.

1: The Clock Failure Detector is enabled.

© 2017 Microchip Technology Inc.

21.13.8 PMC Clock Generator Main Clock Frequency Register

Name:CKGR_MCFR

Address:0xFFFFFC24

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	RCMEAS	–	–	–	MAINFRDY

15	14	13	12	11	10	9	8
MAINF							

7	6	5	4	3	2	1	0
MAINF							

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

MAINF: Main Clock Frequency

Gives the number of Main Clock cycles within 16 Slow Clock periods.

MAINFRDY: Main Clock Frequency Ready

0: MAINF value is not valid or the Main Oscillator is disabled.

1: The Main Oscillator has been enabled previously and MAINF value is available.

RCMEAS: RC Measure (write-only)

0: No effect.

1: Restart a measure of the main RC frequency. MAINF will carry the new frequency as soon as a low to high transition occurs on MAINFRDY flag.

© 2017 Microchip Technology Inc.

21.13.9 PMC Clock Generator PLLA Register

Name:CKGR_PLLAR

Address:0xFFFFFC28

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	ONE	–	–	MULA		

23	22	21	20	19	18	17	16
MULA							

15	14	13	12	11	10	9	8
OUTA		PLLACOUNT					

7	6	5	4	3	2	1	0
DIVA							

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

Possible limitations on PLLA input frequencies and multiplier factors should be checked before using the PMC.

Warning: Bit 29 must always be set to 1 when programming the CKGR_PLLAR.

DIVA: Divider A

0: Divider output is 0

1: Divider is bypassed

2–255: Divider output is the selected clock divided by DIVA

PLLACOUNT: PLLA Counter

Specifies the number of slow clock cycles before the LOCKA bit is set in PMC_SR after CKGR_PLLAR is written.

OUTA: PLLA Clock Frequency Range

To optimize clock performance, this field must be programmed as specified in Section 47.8 “PLL Characteristics”.

MULA: PLLA Multiplier

0: The PLLA is deactivated.

1–254: The PLLA Clock frequency is the PLLA input frequency multiplied by MULA + 1.

ONE: Must Be Set to 1

Bit 29 must always be set to 1 when programming the CKGR_PLLAR.

© 2017 Microchip Technology Inc.

21.13.10 PMC Clock Generator PLLB Register

Name:CKGR_PLLBR

Address:0xFFFFFC2C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	MULB		

23	22	21	20	19	18	17	16
MULB							

15	14	13	12	11	10	9	8
OUTB		PLLBCOUNT					

7	6	5	4	3	2	1	0
DIVB							

Possible limitations on PLLB input frequencies and multiplier factors should be checked before using the PMC.

DIVB: Divider B

0: Divider output is 0

1: Divider is bypassed

2–255: Divider output is the selected clock divided by DIVB

PLLBCOUNT: PLLB Counter

Specifies the number of slow clock cycles before the LOCKB bit is set in PMC_SR after CKGR_PLLBR is written.

OUTB: PLLB Clock Frequency Range

To optimize clock performance, this field must be programmed as specified in Section 47.8 “PLL Characteristics”.

MULB: PLLB Multiplier

0: The PLLB is deactivated.

1–254: The PLLB Clock frequency is the PLLB input frequency multiplied by MULB+1.

© 2017 Microchip Technology Inc.

21.13.11 PMC Master Clock Register

Name:PMC_MCKR

Address:0xFFFFFC30

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	PLLADIV2	–	–	MDIV	
7	6	5	4	3	2	1	0
–	PRES			–	–	CSS	

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

CSS: Master/Processor Clock Source Selection

Value	Name	Description
0	SLOW_CLK	Slow Clock is selected
1	MAIN_CLK	Main Clock is selected
2	PLLA_CLK	PLLACK/PLLADIV2 is selected
3	PLLB_CLK	PLLBCK is selected

PRES: Master/Processor Clock Prescaler

Value	Name	Description
0	CLOCK_DIV1	Selected clock
1	CLOCK_DIV2	Selected clock divided by 2
2	CLOCK_DIV4	Selected clock divided by 4
3	CLOCK_DIV8	Selected clock divided by 8
4	CLOCK_DIV16	Selected clock divided by 16
5	CLOCK_DIV32	Selected clock divided by 32
6	CLOCK_DIV64	Selected clock divided by 64
7	–	Reserved

MDIV: Master Clock Division

Value	Name	Description
0	EQ_PCK	Master Clock is Prescaler Output Clock divided by 1. **Warning**: SysClk DDR and DDRCK are not available.
1	PCK_DIV2	Master Clock is Prescaler Output Clock divided by 2. SysClk DDR is equal to 2 × MCK. DDRCK is equal to MCK.
2	PCK_DIV4	Master Clock is Prescaler Output Clock divided by 4. SysClk DDR is equal to 2 × MCK. DDRCK is equal to MCK.

© 2017 Microchip Technology Inc.

Value	Name	Description
3	PCK_DIV3	Master Clock is Prescaler Output Clock divided by 3. SysClk DDR is equal to 2 × MCK. DDRCK is equal to MCK.

PLLADIV2: PLLA divisor by 2

Value	Name	Description
0	NOT_DIV2	PLLA clock frequency is divided by 1.
1	DIV2	PLLA clock frequency is divided by 2.

© 2017 Microchip Technology Inc.

21.13.12 USB Clock Register

Name:PMC_USB

Address:0xFFFFFC38

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	USBDIV			
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	USBS

USBS: USB OHCI Input Clock Selection

0: USB Clock disabled.

1: USB Clock Input is PLLB.

USBDIV: Divider for USB Clock

USB Clock is Input Clock divided by USBDIV + 1.

21.13.13 PMC Programmable Clock Register

Name:PMC_PCKx

Address:0xFFFFFC40

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	PRES			–	CSS		

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

CSS: Master Clock Source Selection

Value	Name	Description
0	SLOW_CLK	Slow Clock is selected
1	MAIN_CLK	Main Clock is selected
2	PLLA_CLK	PLLACK/PLLADIV2 is selected
3	PLLB_CLK	PLLBCK is selected
4	MCK_CLK	Master Clock is selected

PRES: Programmable Clock Prescaler

Value	Name	Description
0	CLOCK_DIV1	Selected clock
1	CLOCK_DIV2	Selected clock divided by 2
2	CLOCK_DIV4	Selected clock divided by 4
3	CLOCK_DIV8	Selected clock divided by 8
4	CLOCK_DIV16	Selected clock divided by 16
5	CLOCK_DIV32	Selected clock divided by 32
6	CLOCK_DIV64	Selected clock divided by 64
7	–	Reserved

© 2017 Microchip Technology Inc.

21.13.14 PMC Interrupt Enable Register

Name:PMC_IER

Address:0xFFFFFC60

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCXTS

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt

MOSCXTS: Main Crystal Oscillator Status Interrupt Enable

LOCKA: PLLA Lock Interrupt Enable

LOCKB: PLLB Lock Interrupt Enable

MCKRDY: Master Clock Ready Interrupt Enable

PCKRDYx: Programmable Clock Ready x Interrupt Enable

MOSCSELS: Main Oscillator Selection Status Interrupt Enable

MOSCRCS: Main On-chip RC Status Interrupt Enable

CFDEV: Clock Failure Detector Event Interrupt Enable

21.13.15 PMC Interrupt Disable Register

Name:PMC_IDR

Address:0xFFFFFC64

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCXTS

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt

MOSCXTS: Main Crystal Oscillator Status Interrupt Disable

LOCKA: PLLA Lock Interrupt Disable

LOCKB: PLLB Lock Interrupt Disable

MCKRDY: Master Clock Ready Interrupt Disable

PCKRDYx: Programmable Clock Ready x Interrupt Disable

MOSCSELS: Main Oscillator Selection Status Interrupt Disable

MOSCRCS: Main On-chip RC Status Interrupt Disable

CFDEV: Clock Failure Detector Event Interrupt Disable

© 2017 Microchip Technology Inc.

21.13.16 PMC Status Register

Name:PMC_SR

Address:0xFFFFFC68

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	FOS	CFDS	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
OSCSELS	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCXTS

MOSCXTS: Main XTAL Oscillator Status

0: Main XTAL oscillator is not stabilized.

1: Main XTAL oscillator is stabilized.

LOCKA: PLLA Lock Status

0: PLLA is not locked

1: PLLA is locked.

LOCKB: PLLB Lock Status

0: PLLB is not locked

1: PLLB is locked.

MCKRDY: Master Clock Status

0: Master Clock is not ready.

1: Master Clock is ready.

OSCSELS: Slow Clock Oscillator Selection

0: Internal slow clock RC oscillator is selected.

1: External slow clock 32 kHz oscillator is selected.

PCKRDYx: Programmable Clock Ready Status

0: Programmable Clock x is not ready.

1: Programmable Clock x is ready.

MOSCSELS: Main Oscillator Selection Status

0: Selection is in progress.

1: Selection is done.

MOSCRCS: Main On-chip RC Oscillator Status

0: Main on-chip RC oscillator is not stabilized.

1: Main on-chip RC oscillator is stabilized.

CFDEV: Clock Failure Detector Event

0: No clock failure detection of the main on-chip RC oscillator clock has occurred since the last read of PMC_SR.

1: At least one clock failure detection of the main on-chip RC oscillator clock has occurred since the last read of PMC_SR.

© 2017 Microchip Technology Inc.

CFDS: Clock Failure Detector Status

0: A clock failure of the main on-chip RC oscillator clock is not detected.

1: A clock failure of the main on-chip RC oscillator clock is detected.

FOS: Clock Failure Detector Fault Output Status

0: The fault output of the clock failure detector is inactive.

1: The fault output of the clock failure detector is active.

© 2017 Microchip Technology Inc.

21.13.17 PMC Interrupt Mask Register

Name:PMC_IMR

Address:0xFFFFFC6C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCXTS

The following configuration values are valid for all listed bit names of this register:

0: Corresponding interrupt is not enabled.

1: Corresponding interrupt is enabled.

MOSCXTS: Main Crystal Oscillator Status Interrupt Mask

LOCKA: PLLA Lock Interrupt Mask

LOCKB: PLLB Lock Interrupt Mask

MCKRDY: Master Clock Ready Interrupt Mask

PCKRDYx: Programmable Clock Ready x Interrupt Mask

MOSCSELS: Main Oscillator Selection Status Interrupt Mask

MOSCRCS: Main On-chip RC Status Interrupt Mask

CFDEV: Clock Failure Detector Event Interrupt Mask

21.13.18 PLL Charge Pump Current Register

Name:PMC_PLLICPR

Address:0xFFFFFC80

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	ICPLLA

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

ICPLLA: Charge Pump Current

To optimize clock performance, this bit must be programmed as specified in the “PLL Characteristics” in the product “Electrical Characteristics” section.

© 2017 Microchip Technology Inc.

21.13.19 PMC Write Protection Mode Register

Name:PMC_WPMR

Address:0xFFFFFCE4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

WPEN: Write Protection Enable

0: Disables the write protection if WPKEY corresponds to 0x504D43 (“PMC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x504D43 (“PMC” in ASCII).

See Section 21.12 “Register Write Protection” for list of registers that can be write-protected.

WPKEY: Write Protection Key

Value	Name	Description
0x504D43	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

21.13.20 PMC Write Protection Status Register

Name:PMC_WPSR

Address:0xFFFFFCE8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

WPVS: Write Protect Violation Status

0: No write protection violation has occurred since the last read of the PMC_WPSR.

1: A write protection violation has occurred since the last read of the PMC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

WPVSRC: Write Protect Violation Source

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted..

© 2017 Microchip Technology Inc.

21.13.21 PMC Peripheral Control Register

Name:PMC_PCR

Address:0xFFFFFD0C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	EN	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	DIV	

15	14	13	12	11	10	9	8
–	–	–	CMD	–	–	–	–

7	6	5	4	3	2	1	0
–	–	PID					

PID: Peripheral ID

Peripheral ID selection from PID2 to PID31

PID2 to PID31 refer to identifiers as defined in Section 7.2 “Peripheral Identifiers”.

CMD: Command

0: Read mode

1: Write mode

DIV: Divisor value

Value	Name	Description
0	PERIPH_DIV_MCK	Peripheral clock is MCK
1	PERIPH_DIV2_MCK	Peripheral clock is MCK/2
2	PERIPH_DIV4_MCK	Peripheral clock is MCK/4
3	PERIPH_DIV8_MCK	Peripheral clock is MCK/8

EN: Enable

0: Selected Peripheral clock is disabled

1: Selected Peripheral clock is enabled

22. Parallel Input/Output Controller (PIO)

22.1 Description

The Parallel Input/Output Controller (PIO) manages up to 32 fully programmable input/output lines. Each I/O line may be dedicated as a general-purpose I/O or be assigned to a function of an embedded peripheral. This ensures effective optimization of the pins of the product.

Each I/O line is associated with a bit number in all of the 32-bit registers of the 32-bit wide user interface.

Each I/O line of the PIO Controller features:

- An input change interrupt enabling level change detection on any I/O line.
- Additional Interrupt modes enabling rising edge, falling edge, low-level or high-level detection on any I/O line.
- A glitch filter providing rejection of glitches lower than one-half of peripheral clock cycle.
- A debouncing filter providing rejection of unwanted pulses from key or push button operations.
- Multi-drive capability similar to an open drain I/O line.
- Control of the pull-up and pull-down of the I/O line.
- Input visibility and output control.

The PIO Controller also features a synchronous output providing up to 32 bits of data output in a single write operation.

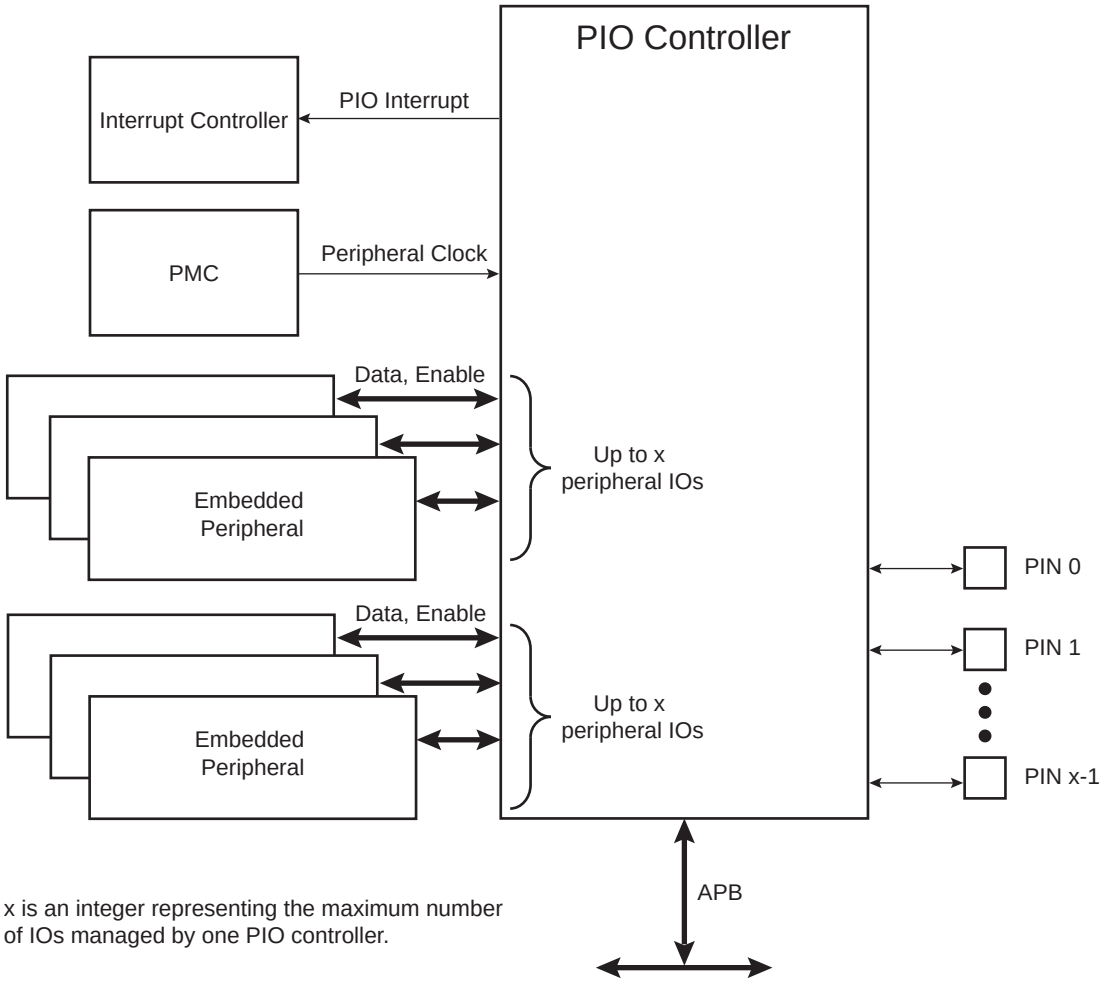
22.2 Embedded Characteristics

- Up to 32 Programmable I/O Lines
- Fully Programmable through Set/Clear Registers
- Multiplexing of Four Peripheral Functions per I/O Line
- For each I/O Line (Whether Assigned to a Peripheral or Used as General Purpose I/O)
  - Input Change Interrupt
  - Programmable Glitch Filter
  - Programmable Debouncing Filter
  - Multi-drive Option Enables Driving in Open Drain
  - Programmable Pull-Up on Each I/O Line
  - Pin Data Status Register, Supplies Visibility of the Level on the Pin at Any Time
  - Additional Interrupt Modes on a Programmable Event: Rising Edge, Falling Edge, Low-Level or High-Level
- Synchronous Output, Provides Set and Clear of Several I/O Lines in a Single Write
- Register Write Protection
- Programmable Schmitt Trigger Inputs
- Programmable I/O Delay
- Programmable I/O Drive

© 2017 Microchip Technology Inc.

22.3 Block Diagram

Figure 22-1: Block Diagram



22.4 Product Dependencies

22.4.1 Pin Multiplexing

Each pin is configurable, depending on the product, as either a general-purpose I/O line only, or as an I/O line multiplexed with one or two peripheral I/Os. As the multiplexing is hardware defined and thus product-dependent, the hardware designer and programmer must carefully determine the configuration of the PIO Controllers required by their application. When an I/O line is general-purpose only, i.e., not multiplexed with any peripheral I/O, programming of the PIO Controller regarding the assignment to a peripheral has no effect and only the PIO Controller can control how the pin is driven by the product.

22.4.2 External Interrupt Lines

The interrupt signals FIQ and IRQ0 to IRQn are generally multiplexed through the PIO Controllers. However, it is not necessary to assign the I/O line to the interrupt function as the PIO Controller has no effect on inputs and the external interrupt lines are used only as inputs.

22.4.3 Power Management

The Power Management Controller controls the peripheral clock in order to save power. Writing any of the registers of the user interface does not require the peripheral clock to be enabled. This means that the configuration of the I/O lines does not require the peripheral clock to be enabled.

However, when the clock is disabled, not all of the features of the PIO Controller are available, including glitch filtering. Note that the input change interrupt, the interrupt modes on a programmable event and the read of the pin level require the clock to be validated.

After a hardware reset, the peripheral clock is disabled by default.

The user must configure the Power Management Controller before any access to the input line information.

22.4.4 Interrupt Sources

For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources. Refer to the PIO Controller peripheral identifier in the Peripheral Identifiers table to identify the interrupt sources dedicated to the PIO Controllers. Using the PIO Controller requires the Interrupt Controller to be programmed first.

The PIO Controller interrupt can be generated only if the peripheral clock is enabled.

Table 22-1: Peripheral IDs

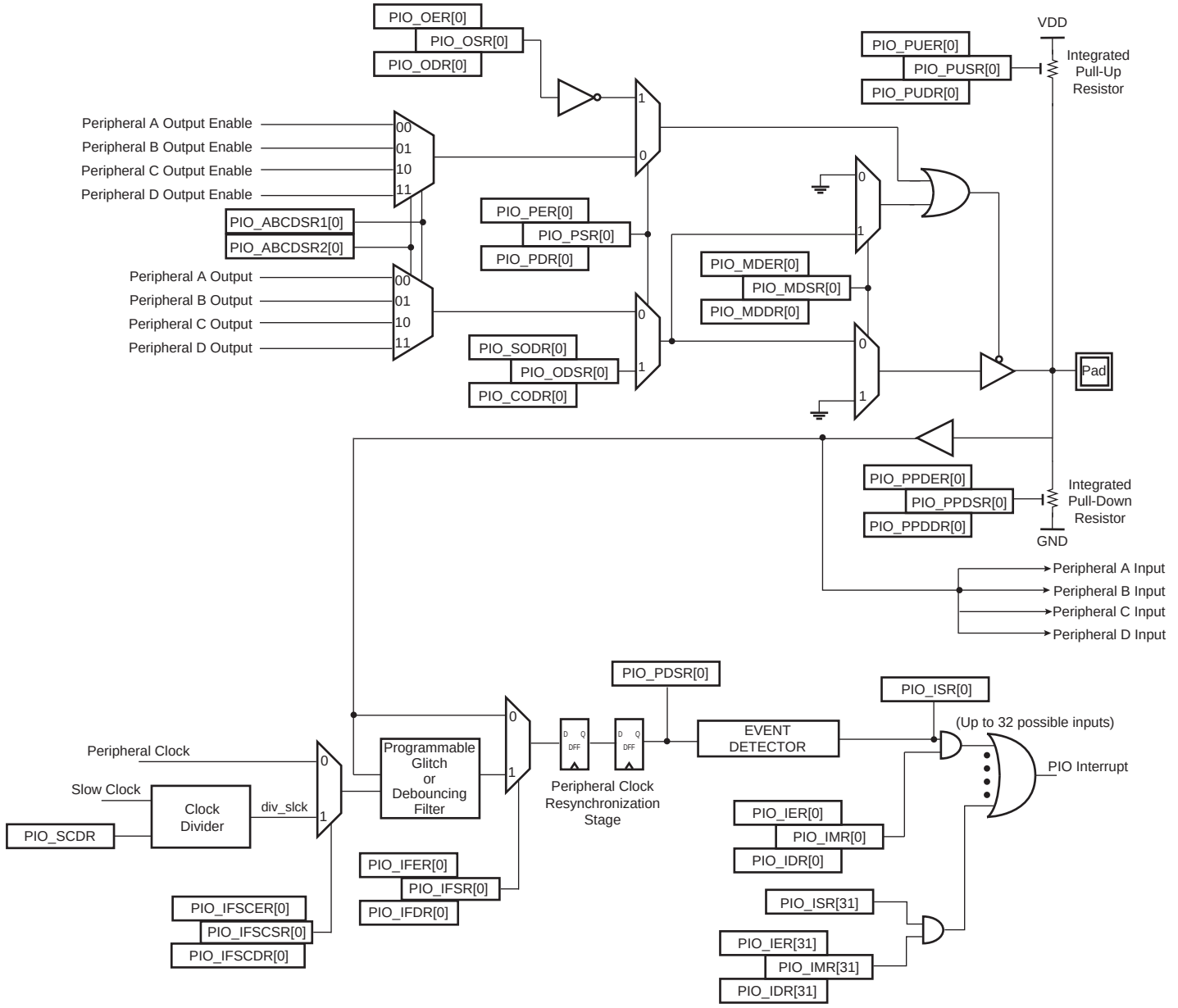
Instance	ID
PIOA	2
PIOB	2
PIOC	3
PIOD	3

22.5 Functional Description

The PIO Controller features up to 32 fully-programmable I/O lines. Most of the control logic associated to each I/O is represented in Figure 22-2. In this description each signal shown represents one of up to 32 possible indexes.

© 2017 Microchip Technology Inc.

Figure 22-2: I/O Line Control Logic



22.5.1 Pull-up and Pull-down Resistor Control

Each I/O line is designed with an embedded pull-up resistor and an embedded pull-down resistor. The pull-up resistor can be enabled or disabled by writing to the Pull-up Enable Register (PIO_PUER) or Pull-up Disable Register (PIO_PUDR), respectively. Writing to these registers results in setting or clearing the corresponding bit in the Pull-up Status Register (PIO_PUSR). Reading a one in PIO_PUSR means the pull-up is disabled and reading a zero means the pull-up is enabled. The pull-down resistor can be enabled or disabled by writing the Pull-down Enable Register (PIO_PPDER) or the Pull-down Disable Register (PIO_PPDDR), respectively. Writing in these registers results in setting or clearing the corresponding bit in the Pull-down Status Register (PIO_PPDSR). Reading a one in PIO_PPDSR means the pull-up is disabled and reading a zero means the pull-down is enabled.

Enabling the pull-down resistor while the pull-up resistor is still enabled is not possible. In this case, the write of PIO_PPDER for the relevant I/O line is discarded. Likewise, enabling the pull-up resistor while the pull-down resistor is still enabled is not possible. In this case, the write of PIO_PUER for the relevant I/O line is discarded.

Control of the pull-up resistor is possible regardless of the configuration of the I/O line.

After reset, depending on the I/O, pull-up or pull-down can be set.

22.5.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the Enable Register (PIO_PER) and the Disable Register (PIO_PDR). The Status Register (PIO_PSR) is the result of the set and clear registers and indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller. A value of zero indicates that the pin is controlled by the corresponding on-chip peripheral selected in the ABCD Select registers (PIO_ABCDSR1 and PIO_ABCDSR2). A value of one indicates the pin is controlled by the PIO Controller.

If a pin is used as a general-purpose I/O line (not multiplexed with an on-chip peripheral), PIO_PER and PIO_PDR have no effect and PIO_PSR returns a one for the corresponding bit.

After reset, the I/O lines are controlled by the PIO Controller, i.e., PIO_PSR resets at one. However, in some events, it is important that PIO lines are controlled by the peripheral (as in the case of memory chip select lines that must be driven inactive after reset, or for address lines that must be driven low for booting out of an external memory). Thus, the reset value of PIO_PSR is defined at the product level and depends on the multiplexing of the device.

22.5.3 Peripheral A or B or C or D Selection

The PIO Controller provides multiplexing of up to four peripheral functions on a single pin. The selection is performed by writing PIO_ABCDSR1 and PIO_ABCDSR2.

For each pin:

- The corresponding bit at level zero in PIO_ABCDSR1 and the corresponding bit at level zero in PIO_ABCDSR2 means peripheral A is selected.
- The corresponding bit at level one in PIO_ABCDSR1 and the corresponding bit at level zero in PIO_ABCDSR2 means peripheral B is selected.
- The corresponding bit at level zero in PIO_ABCDSR1 and the corresponding bit at level one in PIO_ABCDSR2 means peripheral C is selected.
- The corresponding bit at level one in PIO_ABCDSR1 and the corresponding bit at level one in PIO_ABCDSR2 means peripheral D is selected.

Note that multiplexing of peripheral lines A, B, C and D only affects the output line. The peripheral input lines are always connected to the pin input (see Figure 22-2).

Writing in PIO_ABCDSR1 and PIO_ABCDSR2 manages the multiplexing regardless of the configuration of the pin. However, assignment of a pin to a peripheral function requires a write in PIO_ABCDSR1 and PIO_ABCDSR2 in addition to a write in PIO_PDR.

After reset, PIO_ABCDSR1 and PIO_ABCDSR2 are zero, thus indicating that all the PIO lines are configured on peripheral A. However, peripheral A generally does not drive the pin as the PIO Controller resets in I/O line mode.

If the software selects a peripheral A, B, C or D which does not exist for a pin, no alternate functions are enabled for this pin and the selection is taken into account. The PIO Controller does not carry out checks to prevent selection of a peripheral which does not exist.

22.5.4 Output Control

When the I/O line is assigned to a peripheral function, i.e., the corresponding bit in PIO_PSR is at zero, the drive of the I/O line is controlled by the peripheral. Peripheral A or B or C or D depending on the value in PIO_ABCDSR1 and PIO_ABCDSR2 determines whether the pin is driven or not.

When the I/O line is controlled by the PIO Controller, the pin can be configured to be driven. This is done by writing the Output Enable Register (PIO_OER) and Output Disable Register (PIO_ODR). The results of these write operations are detected in the Output Status Register (PIO_OSR). When a bit in this register is at zero, the corresponding I/O line is used as an input only. When the bit is at one, the corresponding I/O line is driven by the PIO Controller.

The level driven on an I/O line can be determined by writing in the Set Output Data Register (PIO_SODR) and the Clear Output Data Register (PIO_CODR). These write operations, respectively, set and clear the Output Data Status Register (PIO_ODSR), which represents the data driven on the I/O lines. Writing in PIO_OER and PIO_ODR manages PIO_OSR whether the pin is configured to be controlled by the PIO Controller or assigned to a peripheral function. This enables configuration of the I/O line prior to setting it to be managed by the PIO Controller.

Similarly, writing in PIO_SODR and PIO_CODR affects PIO_ODSR. This is important as it defines the first level driven on the I/O line.

© 2017 Microchip Technology Inc.

22.5.5 Synchronous Data Output

Clearing one or more PIO line(s) and setting another one or more PIO line(s) synchronously cannot be done by using PIO_SODR and PIO_CODR. It requires two successive write operations into two different registers. To overcome this, the PIO Controller offers a direct control of PIO outputs by single write access to PIO_ODSR. Only bits unmasked by the Output Write Status Register (PIO_OWSR) are written. The mask bits in PIO_OWSR are set by writing to the Output Write Enable Register (PIO_OWER) and cleared by writing to the Output Write Disable Register (PIO_OWDR).

After reset, the synchronous data output is disabled on all the I/O lines as PIO_OWSR resets at 0x0.

22.5.6 Multi-Drive Control (Open Drain)

Each I/O can be independently programmed in open drain by using the multi-drive feature. This feature permits several drivers to be connected on the I/O line which is driven low only by each device. An external pull-up resistor (or enabling of the internal one) is generally required to guarantee a high level on the line.

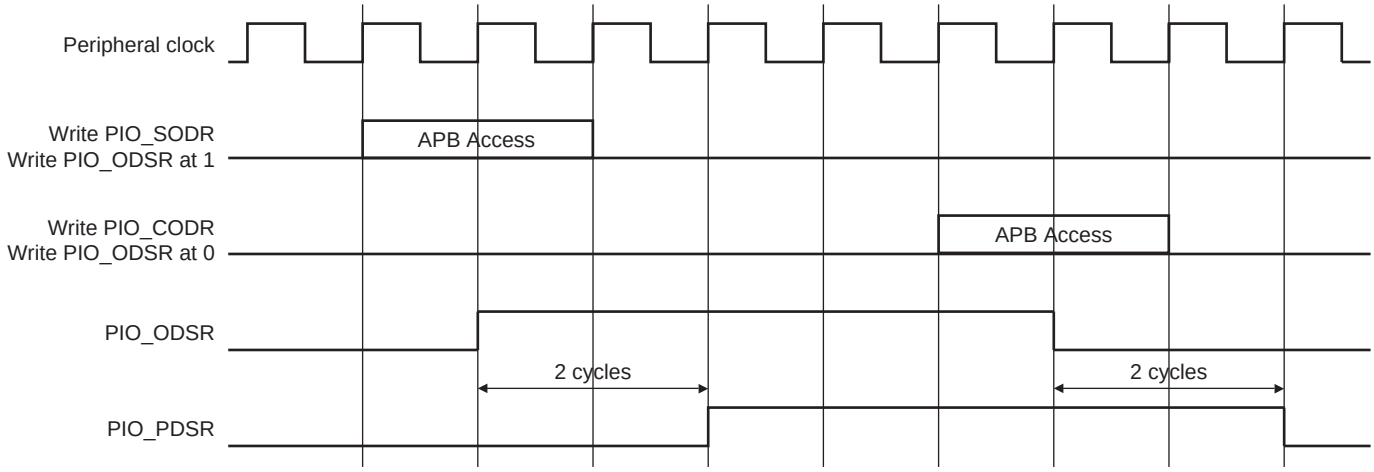
The multi-drive feature is controlled by the Multi-driver Enable Register (PIO_MDER) and the Multi-driver Disable Register (PIO_MDDR). The multi-drive can be selected whether the I/O line is controlled by the PIO Controller or assigned to a peripheral function. The Multi-driver Status Register (PIO_MDSR) indicates the pins that are configured to support external drivers.

After reset, the multi-drive feature is disabled on all pins, i.e., PIO_MDSR resets at value 0x0.

22.5.7 Output Line Timings

Figure 22-3 shows how the outputs are driven either by writing PIO_SODR or PIO_CODR, or by directly writing PIO_ODSR. This last case is valid only if the corresponding bit in PIO_OWSR is set. Figure 22-3 also shows when the feedback in the Pin Data Status Register (PIO_PDSR) is available.

Figure 22-3: Output Line Timings



22.5.8 Inputs

The level on each I/O line can be read through PIO_PDSR. This register indicates the level of the I/O lines regardless of their configuration, whether uniquely as an input, or driven by the PIO Controller, or driven by a peripheral.

Reading the I/O line levels requires the clock of the PIO Controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.

22.5.9 Input Glitch and Debouncing Filters

Optional input glitch and debouncing filters are independently programmable on each I/O line.

The glitch filter can filter a glitch with a duration of less than 1/2 peripheral clock and the debouncing filter can filter a pulse of less than 1/2 period of a programmable divided slow clock.

The selection between glitch filtering or debounce filtering is done by writing in the PIO Input Filter Slow Clock Disable Register (PIO_IFSCDR) and the PIO Input Filter Slow Clock Enable Register (PIO_IFSCER). Writing PIO_IFSCDR and PIO_IFSCER, respectively, sets and clears bits in the Input Filter Slow Clock Status Register (PIO_IFSCSR).

© 2017 Microchip Technology Inc.

The current selection status can be checked by reading the PIO_IFSCSR.

- If PIO_IFSCSR[i] = 0: The glitch filter can filter a glitch with a duration of less than 1/2 master clock period.
- If PIO_IFSCSR[i] = 1: The debouncing filter can filter a pulse with a duration of less than 1/2 programmable divided slow clock period.

For the debouncing filter, the period of the divided slow clock is defined by writing in the DIV field of the Slow Clock Divider Debouncing Register (PIO_SCDR):

$t_{div\_slck} = ((DIV + 1) \times 2) \times t_{slck}$

When the glitch or debouncing filter is enabled, a glitch or pulse with a duration of less than 1/2 selected clock cycle (selected clock represents peripheral clock or divided slow clock depending on PIO_IFSCDR and PIO_IFSCER programming) is automatically rejected, while a pulse with a duration of one selected clock (peripheral clock or divided slow clock) cycle or more is accepted. For pulse durations between 1/2 selected clock cycle and one selected clock cycle, the pulse may or may not be taken into account, depending on the precise timing of its occurrence. Thus for a pulse to be visible, it must exceed one selected clock cycle, whereas for a glitch to be reliably filtered out, its duration must not exceed 1/2 selected clock cycle.

The filters also introduce some latencies, illustrated in Figure 22-4 and Figure 22-5.

The glitch filters are controlled by the Input Filter Enable Register (PIO_IFER), the Input Filter Disable Register (PIO_IFDR) and the Input Filter Status Register (PIO_IFSR). Writing PIO_IFER and PIO_IFDR respectively sets and clears bits in PIO_IFSR. This last register enables the glitch filter on the I/O lines.

When the glitch and/or debouncing filter is enabled, it does not modify the behavior of the inputs on the peripherals. It acts only on the value read in PIO_PDSR and on the input change interrupt detection. The glitch and debouncing filters require that the peripheral clock is enabled.

Figure 22-4: Input Glitch Filter Timing

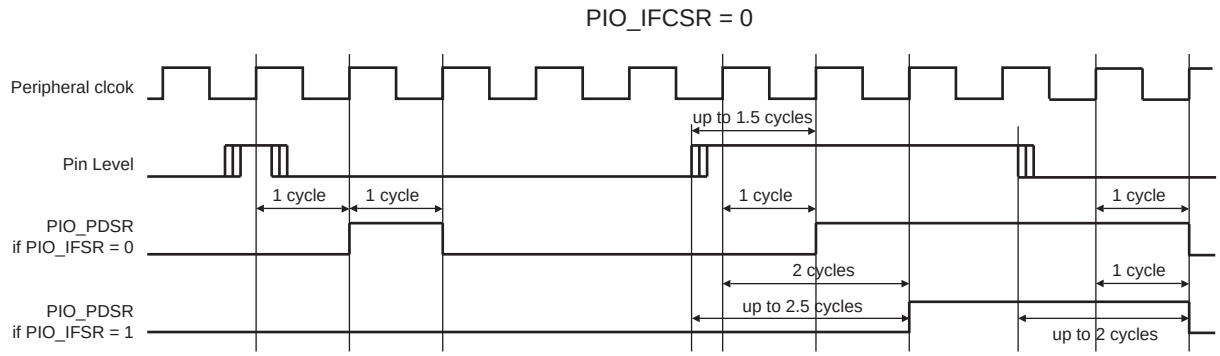
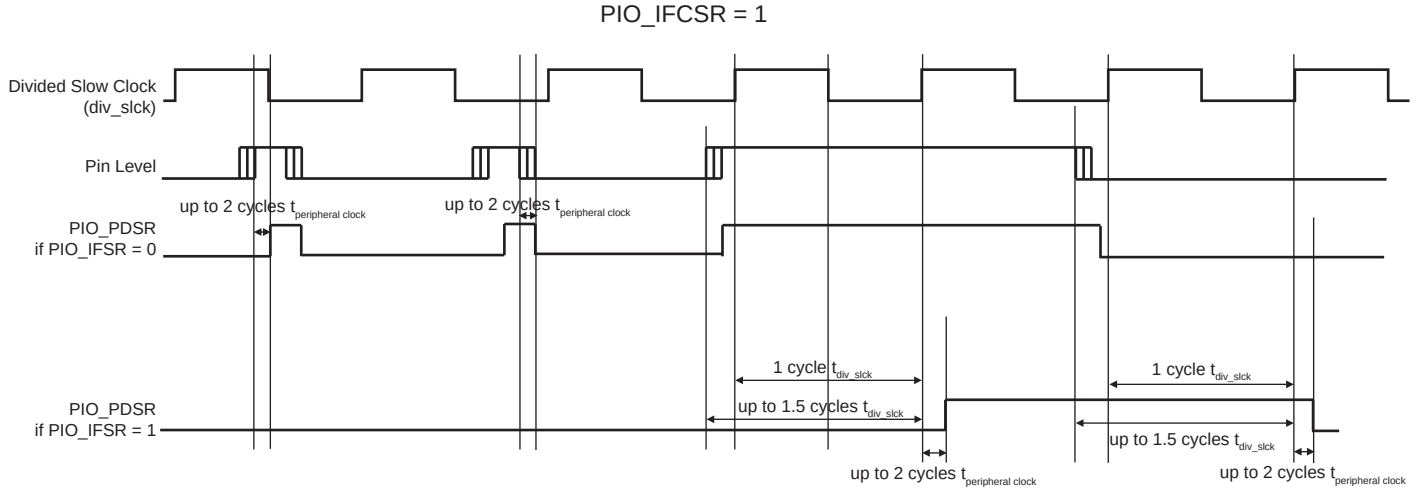


Figure 22-5: Input Debouncing Filter Timing



© 2017 Microchip Technology Inc.

22.5.10 Input Edge/Level Interrupt

The PIO Controller can be programmed to generate an interrupt when it detects an edge or a level on an I/O line. The Input Edge/Level interrupt is controlled by writing the Interrupt Enable Register (PIO_IER) and the Interrupt Disable Register (PIO_IDR), which enable and disable the input change interrupt respectively by setting and clearing the corresponding bit in the Interrupt Mask Register (PIO_IMR). As input change detection is possible only by comparing two successive samplings of the input of the I/O line, the peripheral clock must be enabled. The Input Change interrupt is available regardless of the configuration of the I/O line, i.e., configured as an input only, controlled by the PIO Controller or assigned to a peripheral function.

By default, the interrupt can be generated at any time an edge is detected on the input.

Some additional interrupt modes can be enabled/disabled by writing in the Additional Interrupt Modes Enable Register (PIO_AIMER) and Additional Interrupt Modes Disable Register (PIO_AIMDR). The current state of this selection can be read through the Additional Interrupt Modes Mask Register (PIO_AIMMR).

These additional modes are:

- Rising edge detection
- Falling edge detection
- Low-level detection
- High-level detection

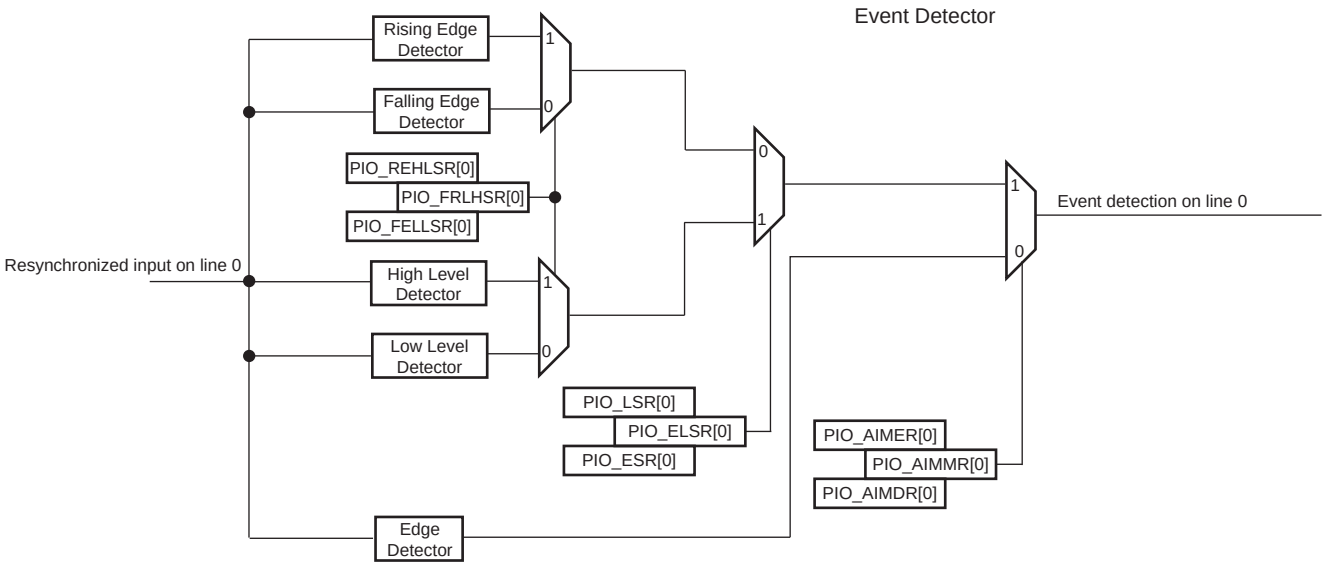
In order to select an additional interrupt mode:

- The type of event detection (edge or level) must be selected by writing in the Edge Select Register (PIO_ESR) and Level Select Register (PIO_LSR) which select, respectively, the edge and level detection. The current status of this selection is accessible through the Edge/Level Status Register (PIO_ELSR).
- The polarity of the event detection (rising/falling edge or high/low-level) must be selected by writing in the Falling Edge/Low-Level Select Register (PIO_FELLSR) and Rising Edge/High-Level Select Register (PIO_REHLSR) which allow to select falling or rising edge (if edge is selected in PIO_ELSR) edge or high- or low-level detection (if level is selected in PIO_ELSR). The current status of this selection is accessible through the Fall/Rise - Low/High Status Register (PIO_FRLHSR).

When an input edge or level is detected on an I/O line, the corresponding bit in the Interrupt Status Register (PIO_ISR) is set. If the corresponding bit in PIO_IMR is set, the PIO Controller interrupt line is asserted.The interrupt signals of the 32 channels are ORed-wired together to generate a single interrupt signal to the interrupt controller.

When the software reads PIO_ISR, all the interrupts are automatically cleared. This signifies that all the interrupts that are pending when PIO_ISR is read must be handled. When an Interrupt is enabled on a “level”, the interrupt is generated as long as the interrupt source is not cleared, even if some read accesses in PIO_ISR are performed.

Figure 22-6: Event Detector on Input Lines (Figure Represents Line 0)



Example of interrupt generation on following lines:

- Rising edge on PIO line 0
- Falling edge on PIO line 1
- Rising edge on PIO line 2
- Low-level on PIO line 3
- High-level on PIO line 4
- High-level on PIO line 5
- Falling edge on PIO line 6
- Rising edge on PIO line 7
- Any edge on the other lines

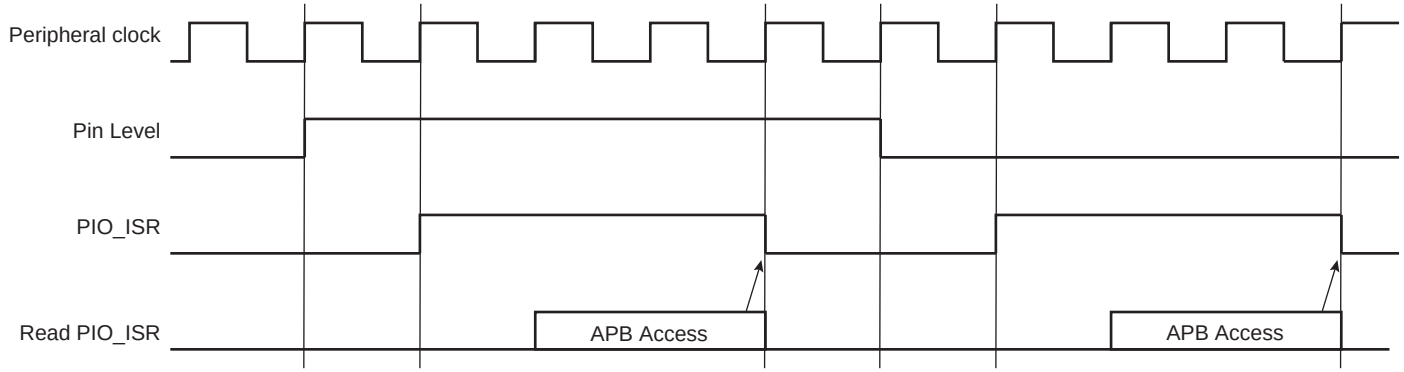
Table 22-2 provides the required configuration for this example.

Table 22-2: Configuration for Example Interrupt Generation

Configuration	Description
Interrupt Mode	All the interrupt sources are enabled by writing 32'hFFFF_FFFF in PIO_IER. Then the additional interrupt mode is enabled for lines 0 to 7 by writing 32'h0000_00FF in PIO_AIMER.
Edge or Level Detection	Lines 3, 4 and 5 are configured in level detection by writing 32'h0000_0038 in PIO_LSR. The other lines are configured in edge detection by default, if they have not been previously configured. Otherwise, lines 0, 1, 2, 6 and 7 must be configured in edge detection by writing 32'h0000_00C7 in PIO_ESR.
Falling/Rising Edge or Low/High-Level Detection	Lines 0, 2, 4, 5 and 7 are configured in rising edge or high-level detection by writing 32'h0000_00B5 in PIO_REHLSR. The other lines are configured in falling edge or low-level detection by default if they have not been previously configured. Otherwise, lines 1, 3 and 6 must be configured in falling edge/low-level detection by writing 32'h0000_004A in PIO_FELLSR.

© 2017 Microchip Technology Inc.

Figure 22-7: Input Change Interrupt Timings When No Additional Interrupt Modes



22.5.11 Programmable I/O Delays

The PIO interface consists of a series of signals driven by peripherals or directly by software. The simultaneous switching outputs on these busses may lead to a peak of current in the internal and external power supply lines.

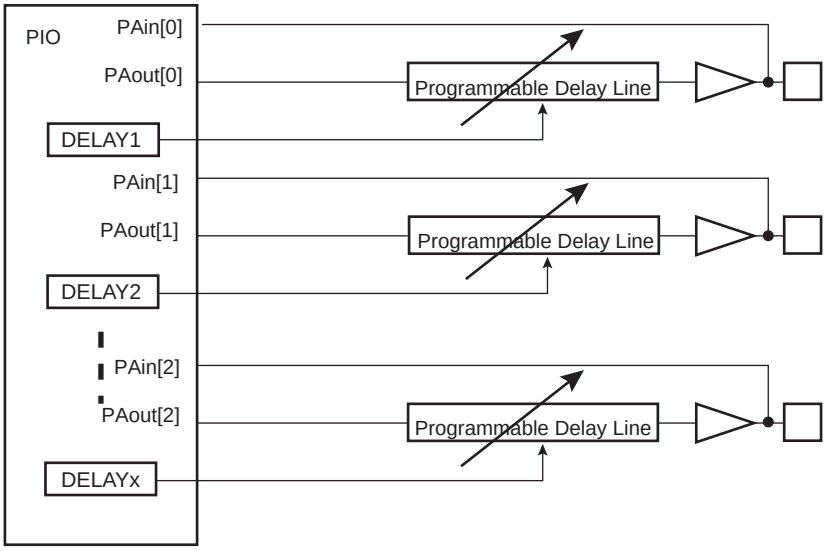
In order to reduce the current peak in such cases, additional propagation delays can be adjusted independently for pad buffers by means of configuration registers, PIO_DELAYR.

For each I/O supporting the additional programmable delay, the delay ranges from 0 to 4 ns (worst case process, voltage, temperature). The delay can differ between I/Os supporting this feature. Delay can be modified per programming for each I/O. The minimal additional delay that can be programmed on a PAD supporting this feature is 1/16 of the maximum programmable delay.

Only pads PA[15:11], and PA[20:18] can be configured.

When programming 0x0 in fields, no delay is added (reset value) and the propagation delay of the pad buffers is the inherent delay of the pad buffer. When programming 0xF in fields, the propagation delay of the corresponding pad is maximal.

Figure 22-8: Programmable I/O Delays



22.5.12 Programmable I/O Drive

It is possible to configure the I/O drive for pads PA[31:0], PB[18:0], PC[31:0]. Refer to the section “Electrical Characteristics”.

22.5.13 Programmable Schmitt Trigger

It is possible to configure each input for the Schmitt trigger. By default the Schmitt trigger is active. Disabling the Schmitt trigger is requested when using the QTouch® Library.

22.5.14 I/O Lines Programming Example

The programming example shown in Table 22-3 is used to obtain the following configuration:

- 4-bit output port on I/O lines 0 to 3 (should be written in a single write operation), open-drain, with pull-up resistor
- Four output signals on I/O lines 4 to 7 (to drive LEDs for example), driven high and low, no pull-up resistor, no pull-down resistor
- Four input signals on I/O lines 8 to 11 (to read push-button states for example), with pull-up resistors, glitch filters and input change interrupts
- Four input signals on I/O line 12 to 15 to read an external device status (polled, thus no input change interrupt), no pull-up resistor, no glitch filter
- I/O lines 16 to 19 assigned to peripheral A functions with pull-up resistor
- I/O lines 20 to 23 assigned to peripheral B functions with pull-down resistor
- I/O lines 24 to 27 assigned to peripheral C with input change interrupt, no pull-up resistor and no pull-down resistor
- I/O lines 28 to 31 assigned to peripheral D, no pull-up resistor and no pull-down resistor

Table 22-3: Programming Example

Register	Value to be Written
PIO_PER	0x0000_FFFF
PIO_PDR	0xFFFF_0000
PIO_OER	0x0000_00FF
PIO_ODR	0xFFFF_FF00
PIO_IFER	0x0000_0F00
PIO_IFDR	0xFFFF_F0FF
PIO_SODR	0x0000_0000
PIO_CODR	0x0FFF_FFFF
PIO_IER	0x0F00_0F00
PIO_IDR	0xF0FF_F0FF
PIO_MDER	0x0000_000F
PIO_MDDR	0xFFFF_FFF0
PIO_PUDR	0xFFF0_00F0
PIO_PUER	0x000F_FF0F
PIO_PPDDR	0xFF0F_FFFF
PIO_PPDER	0x00F0_0000
PIO_ABCDSR1	0xF0F0_0000
PIO_ABCDSR2	0xFF00_0000
PIO_OWER	0x0000_000F
PIO_OWDR	0x0FFF_ FFF0

22.5.15 Register Write Protection

To prevent any single software error from corrupting PIO behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the PIO Write Protection Mode Register (PIO_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the PIO Write Protection Status Register (PIO_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the PIO_WPSR.

The following registers can be write-protected:

- PIO Enable Register
- PIO Disable Register

© 2017 Microchip Technology Inc.

- PIO Output Enable Register
- PIO Output Disable Register
- PIO Input Filter Enable Register
- PIO Input Filter Disable Register
- PIO Multi-driver Enable Register
- PIO Multi-driver Disable Register
- PIO Pull-Up Disable Register
- PIO Pull-Up Enable Register
- PIO Peripheral ABCD Select Register 1
- PIO Peripheral ABCD Select Register 2
- PIO Output Write Enable Register
- PIO Output Write Disable Register
- PIO Pad Pull-Down Disable Register
- PIO Pad Pull-Down Enable Register

© 2017 Microchip Technology Inc.

22.6 Parallel Input/Output Controller (PIO) User Interface

Each I/O line controlled by the PIO Controller is associated with a bit in each of the PIO Controller User Interface registers. Each register is 32-bit wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero. If the I/O line is not multiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO_PSR returns one systematically.

Table 22-4: Register Mapping

Offset	Register	Name	Access	Reset
0x0000	PIO Enable Register	PIO_PER	Write-only	–
0x0004	PIO Disable Register	PIO_PDR	Write-only	–
0x0008	PIO Status Register	PIO_PSR	Read-only	(1)
0x000C	Reserved	–	–	–
0x0010	Output Enable Register	PIO_OER	Write-only	–
0x0014	Output Disable Register	PIO_ODR	Write-only	–
0x0018	Output Status Register	PIO_OSR	Read-only	0x00000000
0x001C	Reserved	–	–	–
0x0020	Glitch Input Filter Enable Register	PIO_IFER	Write-only	–
0x0024	Glitch Input Filter Disable Register	PIO_IFDR	Write-only	–
0x0028	Glitch Input Filter Status Register	PIO_IFSR	Read-only	0x00000000
0x002C	Reserved	–	–	–
0x0030	Set Output Data Register	PIO_SODR	Write-only	–
0x0034	Clear Output Data Register	PIO_CODR	Write-only	
0x0038	Output Data Status Register	PIO_ODSR	Read-only or(2) Read/Write	–
0x003C	Pin Data Status Register	PIO_PDSR	Read-only	(3)
0x0040	Interrupt Enable Register	PIO_IER	Write-only	–
0x0044	Interrupt Disable Register	PIO_IDR	Write-only	–
0x0048	Interrupt Mask Register	PIO_IMR	Read-only	0x00000000
0x004C	Interrupt Status Register(4)	PIO_ISR	Read-only	0x00000000
0x0050	Multi-driver Enable Register	PIO_MDER	Write-only	–
0x0054	Multi-driver Disable Register	PIO_MDDR	Write-only	–
0x0058	Multi-driver Status Register	PIO_MDSR	Read-only	0x00000000
0x005C	Reserved	–	–	–
0x0060	Pull-up Disable Register	PIO_PUDR	Write-only	–
0x0064	Pull-up Enable Register	PIO_PUER	Write-only	–
0x0068	Pad Pull-up Status Register	PIO_PUSR	Read-only	(1)
0x006C	Reserved	–	–	–

© 2017 Microchip Technology Inc.

Table 22-4: Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0070	Peripheral Select Register 1	PIO_ABCDSR1	Read/Write	0x00000000
0x0074	Peripheral Select Register 2	PIO_ABCDSR2	Read/Write	0x00000000
0x0078–0x007C	Reserved	–	–	–
0x0080	Input Filter Slow Clock Disable Register	PIO_IFSCDR	Write-only	–
0x0084	Input Filter Slow Clock Enable Register	PIO_IFSCER	Write-only	–
0x0088	Input Filter Slow Clock Status Register	PIO_IFSCSR	Read-only	0x00000000
0x008C	Slow Clock Divider Debouncing Register	PIO_SCDR	Read/Write	0x00000000
0x0090	Pad Pull-down Disable Register	PIO_PPDDR	Write-only	–
0x0094	Pad Pull-down Enable Register	PIO_PPDER	Write-only	–
0x0098	Pad Pull-down Status Register	PIO_PPDSR	Read-only	(1)
0x009C	Reserved	–	–	–
0x00A0	Output Write Enable	PIO_OWER	Write-only	–
0x00A4	Output Write Disable	PIO_OWDR	Write-only	–
0x00A8	Output Write Status Register	PIO_OWSR	Read-only	0x00000000
0x00AC	Reserved	–	–	–
0x00B0	Additional Interrupt Modes Enable Register	PIO_AIMER	Write-only	–
0x00B4	Additional Interrupt Modes Disable Register	PIO_AIMDR	Write-only	–
0x00B8	Additional Interrupt Modes Mask Register	PIO_AIMMR	Read-only	0x00000000
0x00BC	Reserved	–	–	–
0x00C0	Edge Select Register	PIO_ESR	Write-only	–
0x00C4	Level Select Register	PIO_LSR	Write-only	–
0x00C8	Edge/Level Status Register	PIO_ELSR	Read-only	0x00000000
0x00CC	Reserved	–	–	–
0x00D0	Falling Edge/Low-Level Select Register	PIO_FELLSR	Write-only	–
0x00D4	Rising Edge/High-Level Select Register	PIO_REHLSR	Write-only	–
0x00D8	Fall/Rise - Low/High Status Register	PIO_FRLHSR	Read-only	0x00000000
0x00DC	Reserved	–	–	–
0x00E0	Reserved	–	–	–
0x00E4	Write Protection Mode Register	PIO_WPMR	Read/Write	0x00000000
0x00E8	Write Protection Status Register	PIO_WPSR	Read-only	0x00000000
0x00EC–0x00FC	Reserved	–	–	–
0x0100	Schmitt Trigger Register	PIO_SCHMITT	Read/Write	0x00000000
0x0104–0x010C	Reserved	–	–	–
0x0110	I/O Delay Register	PIO_DELAYR	Read/Write	0x00000000
0x0114	I/O Drive Register 1	PIO_DRIVER1	Read/Write	0x00000000
0x0118	I/O Drive Register 2	PIO_DRIVER2	Read/Write	0x00000000

© 2017 Microchip Technology Inc.

Table 22-4: Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x011C	Reserved	–	–	–
0x0120–0x014C	Reserved	–	–	–

Note 1: Reset value depends on the product implementation.

2: PIO_ODSR is Read-only or Read/Write depending on PIO_OWSR I/O lines.

3: Reset value of PIO_PDSR depends on the level of the I/O lines. Reading the I/O line levels requires the clock of the PIO Controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.

4: PIO_ISR is reset at 0x0. However, the first read of the register may read a different value as input changes may have occurred.

5: If an offset is not listed in the table it must be considered as reserved.

© 2017 Microchip Technology Inc.

22.6.1 PIO Enable Register

Name:PIO_PER

Address:0xFFFFF400 (PIOA), 0xFFFFF600 (PIOB), 0xFFFFF800 (PIOC), 0xFFFFFA00 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: PIO Enable

0: No effect.

1: Enables the PIO to control the corresponding pin (disables peripheral control of the pin).

22.6.2 PIO Disable Register

Name:PIO_PDR

Address:0xFFFFF404 (PIOA), 0xFFFFF604 (PIOB), 0xFFFFF804 (PIOC), 0xFFFFFA04 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: PIO Disable

0: No effect.

1: Disables the PIO from controlling the corresponding pin (enables peripheral control of the pin).

© 2017 Microchip Technology Inc.

22.6.3 PIO Status Register

Name:PIO_PSR

Address:0xFFFFF408 (PIOA), 0xFFFFF608 (PIOB), 0xFFFFF808 (PIOC), 0xFFFFFA08 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: PIO Status

0: PIO is inactive on the corresponding I/O line (peripheral is active).

1: PIO is active on the corresponding I/O line (peripheral is inactive).

22.6.4 PIO Output Enable Register

Name:PIO_OER

Address:0xFFFFF410 (PIOA), 0xFFFFF610 (PIOB), 0xFFFFF810 (PIOC), 0xFFFFFA10 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Output Enable

0: No effect.

1: Enables the output on the I/O line.

© 2017 Microchip Technology Inc.

22.6.5 PIO Output Disable Register

Name:PIO_ODR

Address:0xFFFFF414 (PIOA), 0xFFFFF614 (PIOB), 0xFFFFF814 (PIOC), 0xFFFFFA14 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Output Disable

0: No effect.

1: Disables the output on the I/O line.

22.6.6 PIO Output Status Register

Name:PIO_OSR

Address:0xFFFFF418 (PIOA), 0xFFFFF618 (PIOB), 0xFFFFF818 (PIOC), 0xFFFFFA18 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Output Status

0: The I/O line is a pure input.

1: The I/O line is enabled in output.

© 2017 Microchip Technology Inc.

22.6.7 PIO Input Filter Enable Register

Name:PIO_IFER

Address:0xFFFFF420 (PIOA), 0xFFFFF620 (PIOB), 0xFFFFF820 (PIOC), 0xFFFFFA20 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Input Filter Enable

0: No effect.

1: Enables the input glitch filter on the I/O line.

22.6.8 PIO Input Filter Disable Register

Name:PIO_IFDR

Address:0xFFFFF424 (PIOA), 0xFFFFF624 (PIOB), 0xFFFFF824 (PIOC), 0xFFFFFA24 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Input Filter Disable

0: No effect.

1: Disables the input glitch filter on the I/O line.

© 2017 Microchip Technology Inc.

22.6.9 PIO Input Filter Status Register

Name:PIO_IFSR

Address:0xFFFFF428 (PIOA), 0xFFFFF628 (PIOB), 0xFFFFF828 (PIOC), 0xFFFFFA28 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Input Filter Status

0: The input glitch filter is disabled on the I/O line.

1: The input glitch filter is enabled on the I/O line.

22.6.10 PIO Set Output Data Register

Name:PIO_SODR

Address:0xFFFFF430 (PIOA), 0xFFFFF630 (PIOB), 0xFFFFF830 (PIOC), 0xFFFFFA30 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Set Output Data

0: No effect.

1: Sets the data to be driven on the I/O line.

© 2017 Microchip Technology Inc.

22.6.11 PIO Clear Output Data Register

Name:PIO_CODR

Address:0xFFFFF434 (PIOA), 0xFFFFF634 (PIOB), 0xFFFFF834 (PIOC), 0xFFFFFA34 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Clear Output Data

0: No effect.

1: Clears the data to be driven on the I/O line.

22.6.12 PIO Output Data Status Register

Name:PIO_ODSR

Address:0xFFFFF438 (PIOA), 0xFFFFF638 (PIOB), 0xFFFFF838 (PIOC), 0xFFFFFA38 (PIOD)

Access: Read-only or Read/Write

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Output Data Status

0: The data to be driven on the I/O line is 0.

1: The data to be driven on the I/O line is 1.

© 2017 Microchip Technology Inc.

22.6.13 PIO Pin Data Status Register

Name:PIO_PDSR

Address:0xFFFFF43C (PIOA), 0xFFFFF63C (PIOB), 0xFFFFF83C (PIOC), 0xFFFFFA3C (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Output Data Status

0: The I/O line is at level 0.

1: The I/O line is at level 1.

22.6.14 PIO Interrupt Enable Register

Name:PIO_IER

Address:0xFFFFF440 (PIOA), 0xFFFFF640 (PIOB), 0xFFFFF840 (PIOC), 0xFFFFFA40 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Input Change Interrupt Enable

0: No effect.

1: Enables the input change interrupt on the I/O line.

© 2017 Microchip Technology Inc.

22.6.15 PIO Interrupt Disable Register

Name:PIO_IDR

Address:0xFFFFF444 (PIOA), 0xFFFFF644 (PIOB), 0xFFFFF844 (PIOC), 0xFFFFFA44 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Input Change Interrupt Disable

0: No effect.

1: Disables the input change interrupt on the I/O line.

22.6.16 PIO Interrupt Mask Register

Name:PIO_IMR

Address:0xFFFFF448 (PIOA), 0xFFFFF648 (PIOB), 0xFFFFF848 (PIOC), 0xFFFFFA48 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Input Change Interrupt Mask

0: Input change interrupt is disabled on the I/O line.

1: Input change interrupt is enabled on the I/O line.

© 2017 Microchip Technology Inc.

22.6.17 PIO Interrupt Status Register

Name:PIO_ISR

Address:0xFFFFF44C (PIOA), 0xFFFFF64C (PIOB), 0xFFFFF84C (PIOC), 0xFFFFFA4C (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Input Change Interrupt Status

0: No input change has been detected on the I/O line since PIO_ISR was last read or since reset.

1: At least one input change has been detected on the I/O line since PIO_ISR was last read or since reset.

22.6.18 PIO Multi-driver Enable Register

Name:PIO_MDER

Address:0xFFFFF450 (PIOA), 0xFFFFF650 (PIOB), 0xFFFFF850 (PIOC), 0xFFFFFA50 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0-P31: Multi-drive Enable

0: No effect.

1: Enables multi-drive on the I/O line.

© 2017 Microchip Technology Inc.

22.6.19 PIO Multi-driver Disable Register

Name:PIO_MDDR

Address:0xFFFFF454 (PIOA), 0xFFFFF654 (PIOB), 0xFFFFF854 (PIOC), 0xFFFFFA54 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Multi-drive Disable

0: No effect.

1: Disables multi-drive on the I/O line.

22.6.20 PIO Multi-driver Status Register

Name:PIO_MDSR

Address:0xFFFFF458 (PIOA), 0xFFFFF658 (PIOB), 0xFFFFF858 (PIOC), 0xFFFFFA58 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Multi-drive Status

0: The multi-drive is disabled on the I/O line. The pin is driven at high- and low-level.

1: The multi-drive is enabled on the I/O line. The pin is driven at low-level only.

© 2017 Microchip Technology Inc.

22.6.21 PIO Pull-Up Disable Register

Name:PIO_PUDR

Address:0xFFFFF460 (PIOA), 0xFFFFF660 (PIOB), 0xFFFFF860 (PIOC), 0xFFFFFA60 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Pull-Up Disable

0: No effect.

1: Disables the pull-up resistor on the I/O line.

22.6.22 PIO Pull-Up Enable Register

Name:PIO_PUER

Address:0xFFFFF464 (PIOA), 0xFFFFF664 (PIOB), 0xFFFFF864 (PIOC), 0xFFFFFA64 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Pull-Up Enable

0: No effect.

1: Enables the pull-up resistor on the I/O line.

© 2017 Microchip Technology Inc.

22.6.23 PIO Pull-Up Status Register

Name:PIO_PUSR

Address:0xFFFFF468 (PIOA), 0xFFFFF668 (PIOB), 0xFFFFF868 (PIOC), 0xFFFFFA68 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Pull-Up Status

0: Pull-up resistor is enabled on the I/O line.

1: Pull-up resistor is disabled on the I/O line.

22.6.24 PIO Peripheral ABCD Select Register 1

Name:PIO_ABCDSR1

Access: Read/Write

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Peripheral Select

If the same bit is set to 0 in PIO_ABCDSR2:

0: Assigns the I/O line to the Peripheral A function.

1: Assigns the I/O line to the Peripheral B function.

If the same bit is set to 1 in PIO_ABCDSR2:

0: Assigns the I/O line to the Peripheral C function.

1: Assigns the I/O line to the Peripheral D function.

© 2017 Microchip Technology Inc.

22.6.25 PIO Peripheral ABCD Select Register 2

Name:PIO_ABCDSR2

Address:0xFFFFF470 (PIOA), 0xFFFFF670 (PIOB), 0xFFFFF870 (PIOC), 0xFFFFFA70 (PIOD)

Access: Read/Write

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Peripheral Select

If the same bit is set to 0 in PIO_ABCDSR1:

0: Assigns the I/O line to the Peripheral A function.

1: Assigns the I/O line to the Peripheral C function.

If the same bit is set to 1 in PIO_ABCDSR1:

0: Assigns the I/O line to the Peripheral B function.

1: Assigns the I/O line to the Peripheral D function.

22.6.26 PIO Input Filter Slow Clock Disable Register

Name:PIO_IFSCDR

Address:0xFFFFF480 (PIOA), 0xFFFFF680 (PIOB), 0xFFFFF880 (PIOC), 0xFFFFFA80 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Peripheral Clock Glitch Filtering Select

0: No effect.

1: The glitch filter is able to filter glitches with a duration < $t_{peripheral\ clock}/2$.

© 2017 Microchip Technology Inc.

22.6.27 PIO Input Filter Slow Clock Enable Register

Name:PIO_IFSCER

Address:0xFFFFF484 (PIOA), 0xFFFFF684 (PIOB), 0xFFFFF884 (PIOC), 0xFFFFFA84 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Slow Clock Debouncing Filtering Select

0: No effect.

1: The debouncing filter is able to filter pulses with a duration < $t_{div\_slck}/2$.

22.6.28 PIO Input Filter Slow Clock Status Register

Name:PIO_IFSCSR

Address:0xFFFFF488 (PIOA), 0xFFFFF688 (PIOB), 0xFFFFF888 (PIOC), 0xFFFFFA88 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Glitch or Debouncing Filter Selection Status

0: The glitch filter is able to filter glitches with a duration < $t_{peripheral\ clock}/2$.

1: The debouncing filter is able to filter pulses with a duration < $t_{div\_slck}/2$.

© 2017 Microchip Technology Inc.

22.6.29 PIO Slow Clock Divider Debouncing Register

Name:PIO_SCDR

Address:0xFFFFF48C (PIOA), 0xFFFFF68C (PIOB), 0xFFFFF88C (PIOC), 0xFFFFFA8C (PIOD)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	DIV					
7	6	5	4	3	2	1	0
DIV							

DIV: Slow Clock Divider Selection for Debouncing

$t_{div\_slck} = ((DIV + 1) \times 2) \times t_{slck}$

22.6.30 PIO Pad Pull-Down Disable Register

Name:PIO_PPDDR

Address:0xFFFFF490 (PIOA), 0xFFFFF690 (PIOB), 0xFFFFF890 (PIOC), 0xFFFFFA90 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Pull-Down Disable

0: No effect.

1: Disables the pull-down resistor on the I/O line.

© 2017 Microchip Technology Inc.

22.6.31 PIO Pad Pull-Down Enable Register

Name:PIO_PPDER

Address:0xFFFFF494 (PIOA), 0xFFFFF694 (PIOB), 0xFFFFF894 (PIOC), 0xFFFFFA94 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Pull-Down Enable

0: No effect.

1: Enables the pull-down resistor on the I/O line.

22.6.32 PIO Pad Pull-Down Status Register

Name:PIO_PPDSR

Address:0xFFFFF498 (PIOA), 0xFFFFF698 (PIOB), 0xFFFFF898 (PIOC), 0xFFFFFA98 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Pull-Down Status

0: Pull-down resistor is enabled on the I/O line.

1: Pull-down resistor is disabled on the I/O line.

© 2017 Microchip Technology Inc.

22.6.33 PIO Output Write Enable Register

Name:PIO_OWER

Address:0xFFFFF4A0 (PIOA), 0xFFFFF6A0 (PIOB), 0xFFFFF8A0 (PIOC), 0xFFFFFAA0 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Output Write Enable

0: No effect.

1: Enables writing PIO_ODSR for the I/O line.

22.6.34 PIO Output Write Disable Register

Name:PIO_OWDR

Address:0xFFFFF4A4 (PIOA), 0xFFFFF6A4 (PIOB), 0xFFFFF8A4 (PIOC), 0xFFFFFAA4 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

P0–P31: Output Write Disable

0: No effect.

1: Disables writing PIO_ODSR for the I/O line.

© 2017 Microchip Technology Inc.

22.6.35 PIO Output Write Status Register

Name:PIO_OWSR

Address:0xFFFFF4A8 (PIOA), 0xFFFFF6A8 (PIOB), 0xFFFFF8A8 (PIOC), 0xFFFFFAA8 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Output Write Status

0: Writing PIO_ODSR does not affect the I/O line.

1: Writing PIO_ODSR affects the I/O line.

22.6.36 PIO Additional Interrupt Modes Enable Register

Name:PIO_AIMER

Address:0xFFFFF4B0 (PIOA), 0xFFFFF6B0 (PIOB), 0xFFFFF8B0 (PIOC), 0xFFFFFAB0 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Additional Interrupt Modes Enable

0: No effect.

1: The interrupt source is the event described in PIO_ELSR and PIO_FRLHSR.

© 2017 Microchip Technology Inc.

22.6.37 PIO Additional Interrupt Modes Disable Register

Name:PIO_AIMDR

Address:0xFFFFF4B4 (PIOA), 0xFFFFF6B4 (PIOB), 0xFFFFF8B4 (PIOC), 0xFFFFFAB4 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Additional Interrupt Modes Disable

0: No effect.

1: The interrupt mode is set to the default interrupt mode (both-edge detection).

22.6.38 PIO Additional Interrupt Modes Mask Register

Name:PIO_AIMMR

Address:0xFFFFF4B8 (PIOA), 0xFFFFF6B8 (PIOB), 0xFFFFF8B8 (PIOC), 0xFFFFFAB8 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: IO Line Index

Selects the IO event type triggering an interrupt.

0: The interrupt source is a both-edge detection event.

1: The interrupt source is described by the registers PIO_ELSR and PIO_FRLHSR.

© 2017 Microchip Technology Inc.

22.6.39 PIO Edge Select Register

Name:PIO_ESR

Address:0xFFFFF4C0 (PIOA), 0xFFFFF6C0 (PIOB), 0xFFFFF8C0 (PIOC), 0xFFFFFAC0 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Edge Interrupt Selection

0: No effect.

1: The interrupt source is an edge-detection event.

22.6.40 PIO Level Select Register

Name:PIO_LSR

Address:0xFFFFF4C4 (PIOA), 0xFFFFF6C4 (PIOB), 0xFFFFF8C4 (PIOC), 0xFFFFFAC4 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Level Interrupt Selection

0: No effect.

1: The interrupt source is a level-detection event.

© 2017 Microchip Technology Inc.

22.6.41 PIO Edge/Level Status Register

Name:PIO_ELSR

Address:0xFFFFF4C8 (PIOA), 0xFFFFF6C8 (PIOB), 0xFFFFF8C8 (PIOC), 0xFFFFFAC8 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Edge/Level Interrupt Source Selection

0: The interrupt source is an edge-detection event.

1: The interrupt source is a level-detection event.

22.6.42 PIO Falling Edge/Low-Level Select Register

Name:PIO_FELLSR

Address:0xFFFFF4D0 (PIOA), 0xFFFFF6D0 (PIOB), 0xFFFFF8D0 (PIOC), 0xFFFFFAD0 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Falling Edge/Low-Level Interrupt Selection

0: No effect.

1: The interrupt source is set to a falling edge detection or low-level detection event, depending on PIO_ELSR.

© 2017 Microchip Technology Inc.

22.6.43 PIO Rising Edge/High-Level Select Register

Name:PIO_REHLSR

Address:0xFFFFF4D4 (PIOA), 0xFFFFF6D4 (PIOB), 0xFFFFF8D4 (PIOC), 0xFFFFFAD4 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Rising Edge/High-Level Interrupt Selection

0: No effect.

1: The interrupt source is set to a rising edge detection or high-level detection event, depending on PIO_ELSR.

22.6.44 PIO Fall/Rise - Low/High Status Register

Name:PIO_FRLHSR

Address:0xFFFFF4D8 (PIOA), 0xFFFFF6D8 (PIOB), 0xFFFFF8D8 (PIOC), 0xFFFFFAD8 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

P0–P31: Edge/Level Interrupt Source Selection

0: The interrupt source is a falling edge detection (if PIO_ELSR = 0) or low-level detection event (if PIO_ELSR = 1).

1: The interrupt source is a rising edge detection (if PIO_ELSR = 0) or high-level detection event (if PIO_ELSR = 1).

© 2017 Microchip Technology Inc.

22.6.45 PIO Write Protection Mode Register

Name:PIO_WPMR

Address:0xFFFFF4E4 (PIOA), 0xFFFFF6E4 (PIOB), 0xFFFFF8E4 (PIOC), 0xFFFFFAE4 (PIOD)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

WPEN: Write Protection Enable

0: Disables the write protection if WPKEY corresponds to 0x50494F (“PIO” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x50494F (“PIO” in ASCII).

See Section 22.5.15 “Register Write Protection” for the list of registers that can be protected.

WPKEY: Write Protection Key

Value	Name	Description
0x50494F	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

22.6.46 PIO Write Protection Status Register

Name:PIO_WPSR

Address:0xFFFFF4E8 (PIOA), 0xFFFFF6E8 (PIOB), 0xFFFFF8E8 (PIOC), 0xFFFFFAE8 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
WPVSRC							
15	14	13	12	11	10	9	8
WPVSRC							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

WPVS: Write Protection Violation Status

0: No write protection violation has occurred since the last read of the PIO_WPSR.

1: A write protection violation has occurred since the last read of the PIO_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

WPVSRC: Write Protection Violation Source

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

© 2017 Microchip Technology Inc.

22.6.47 PIO Schmitt Trigger Register

Name:PIO_SCHMITT

Address:0xFFFFF500 (PIOA), 0xFFFFF700 (PIOB), 0xFFFFF900 (PIOC), 0xFFFFFB00 (PIOD)

Access: Read/Write

31	30	29	28	27	26	25	24
SCHMITT31	SCHMITT30	SCHMITT29	SCHMITT28	SCHMITT27	SCHMITT26	SCHMITT25	SCHMITT24
23	22	21	20	19	18	17	16
SCHMITT23	SCHMITT22	SCHMITT21	SCHMITT20	SCHMITT19	SCHMITT18	SCHMITT17	SCHMITT16
15	14	13	12	11	10	9	8
SCHMITT15	SCHMITT14	SCHMITT13	SCHMITT12	SCHMITT11	SCHMITT10	SCHMITT9	SCHMITT8
7	6	5	4	3	2	1	0
SCHMITT7	SCHMITT6	SCHMITT5	SCHMITT4	SCHMITT3	SCHMITT2	SCHMITT1	SCHMITT0

SCHMITTx [x=0..31]: Schmitt Trigger Control

0: Schmitt trigger is enabled.

1: Schmitt trigger is disabled.

22.6.48 PIO I/O Delay Register

Name:PIO_DELAYR

Address:0xFFFFF510 (PIOA), 0xFFFFF710 (PIOB), 0xFFFFF910 (PIOC), 0xFFFFFB10 (PIOD)

Access: Read/Write

31	30	29	28	27	26	25	24
Delay7				Delay6			
23	22	21	20	19	18	17	16
Delay5				Delay4			
15	14	13	12	11	10	9	8
Delay3				Delay2			
7	6	5	4	3	2	1	0
Delay1				Delay0			

Delayx [x=0..7]: Delay Control for Simultaneous Switch Reduction

Gives the number of elements in the delay line associated to pad x.

© 2017 Microchip Technology Inc.

22.6.49 PIO I/O Drive Register 1

Name:PIO_DRIVER1

Address:0xFFFFF514 (PIOA), 0xFFFFF714 (PIOB), 0xFFFFF914 (PIOC), 0xFFFFFB14 (PIOD)

Access: Read/Write

31	30	29	28	27	26	25	24
LINE15		LINE14		LINE13		LINE12	
23	22	21	20	19	18	17	16
LINE11		LINE10		LINE9		LINE8	
15	14	13	12	11	10	9	8
LINE7		LINE6		LINE5		LINE4	
7	6	5	4	3	2	1	0
LINE3		LINE2		LINE1		LINE0	

LINEx [x=0..15]: Drive of PIO Line x

Value	Name	Description
0	HI_DRIVE	High drive
1	ME_DRIVE	Medium drive
2	LO_DRIVE	Low drive
3	–	Reserved

22.6.50 PIO I/O Drive Register 2

Name:PIO_DRIVER2

Address:0xFFFFF518 (PIOA), 0xFFFFF718 (PIOB), 0xFFFFF918 (PIOC), 0xFFFFFB18 (PIOD)

Access: Read/Write

31	30	29	28	27	26	25	24
LINE31		LINE30		LINE29		LINE28	
23	22	21	20	19	18	17	16
LINE27		LINE26		LINE25		LINE24	
15	14	13	12	11	10	9	8
LINE23		LINE22		LINE21		LINE20	
7	6	5	4	3	2	1	0
LINE19		LINE18		LINE17		LINE16	

LINEx [x=16..31]: Drive of PIO line x

Value	Name	Description
0	HI_DRIVE	High drive
1	ME_DRIVE	Medium drive
2	LO_DRIVE	Low drive
3	–	Reserved

© 2017 Microchip Technology Inc.

23. Debug Unit (DBGU)

23.1 Description

The Debug Unit (DBGU) provides a single entry point from the processor for access to all the debug capabilities of Microchip Arm-based systems.

The Debug Unit features a two-pin UART that can be used for several debug and trace purposes and offers an ideal medium for in-situ programming solutions and debug monitor communications. The Debug Unit two-pin UART can be used stand-alone for general purpose serial communication. Moreover, the association with DMA controller channels permits packet handling for these tasks with processor time reduced to a minimum.

The Debug Unit also makes the Debug Communication Channel (DCC) signals provided by the In-circuit Emulator of the Arm processor visible to the software. These signals indicate the status of the DCC read and write registers and generate an interrupt to the Arm processor, making possible the handling of the DCC under interrupt control.

Chip Identifier registers permit recognition of the device and its revision. These registers indicate the sizes and types of the on-chip memories, as well as the set of embedded peripherals.

Finally, the Debug Unit features a Force NTRST capability that enables the software to decide whether to prevent access to the system via the In-circuit Emulator. This permits protection of the code, stored in ROM.

23.2 Embedded Characteristics

- System Peripheral to Facilitate Debug of Microchip Arm-based Systems
- Composed of Four Functions
  - Two-pin UART
  - Debug Communication Channel (DCC) Support
  - Chip ID Registers
  - ICE Access Prevention
- Two-pin UART
  - Implemented Features are USART Compatible
  - Independent Receiver and Transmitter with a Common Programmable Baud Rate Generator
  - Even, Odd, Mark or Space Parity Generation
  - Parity, Framing and Overrun Error Detection
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
  - Interrupt Generation
  - Support for Two DMA Channels with Connection to Receiver and Transmitter
- Debug Communication Channel Support
  - Offers Visibility of COMMRX and COMMTX Signals from the Arm Processor
  - Interrupt Generation
- Chip ID Registers
  - Identification of the Device Revision, Sizes of the Embedded Memories, Set of Peripherals
- ICE Access Prevention
  - Enables Software to Prevent System Access Through the Arm Processor’s ICE
  - Prevention is Made by Asserting the NTRST Line of the Arm Processor’s ICE

© 2017 Microchip Technology Inc.

23.3 Block Diagram

Figure 23-1: Debug Unit Functional Block Diagram

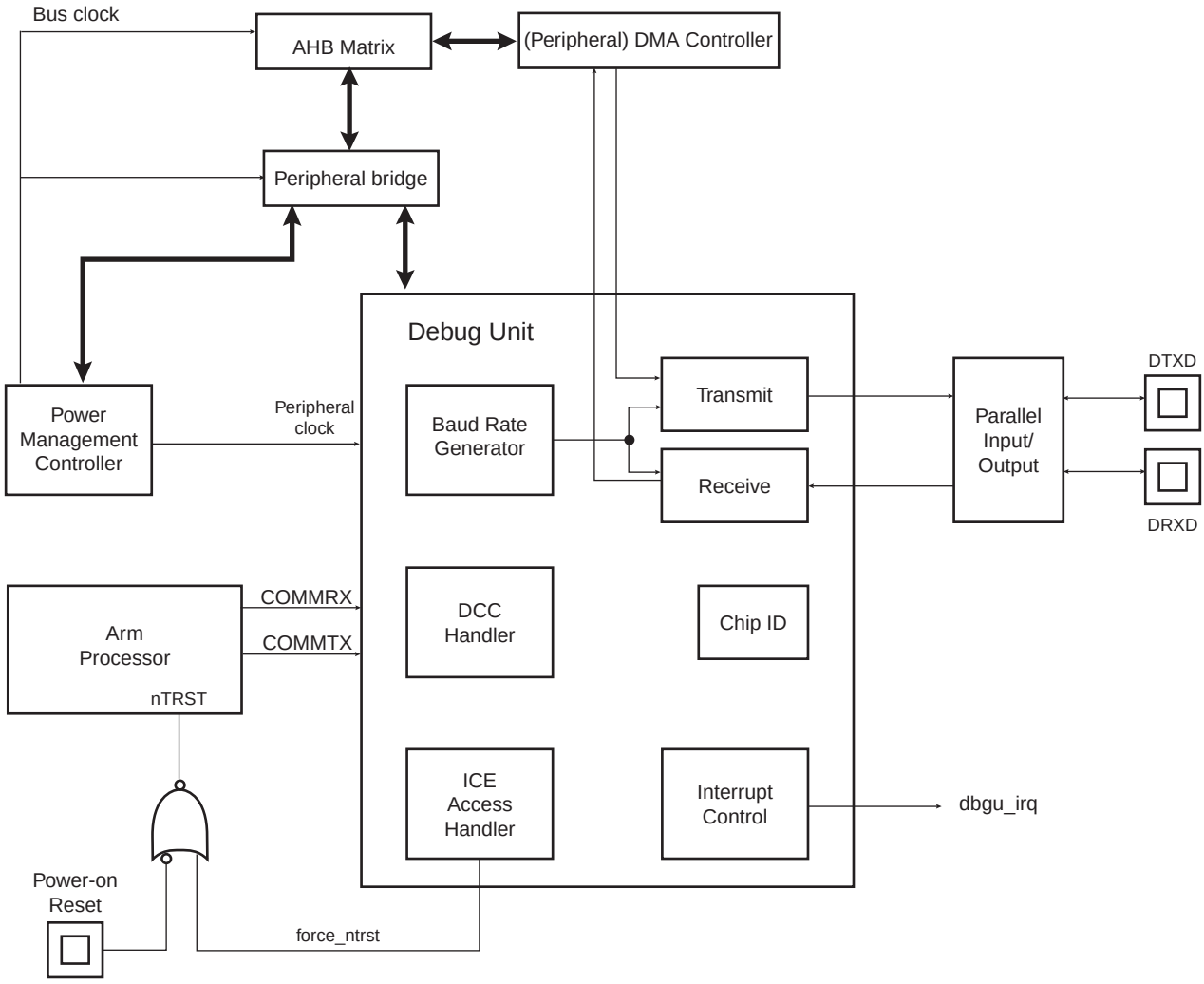
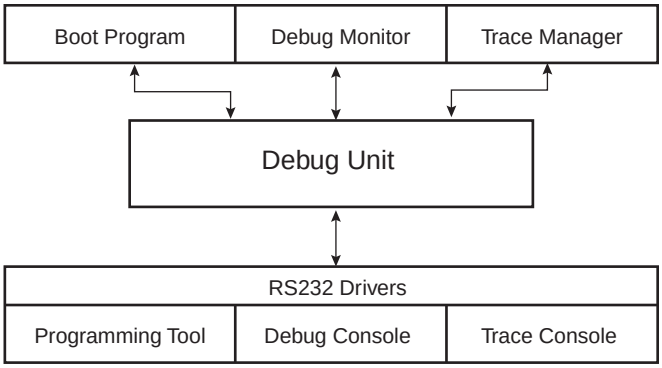


Table 23-1: Debug Unit Pin Description

Pin Name	Description	Type
DRXD	Debug Receive Data	Input
DTXD	Debug Transmit Data	Output

© 2017 Microchip Technology Inc.

Figure 23-2: Debug Unit Application Example



23.4 Product Dependencies

23.4.1 I/O Lines

Depending on product integration, the Debug Unit pins may be multiplexed with PIO lines. In this case, the programmer must first configure the corresponding PIO Controller to enable I/O lines operations of the Debug Unit.

Table 23-2: I/O Lines

Instance	Signal	I/O Line	Peripheral
DBGU	DRXD	PA9	A
DBGU	DTXD	PA10	A

23.4.2 Power Management

Depending on product integration, the Debug Unit clock may be controllable through the Power Management Controller. In this case, the programmer must first configure the PMC to enable the Debug Unit clock. Usually, the peripheral identifier used for this purpose is 1.

23.4.3 Interrupt Source

Depending on product integration, the Debug Unit interrupt line is connected to one of the interrupt sources of the Advanced Interrupt Controller. Interrupt handling requires programming of the AIC before configuring the Debug Unit. Usually, the Debug Unit interrupt line connects to the interrupt source 1 of the AIC, which may be shared with the real-time clock, the system timer interrupt lines and other system peripheral interrupts, as shown in Figure 23-1. This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

23.5 UART Operations

The Debug Unit operates as a UART, (asynchronous mode only) and supports only 8-bit character handling (with parity). It has no clock pin.

The Debug Unit's UART is made up of a receiver and a transmitter that operate independently, and a common baud rate generator. Receiver timeout and transmitter time guard are not implemented. However, all the implemented features are compatible with those of a standard USART.

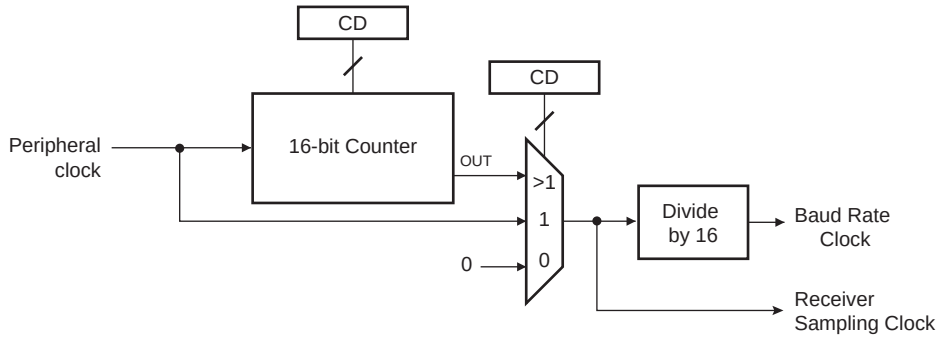
23.5.1 Baud Rate Generator

The baud rate generator provides the bit period clock named baud rate clock to both the receiver and the transmitter.

The baud rate clock is the peripheral clock divided by 16 times the value (CD) written in the Debug Unit Baud Rate Generator register (DBGU_BRGR). If DBGU_BRGR is set to 0, the baud rate clock is disabled and the Debug Unit's UART remains inactive. The maximum allowable baud rate is peripheral clock divided by 16. The minimum allowable baud rate is peripheral clock divided by (16 x 65536).

$$\text{Baud Rate} = \frac{f_{\text{peripheral clock}}}{16 \times \text{CD}}$$

Figure 23-3: Baud Rate Generator



23.5.2 Receiver

23.5.2.1 Receiver Reset, Enable and Disable

After device reset, the Debug Unit receiver is disabled and must be enabled before being used. The receiver can be enabled by writing one to the RXEN bit in the Debug Unit Control register (DBGU_CR). At this command, the receiver starts looking for a start bit.

The programmer can disable the receiver by writing a one to the RXDIS bit in the DBGU_CR. If the receiver is waiting for a start bit, it is immediately stopped. However, if the receiver has already detected a start bit and is receiving the data, it waits for the stop bit before actually stopping its operation.

The programmer can also put the receiver in its reset state by writing a one to the RSTRX bit in the DBGU_CR. In doing so, the receiver immediately stops its current operations and is disabled, whatever its current state. If RSTRX is applied when data is being processed, this data is lost.

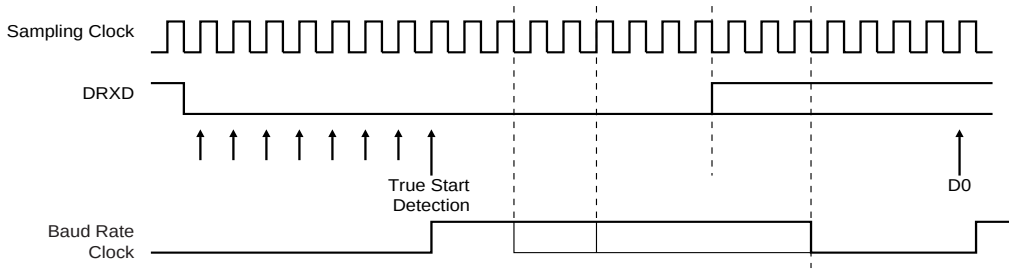
23.5.2.2 Start Detection and Data Sampling

The Debug Unit only supports asynchronous operations, and this affects only its receiver. The Debug Unit receiver detects the start of a received character by sampling the DRXD signal until it detects a valid start bit. A low level (space) on DRXD is interpreted as a valid start bit if it is detected for more than 7 cycles of the sampling clock, which is 16 times the baud rate. Hence, a space that is longer than 7/16 of the bit period is detected as a valid start bit. A space which is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

When a valid start bit has been detected, the receiver samples the DRXD at the theoretical midpoint of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock (1-bit period) so the bit sampling point is eight cycles (0.5-bit period) after the start of the bit. The first sampling point is therefore 24 cycles (1.5-bit periods) after the falling edge of the start bit was detected.

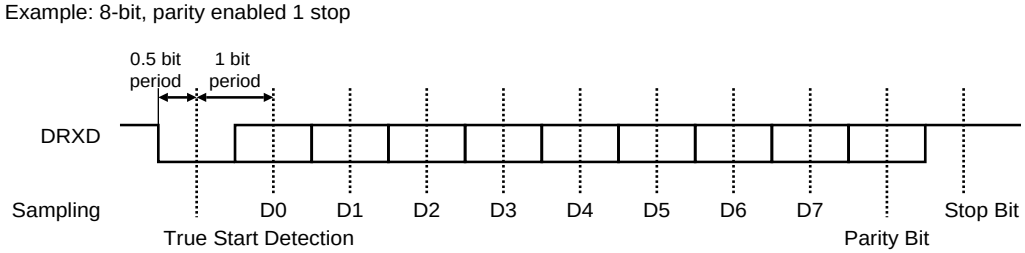
Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

Figure 23-4: Start Bit Detection



© 2017 Microchip Technology Inc.

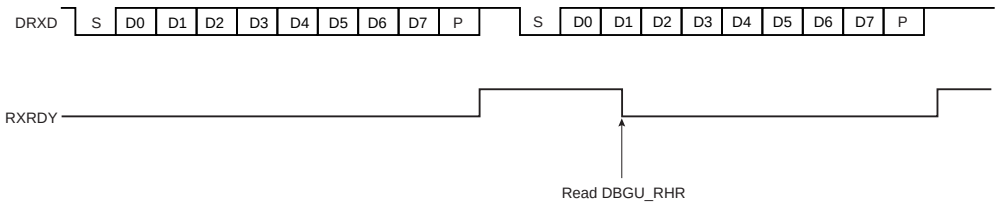
Figure 23-5: Character Reception



23.5.2.3 Receiver Ready

When a complete character is received, it is transferred to the Debug Unit Receive Holding register (DBGU_RHR) and the RXRDY status bit in the Debug Unit Status register (DBGU_SR) is set. The bit RXRDY is automatically cleared when the receive holding register DBGU_RHR is read.

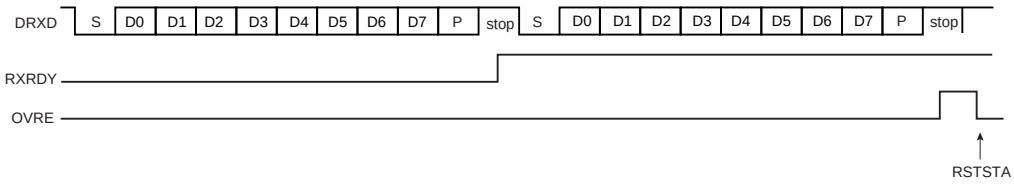
Figure 23-6: Receiver Ready



23.5.2.4 Receiver Overrun

If DBGU_RHR has not been read by the software (or the Peripheral Data Controller or DMA Controller) since the last transfer, the RXRDY bit is still set and a new character is received, the OVRE status bit in DBGU_SR is set. OVRE is cleared when the software writes a one to the bit RSTSTA (Reset Status) in the DBGU_CR.

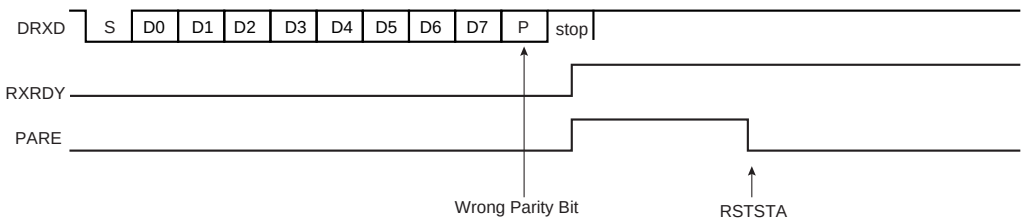
Figure 23-7: Receiver Overrun



23.5.2.5 Parity Error

Each time a character is received, the receiver calculates the parity of the received data bits, in accordance with the field PAR in the Debug Unit Mode register (DBGU_MR). It then compares the result with the received parity bit. If different, the parity error bit PARE in DBGU_SR is set at the same time as the RXRDY is set. The parity bit is cleared when a one is written to the bit RSTSTA (Reset Status) in the DBGU_CR. If a new character is received before the reset status command is written, the PARE bit remains at 1.

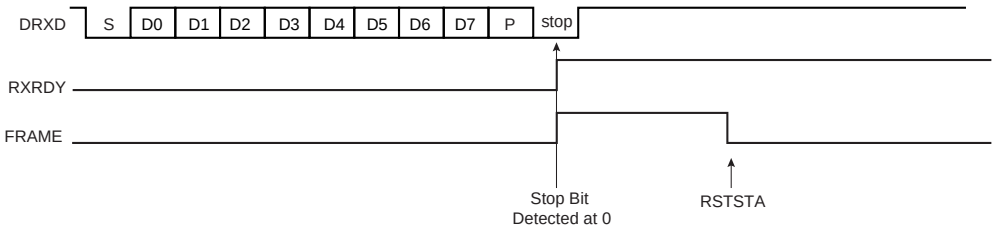
Figure 23-8: Parity Error



23.5.2.6 Receiver Framing Error

When a start bit is detected, it generates a character reception when all the data bits have been sampled. The stop bit is also sampled and when it is detected at 0, the FRAME (Framing Error) bit in DBGU_SR is set at the same time as the RXRDY bit is set. The bit FRAME remains high until a one is written to the RSTSTA bit in the DBGU_CR.

Figure 23-9: Receiver Framing Error



23.5.3 Transmitter

23.5.3.1 Transmitter Reset, Enable and Disable

After device reset, the Debug Unit transmitter is disabled and it must be enabled before being used. The transmitter is enabled by writing a one to the TXEN bit in DBGU_CR. From this command, the transmitter waits for a character to be written in the Transmit Holding register (DBGU_THR) before actually starting the transmission.

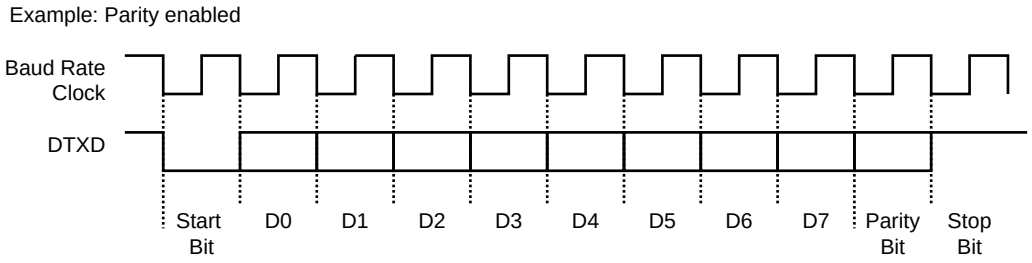
The programmer can disable the transmitter by writing a one to the TXDIS bit in the DBGU_CR. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters are completed before the transmitter is actually stopped.

The programmer can also put the transmitter in its reset state by writing a one to the RSTTX bit in the DBGU_CR. This immediately stops the transmitter, whether or not it is processing characters.

23.5.3.2 Transmit Format

The Debug Unit transmitter drives the pin DTXD at the baud rate clock speed. The line is driven depending on the format defined in DBGU_MR and the data stored in the Shift Register. One start bit at level 0, then the 8 data bits, from the lowest to the highest bit, one optional parity bit and one stop bit at 1 are consecutively shifted out as shown on the following figure. The field PARE in DBGU_MR defines whether or not a parity bit is shifted out. When a parity bit is enabled, it can be selected between an odd parity, an even parity, or a fixed space or mark bit.

Figure 23-10: Character Transmission



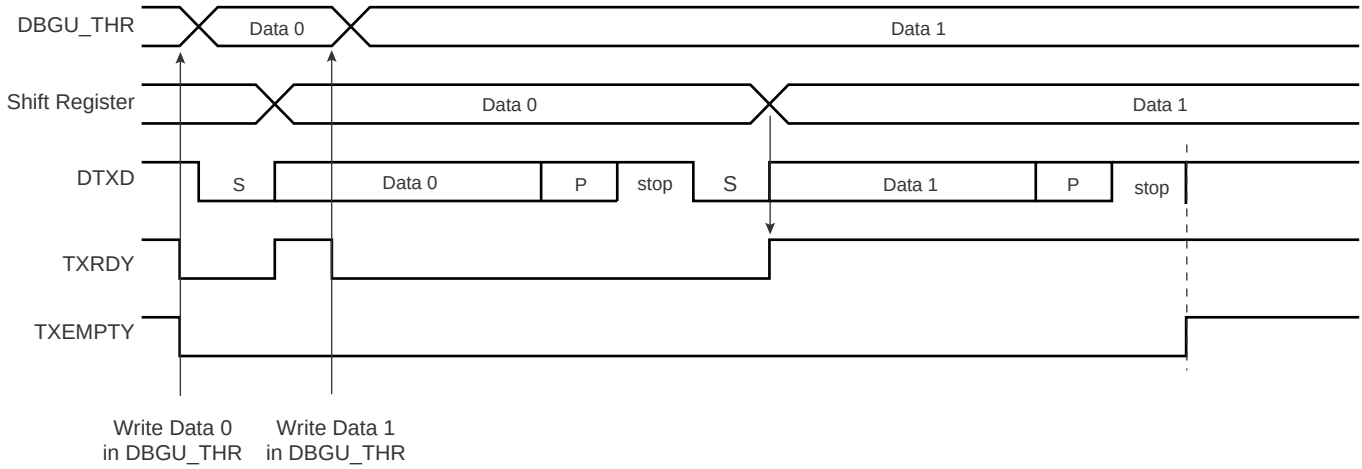
23.5.3.3 Transmitter Control

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in DBGU_SR. The transmission starts when the programmer writes in DBGU_THR, and after the written character is transferred from DBGU_THR to the Shift Register. The bit TXRDY remains high until a second character is written in DBGU_THR. As soon as the first character is completed, the last character written in DBGU_THR is transferred into the shift register and TXRDY rises again, showing that the holding register is empty.

When both the Shift Register and the DBGU_THR are empty, i.e., all the characters written in DBGU_THR have been processed, the bit TXEMPTY rises after the last stop bit has been completed.

© 2017 Microchip Technology Inc.

Figure 23-11: Transmitter Control



23.5.4 DMA Support

Both the receiver and the transmitter of the Debug Unit's UART are connected to a DMA Controller (DMAC) channel.

The DMA Controller channels are programmed via registers that are mapped within the DMAC user interface.

23.5.5 Test Modes

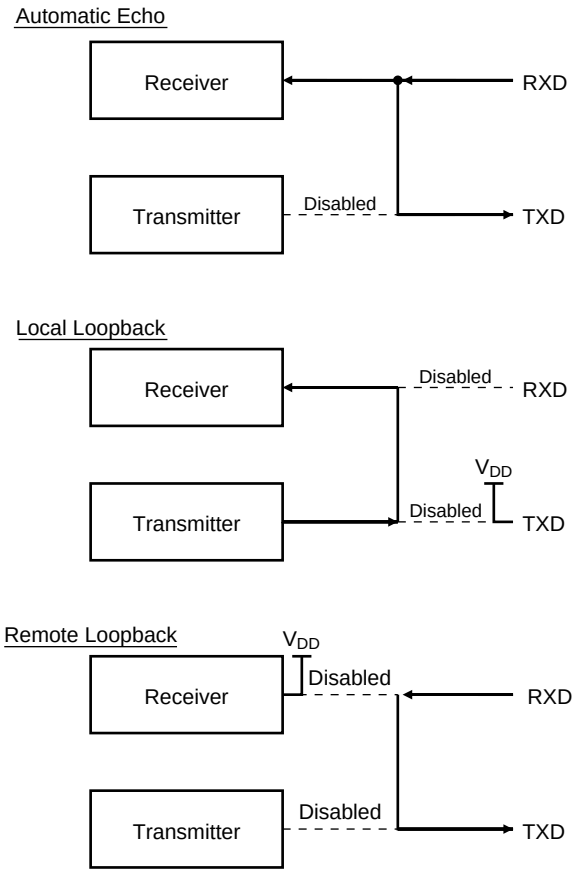
The Debug Unit supports three tests modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in DBGU_MR.

The Automatic Echo mode allows bit-by-bit retransmission. When a bit is received on the DRXD line, it is sent to the DTXD line. The transmitter operates normally, but has no effect on the DTXD line.

The Local Loopback mode allows the transmitted characters to be received. DTXD and DRXD pins are not used and the output of the transmitter is internally connected to the input of the receiver. The DRXD pin level has no effect and the DTXD line is held high, as in idle state.

The Remote Loopback mode directly connects the DRXD pin to the DTXD line. The transmitter and the receiver are disabled and have no effect. This mode allows a bit-by-bit retransmission.

Figure 23-12: Test Modes



23.5.6 Debug Communication Channel Support

The Debug Unit handles the signals COMMRX and COMMTX that come from the Debug Communication Channel of the Arm Processor and are driven by the In-circuit Emulator.

The Debug Communication Channel contains two registers that are accessible through the ICE Breaker on the JTAG side and through the coprocessor 0 on the Arm Processor side.

As a reminder, the following instructions are used to read and write the Debug Communication Channel:

```
MRC             p14, 0, Rd, c1, c0, 0
```

Returns the debug communication data read register into Rd

```
MCR             p14, 0, Rd, c1, c0, 0
```

Writes the value in Rd to the debug communication data write register.

The bits COMMRX and COMMTX, which indicate, respectively, that the read register has been written by the debugger but not yet read by the processor, and that the write register has been written by the processor and not yet read by the debugger, are wired on the two highest bits of DBGU_SR. These bits can generate an interrupt. This feature permits handling under interrupt a debug link between a debug monitor running on the target system and a debugger.

23.5.7 Chip Identifier

The Debug Unit features two chip identifier registers, Debug Unit Chip ID register (DBGU_CIDR) and Debug Unit Extension ID register (DBGU_EXID). Both registers contain a hard-wired value that is read-only.

The first register (DBGU_CIDR) contains the following fields:

- EXT: shows the use of the extension identifier register
- NVPTYP and NVPSIZ: identifies the type of embedded non-volatile memory and its size

© 2017 Microchip Technology Inc.

- ARCH: identifies the set of embedded peripherals
- SRAMSIZ: indicates the size of the embedded SRAM
- EPROC: indicates the embedded Arm processor
- VERSION: gives the revision of the silicon

The second register (DBGU_EXID) is device-dependent and is read as 0 if the bit EXT is 0 in DBGU_CIDR.

23.5.8 ICE Access Prevention

The Debug Unit allows blockage of access to the system through the Arm processor's ICE interface. This feature is implemented via the Debug Unit Force NTRST register (DBGU_FNR), that allows assertion of the NTRST signal of the ICE Interface. Writing the bit FNTRST (Force NTRST) to 1 in this register prevents any activity on the TAP controller.

On standard devices, the bit FNTRST resets to 0 and thus does not prevent ICE access.

This feature is especially useful on custom ROM devices for customers who do not want their on-chip code to be visible.

23.6 Debug Unit (DBGU) User Interface

Table 23-3: Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Control Register	DBGU_CR	Write-only	–
0x0004	Mode Register	DBGU_MR	Read/Write	0x0
0x0008	Interrupt Enable Register	DBGU_IER	Write-only	–
0x000C	Interrupt Disable Register	DBGU_IDR	Write-only	–
0x0010	Interrupt Mask Register	DBGU_IMR	Read-only	0x0
0x0014	Status Register	DBGU_SR	Read-only	–
0x0018	Receive Holding Register	DBGU_RHR	Read-only	0x0
0x001C	Transmit Holding Register	DBGU_THR	Write-only	–
0x0020	Baud Rate Generator Register	DBGU_BRGR	Read/Write	0x0
0x0024 - 0x003C	Reserved	–	–	–
0x0040	Chip ID Register	DBGU_CIDR	Read-only	–
0x0044	Chip ID Extension Register	DBGU_EXID	Read-only	–
0x0048	Force NTRST Register	DBGU_FNR	Read/Write	0x0
0x004C - 0x00FC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

23.6.1 Debug Unit Control Register

Name:DBGU_CR

Address:0xFFFFF200

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RSTSTA
7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

RSTRX: Reset Receiver

0: No effect.

1: The receiver logic is reset and disabled. If a character is being received, the reception is aborted.

RSTTX: Reset Transmitter

0: No effect.

1: The transmitter logic is reset and disabled. If a character is being transmitted, the transmission is aborted.

RXEN: Receiver Enable

0: No effect.

1: The receiver is enabled if RXDIS is 0.

RXDIS: Receiver Disable

0: No effect.

1: The receiver is disabled. If a character is being processed and RSTRX is not set, the character is completed before the receiver is stopped.

TXEN: Transmitter Enable

0: No effect.

1: The transmitter is enabled if TXDIS is 0.

TXDIS: Transmitter Disable

0: No effect.

1: The transmitter is disabled. If a character is being processed and a character has been written in the DBGU_THR and RSTTX is not set, both characters are completed before the transmitter is stopped.

RSTSTA: Reset Status Bits

0: No effect.

1: Resets the status bits PARE, FRAME and OVRE in DBGU_SR.

23.6.2 Debug Unit Mode Register

Name:DBGU_MR

Address:0xFFFFF204

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CHMODE		–	–	PAR			–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

PAR: Parity Type

Value	Name	Description
0b000	EVEN	Even Parity
0b001	ODD	Odd Parity
0b010	SPACE	Space: Parity forced to 0
0b011	MARK	Mark: Parity forced to 1
0b1xx	NONE	No Parity

CHMODE: Channel Mode

Value	Name	Description
0b00	NORM	Normal Mode
0b01	AUTO	Automatic Echo
0b10	LOCLOOP	Local Loopback
0b11	REMLOOP	Remote Loopback

© 2017 Microchip Technology Inc.

23.6.3 Debug Unit Interrupt Enable Register

Name:DBGU_IER

Address:0xFFFFF208

Access: Write-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

RXRDY: Enable RXRDY Interrupt

TXRDY: Enable TXRDY Interrupt

OVRE: Enable Overrun Error Interrupt

FRAME: Enable Framing Error Interrupt

PARE: Enable Parity Error Interrupt

TXEMPTY: Enable TXEMPTY Interrupt

COMMTX: Enable COMMTX (from Arm) Interrupt

COMMRX: Enable COMMRX (from Arm) Interrupt

0: No effect.

1: Enables the corresponding interrupt.

23.6.4 Debug Unit Interrupt Disable Register

Name:DBGU_IDR

Address:0xFFFFF20C

Access: Write-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

RXRDY: Disable RXRDY Interrupt

TXRDY: Disable TXRDY Interrupt

OVRE: Disable Overrun Error Interrupt

FRAME: Disable Framing Error Interrupt

PARE: Disable Parity Error Interrupt

TXEMPTY: Disable TXEMPTY Interrupt

COMMTX: Disable COMMTX (from Arm) Interrupt

COMMRX: Disable COMMRX (from Arm) Interrupt

0: No effect.

1: Disables the corresponding interrupt.

© 2017 Microchip Technology Inc.

23.6.5 Debug Unit Interrupt Mask Register

Name:DBGU_IMR

Address:0xFFFFF210

Access: Read-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

RXRDY: Mask RXRDY Interrupt

TXRDY: Disable TXRDY Interrupt

OVRE: Mask Overrun Error Interrupt

FRAME: Mask Framing Error Interrupt

PARE: Mask Parity Error Interrupt

TXEMPTY: Mask TXEMPTY Interrupt

COMMTX: Mask COMMTX Interrupt

COMMRX: Mask COMMRX Interrupt

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

23.6.6 Debug Unit Status Register

Name:DBGU_SR

Address:0xFFFFF214

Access: Read-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

RXRDY: Receiver Ready

0: No character has been received since the last read of the DBGU_RHR, or the receiver is disabled.

1: At least one complete character has been received, transferred to DBGU_RHR and not yet read.

TXRDY: Transmitter Ready

0: A character has been written to DBGU_THR and not yet transferred to the Shift Register, or the transmitter is disabled.

1: There is no character written to DBGU_THR not yet transferred to the Shift Register.

OVRE: Overrun Error

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

FRAME: Framing Error

0: No framing error has occurred since the last RSTSTA.

1: At least one framing error has occurred since the last RSTSTA.

PARE: Parity Error

0: No parity error has occurred since the last RSTSTA.

1: At least one parity error has occurred since the last RSTSTA.

TXEMPTY: Transmitter Empty

0: There are characters in DBGU_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1: There are no characters in DBGU_THR and there are no characters being processed by the transmitter.

COMMTX: Debug Communication Channel Write Status

0: COMMTX from the Arm processor is inactive.

1: COMMTX from the Arm processor is active.

COMMRX: Debug Communication Channel Read Status

0: COMMRX from the Arm processor is inactive.

1: COMMRX from the Arm processor is active.

© 2017 Microchip Technology Inc.

23.6.7 Debug Unit Receive Holding Register

Name:DBGU_RHR

Address:0xFFFFF218

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
RXCHR							

RXCHR: Received Character

Last received character if RXRDY is set.

23.6.8 Debug Unit Transmit Holding Register

Name:DBGU_THR

Address:0xFFFFF21C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
TXCHR							

TXCHR: Character to be Transmitted

Next character to be transmitted after the current character if TXRDY is not set.

© 2017 Microchip Technology Inc.

23.6.9 Debug Unit Baud Rate Generator Register

Name:DBGU_BRGR

Address:0xFFFFF220

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CD							
7	6	5	4	3	2	1	0
CD							

CD: Clock Divisor

Value	Name	Description
0	DISABLED	DBGU Disabled
1	MCK	Peripheral clock
2 to 65535	–	Peripheral clock/ (CD x 16)

23.6.10 Debug Unit Chip ID Register

Name:DBGU_CIDR

Address:0xFFFFF240

Access: Read-only

31	30	29	28	27	26	25	24
EXT	NVPTYP			ARCH			
23	22	21	20	19	18	17	16
ARCH				SRAMSIZ			
15	14	13	12	11	10	9	8
NVPSIZ2				NVPSIZ			
7	6	5	4	3	2	1	0
EPROC			VERSION				

VERSION: Version of the Device

Values depend on the version of the device.

EPROC: Embedded Processor

Value	Name	Description
1	ARM946ES	Arm946ES
2	ARM7TDMI	Arm7TDMI
3	CM3	Cortex-M3
4	ARM920T	Arm920T
5	ARM926EJS	Arm926EJ-S
6	CA5	Cortex-A5

NVPSIZ: Nonvolatile Program Memory Size

Value	Name	Description
0	NONE	None
1	8K	8 Kbytes
2	16K	16 Kbytes
3	32K	32 Kbytes
4	–	Reserved
5	64K	64 Kbytes
6	–	Reserved
7	128K	128 Kbytes
8	–	Reserved
9	256K	256 Kbytes
10	512K	512 Kbytes
11	–	Reserved
12	1024K	1024 Kbytes
13	–	Reserved
14	2048K	2048 Kbytes
15	–	Reserved

© 2017 Microchip Technology Inc.

NVPSIZ2: Second Nonvolatile Program Memory Size

Value	Name	Description
0	NONE	None
1	8K	8 Kbytes
2	16K	16 Kbytes
3	32K	32 Kbytes
4	–	Reserved
5	64K	64 Kbytes
6		Reserved
7	128K	128 Kbytes
8	–	Reserved
9	256K	256 Kbytes
10	512K	512 Kbytes
11	–	Reserved
12	1024K	1024 Kbytes
13	–	Reserved
14	2048K	2048 Kbytes
15	–	Reserved

SRAMSIZ: Internal SRAM Size

Value	Name	Description
0	–	Reserved
1	1K	1 Kbytes
2	2K	2 Kbytes
3	6K	6 Kbytes
4	112K	112 Kbytes
5	4K	4 Kbytes
6	80K	80 Kbytes
7	160K	160 Kbytes
8	8K	8 Kbytes
9	16K	16 Kbytes
10	32K	32 Kbytes
11	64K	64 Kbytes
12	128K	128 Kbytes
13	256K	256 Kbytes
14	96K	96 Kbytes
15	512K	512 Kbytes

© 2017 Microchip Technology Inc.

ARCH: Architecture Identifier

Value	Name	Description
0x19	AT91SAM9xx	AT91SAM9xx Series
0x29	AT91SAM9XExx	AT91SAM9XExx Series
0x34	AT91x34	AT91x34 Series
0x37	CAP7	CAP7 Series
0x39	CAP9	CAP9 Series
0x3B	CAP11	CAP11 Series
0x40	AT91x40	AT91x40 Series
0x42	AT91x42	AT91x42 Series
0x55	AT91x55	AT91x55 Series
0x60	AT91SAM7Axx	AT91SAM7Axx Series
0x61	AT91SAM7AQxx	AT91SAM7AQxx Series
0x63	AT91x63	AT91x63 Series
0x70	AT91SAM7Sxx	AT91SAM7Sxx Series
0x71	AT91SAM7XCxx	AT91SAM7XCxx Series
0x72	AT91SAM7SExx	AT91SAM7SExx Series
0x73	AT91SAM7Lxx	AT91SAM7Lxx Series
0x75	AT91SAM7Xxx	AT91SAM7Xxx Series
0x76	AT91SAM7SLxx	AT91SAM7SLxx Series
0x80	ATSAM3UxC	ATSAM3UxC Series (100-pin version)
0x81	ATSAM3UxE	ATSAM3UxE Series (144-pin version)
0x83	ATSAM3AxC	ATSAM3AxC Series (100-pin version)
0x84	ATSAM3XxC	ATSAM3XxC Series (100-pin version)
0x85	ATSAM3XxE	ATSAM3XxE Series (144-pin version)
0x86	ATSAM3XxG	ATSAM3XxG Series (208/217-pin version)
0x88	ATSAM3SxA	ATSAM3SxA Series (48-pin version)
0x89	ATSAM3SxB	ATSAM3SxB Series (64-pin version)
0x8A	ATSAM3SxC	ATSAM3SxC Series (100-pin version)
0x92	AT91x92	AT91x92 Series
0x93	ATSAM3NxA	ATSAM3NxA Series (48-pin version)
0x94	ATSAM3NxB	ATSAM3NxB Series (64-pin version)
0x95	ATSAM3NxC	ATSAM3NxC Series (100-pin version)
0x98	ATSAM3SDxA	ATSAM3SDxA Series (48-pin version)
0x99	ATSAM3SDxB	ATSAM3SDxB Series (64-pin version)
0x9A	ATSAM3SDxC	ATSAM3SDxC Series (100-pin version)
0xA5	ATSAMA5xx	ATSAMA5xx Series
0xF0	AT75Cxx	AT75Cxx Series

© 2017 Microchip Technology Inc.

NVPTYP: Nonvolatile Program Memory Type

Value	Name	Description
0	ROM	ROM
1	ROMLESS	ROMless or on-chip Flash
4	SRAM	SRAM emulating ROM
2	FLASH	Embedded Flash Memory
3	ROM_FLASH	ROM and Embedded Flash Memory NVPSIZ is ROM size NVPSIZ2 is Flash size

EXT: Extension Flag

0: Chip ID has a single register definition without extension.

1: An extended Chip ID exists.

23.6.11 Debug Unit Chip ID Extension Register

Name:DBGU_EXID

Address:0xFFFFF244

Access: Read-only

31	30	29	28	27	26	25	24
EXID							

23	22	21	20	19	18	17	16
EXID							

15	14	13	12	11	10	9	8
EXID							

7	6	5	4	3	2	1	0
EXID							

EXID: Chip ID Extension

Read as 0 if the bit EXT in DBGU_CIDR is 0.

© 2017 Microchip Technology Inc.

23.6.12 Debug Unit Force NTRST Register

Name: DBGU_FNR

Address:0xFFFFF248

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	FNTRST

FNTRST: Force NTRST

0: NTRST of the Arm processor’s TAP controller is driven by the power_on_reset signal.

1: NTRST of the Arm processor’s TAP controller is held low.

24. Fuse Controller (FUSE)

24.1 Description

The Fuse Controller (FUSE) supports software fuse programming through a 32-bit register, only fuses set to level “1” are programmed.

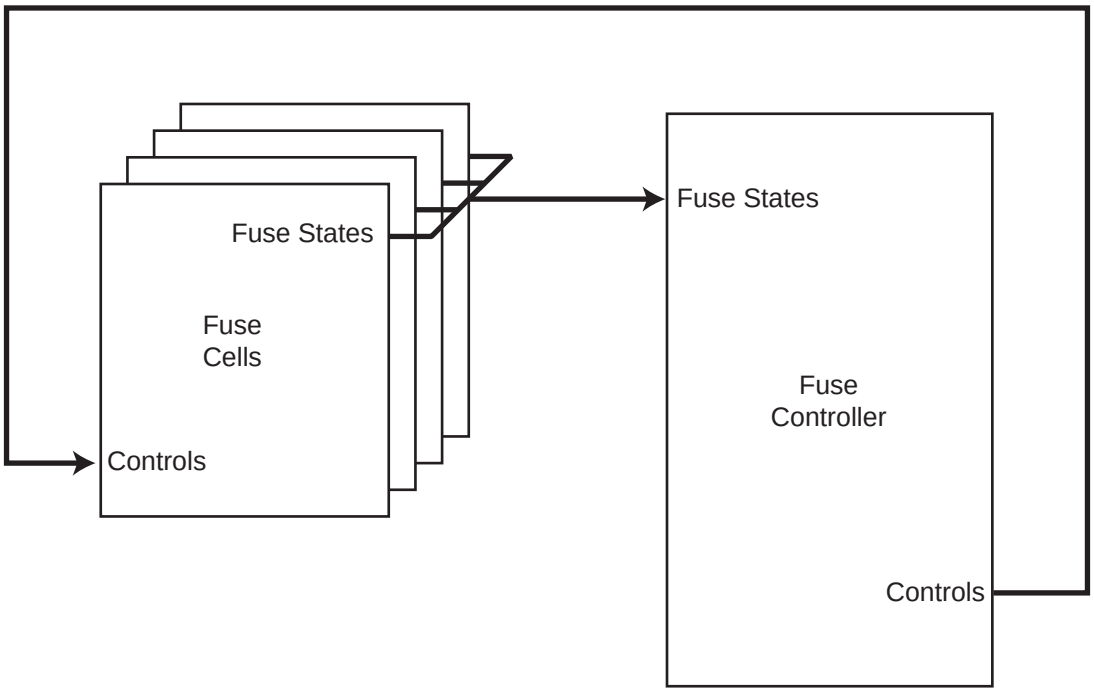
It reads the fuse states on startup and stores them into 32-bit registers. The first 8 Fuse Status registers (FUSE_SRx) can be masked and will read as a value of “0” regardless of the fuse state when masked.

24.2 Embedded Characteristics

- Software Fuse Programming
- User Write Access for Fuse
- Part of Fuse can be Masked After Read

24.3 Block Diagram

Figure 24-1: Fuse Controller Block Diagram



24.4 Functional Description

24.4.1 Fuse Reading

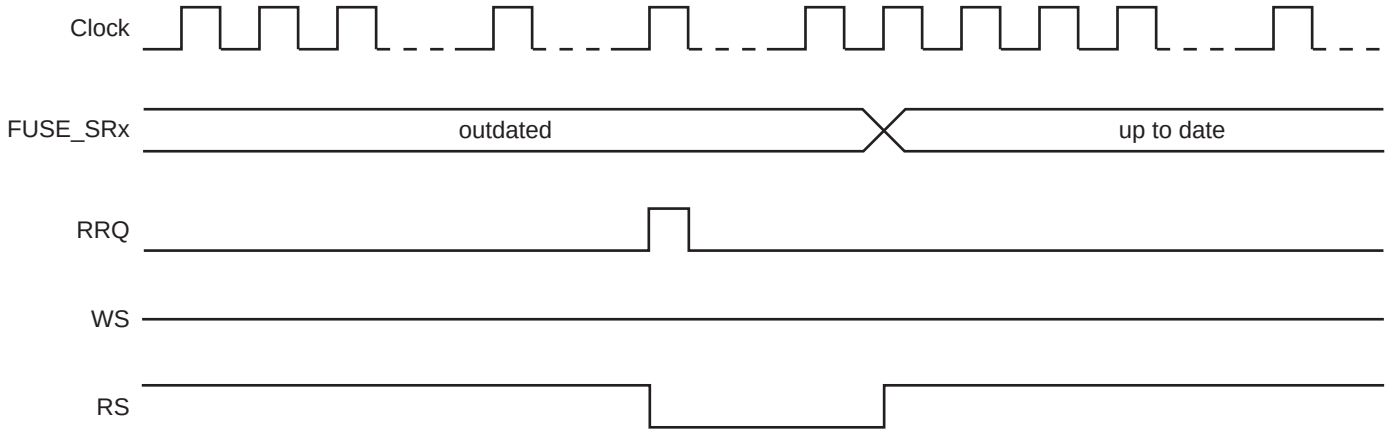
The fuse states are automatically read on CORE startup and are available for reading in the 10 Fuse Status (FUSE_SRx) registers.

The fuse states of bits 31 to 0 will be available at FUSE_SR0, the fuse states of bits 63 to 32 will be available at FUSE_SR1 and so on.

FUSE_SRx registers can be updated manually by using the RRQ bit of the Fuse Control register (FUSE_CR). RS and WS bits of the Fuse Index register (FUSE_IR) must be at level one before issuing the read request.

© 2017 Microchip Technology Inc.

Figure 24-2: Fuse Read



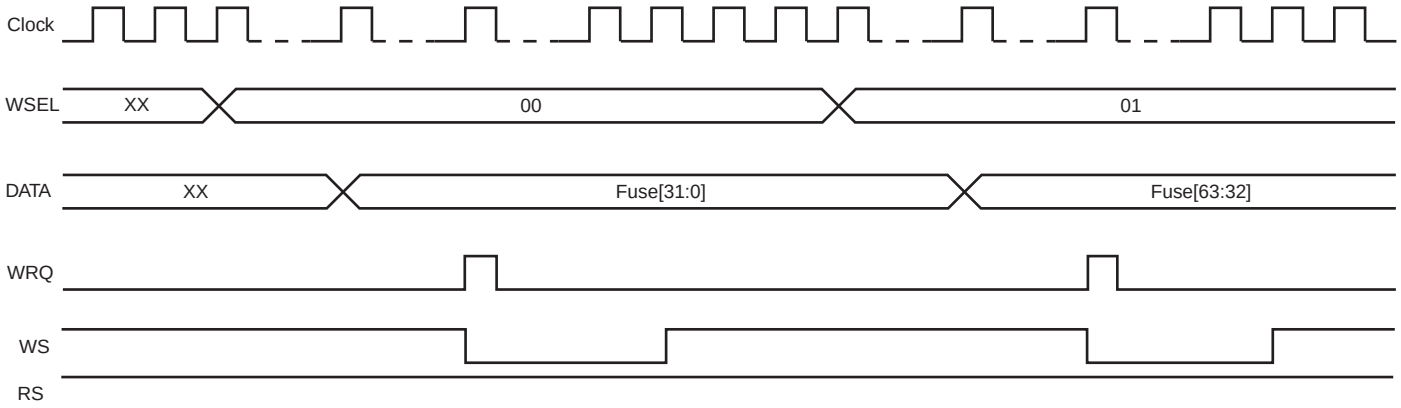
24.4.2 Fuse Programming

All the fuses can be written by software. To program fuses, strictly follow the order of the sequence instructions as provided below:

1. Select the word to write, using the SELW field of the Fuse_Index register (FUSE_IR).
2. Write the word to program in the Fuse_Data register (FUSE_DR).
3. Check that RS and WS bits of the Fuse_Index register are at level one (no read and no write pending).
4. Write the WRQ bit of the Fuse_Control register (FUSE_CR) to begin the fuse programming. The KEY field must be written at the same time with a value 0xFB to make the write request valid. Writing the WRQ bit will clear the WS bit.
5. Check the WS bit of FUSE_SRx, when WS has a value of “1” the fuse write process is over.

Only fuses to be set to level “1” are written.

Figure 24-3: Fuse Write



24.4.3 Fuse Masking

It is possible to mask the first 8 FUSE_SRx registers so that they will be read at a value of “0”, regardless of the fuse state.

To activate fuse masking on the first 8 FUSE_SRx registers, the MSK bit of the Fuse Mode register (FUSE_MR) must be written to level “1”. The MSK bit is write-only. Solely a general reset can disable fuse masking.

24.5 Fuse Controller (FUSE) User Interface

Table 24-1: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Fuse Control Register	FUSE_CR	Write-only	–
0x04	Fuse Mode Register	FUSE_MR	Write-only	–
0x08	Fuse Index Register	FUSE_IR	Read/Write	0x00000000
0x0C	Fuse Data Register	FUSE_DR	Read/Write	–
0x10	Fuse Status Register 0	FUSE_SR0	Read-only	0x00000000
0x14	Fuse Status Register 1	FUSE_SR1	Read-only	0x00000000
...	...	...	...	...
0x34	Fuse Status Register 9	FUSE_SR9	Read-only	0x00000000
0x38–0xDC	Reserved	–	–	–
0xE0–0xFC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

24.5.1 Fuse Control Register

Name:FUSE_CR

Address:0xFFFFDC00

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RRQ	WRQ

WRQ: Write Request

0: No effect.

1: Request the word DATA to be programmed, ignored if KEY field is not filled with 0xFB.

RRQ: Read Request

0: No effect.

1: Requests the fuses to be read and FUSE_SRx registers are then updated. Ignored if KEY field is not filled with 0xFB.

KEY: Password Key

Value	Name	Description
0xFB	PASSWORD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

24.5.2 Fuse Mode Register

Name:FUSE_MR

Address:0xFFFFDC04

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	MSK

MSK: Mask Fuse Status Registers

0: No effect.

1: Masks the first 8 FUSE_SRx registers.

© 2017 Microchip Technology Inc.

24.5.3 Fuse Index Register

Name:FUSE_IR

Address:0xFFFFDC08

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	WSEL			
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RS	WS

WS: Write Status

0: Write is pending or no write has been requested since general reset.

1: Write of fuses is done.

RS: Read Status

0: Read is pending or no read has been requested since general reset.

1: Read of fuses is done.

WSEL: Word Selection

0-15: Selects the word to write.

24.5.4 Fuse Data Register

Name:FUSE_DR

Address:0xFFFFDC0C

Access: Read/Write

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

DATA: Data to Program

Data to program. Only bits of with a value of “1” will be programmed.

© 2017 Microchip Technology Inc.

24.5.5 Fuse Status Register

Name:FUSE_SRx [x=0..9]

Address:0xFFFFDC10

Access: Read-only

31	30	29	28	27	26	25	24
FUSE							
23	22	21	20	19	18	17	16
FUSE							
15	14	13	12	11	10	9	8
FUSE							
7	6	5	4	3	2	1	0
FUSE							

FUSE: Fuse Status

Indicates the status of corresponding fuses:

0: Unprogrammed.

1: Programmed.

25. Bus Matrix (MATRIX)

25.1 Description

The Bus Matrix implements a multi-layer AHB, based on the AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system, thus increasing the overall bandwidth. The Bus Matrix interconnects up to 16 AHB masters to up to 16 AHB slaves. The normal latency to connect a master to a slave is one cycle except for the default master of the accessed slave which is connected directly (zero cycle latency).

The Bus Matrix user interface is compliant with Arm Advanced Peripheral Bus and provides a Chip Configuration User Interface with Registers that allow the Bus Matrix to support application specific features.

25.2 Embedded Characteristics

- 6-layer Matrix, handling requests from 6 masters
- Programmable Arbitration strategy
  - Fixed-priority Arbitration
  - Round-Robin Arbitration, either with no default master, last accessed default master or fixed default master
- Burst Management
  - Breaking with Slot Cycle Limit Support
  - Undefined Burst Length Support
- One Address Decoder provided per Master
  - Three different slaves may be assigned to each decoded memory area: one for internal ROM boot, one after remap
- Boot Mode Select
  - Non-volatile Boot Memory can be internal ROM or external memory on EBI_NCS0
- Remap Command
  - Allows Remapping of an Internal SRAM in Place of the Boot Non-Volatile Memory (ROM or External Flash)
  - Allows Handling of Dynamic Exception Vectors

25.3 Matrix Masters

The Bus Matrix of the SAM9N12/SAM9CN11/SAM9CN12 product manages six masters, which means that each master can perform an access concurrently with others, to an available slave.

Each master has its own decoder, which is defined specifically for each master. In order to simplify the addressing, all the masters have the same decodings.

Table 25-1: List of Bus Matrix Masters

Master 0	Arm926 Instruction
Master 1	Arm926 Data
Master 2&3	DMA Controller
Master 4	USB Host DMA
Master 5	LCD DMA

25.4 Matrix Slaves

The Bus Matrix of the SAM9N12/SAM9CN11/SAM9CN12 product manages five slaves. Each slave has its own arbiter, allowing a different arbitration per slave.

© 2017 Microchip Technology Inc.

Table 25-2: List of Bus Matrix Slaves

Slave 0	Internal SRAM
Slave 1	Internal ROM
	USB Host User Interface
Slave 2	External Bus Interface
Slave 3	Peripheral Bridge 0
Slave 4	Peripheral Bridge 1

25.5 Master to Slave Access

All the Masters can normally access all the Slaves. However, some paths do not make sense, for example allowing access from the USB Device High speed DMA to the Internal Peripherals. Thus, these paths are forbidden or simply not wired, and shown as "–" in the following table.

Table 25-3: SAM9N12/SAM9CN11/SAM9CN12 Master to Slave Access

Masters		0	1	2&3	4	5
Slaves		Arm926 Instruction	Arm926 Data	DMA	USB Host DMA	LCD DMA
0	Internal SRAM	X	X	X	X	X
1	Internal ROM USB Host User Interface	X	X	X	–	–
2	External Bus Interface	X	X	X	X	X
3	Peripheral Bridge 0	X	X	X	–	–
4	Peripheral Bridge 1	X	X	X	–	–

25.6 Memory Mapping

The Bus Matrix provides one decoder for every AHB master interface. The decoder offers each AHB master several memory mappings. In fact, depending on the product, each memory area may be assigned to several slaves. Booting at the same address while using different AHB slaves (i.e., external RAM, internal ROM or internal Flash, etc.) becomes possible.

The Bus Matrix user interface provides Master Remap Control Register (MATRIX_MRCR) that performs remap action for every master independently.

25.7 Special Bus Granting Mechanism

The Bus Matrix provides some speculative bus granting techniques in order to anticipate access requests from some masters. This mechanism reduces latency at first access of a burst or single transfer as long as the slave is free from any other master access, but does not provide any benefit as soon as the slave is continuously accessed by more than one master, since arbitration is pipelined and then has no negative effect on the slave bandwidth or access latency.

This bus granting mechanism sets a different default master for every slave.

At the end of the current access, if no other request is pending, the slave remains connected to its associated default master. A slave can be associated with three kinds of default masters: no default master, last access master and fixed default master.

To change from one kind of default master to another, the Bus Matrix user interface provides the Slave Configuration Registers, one for each slave, that set a default master for each slave. The Slave Configuration Register contains two fields: DEFMSTR_TYPE and FIXED_DEFMSTR. The 2-bit DEFMSTR_TYPE field selects the default master type (no default, last access master, fixed default master), whereas the 4-bit FIXED_DEFMSTR field selects a fixed default master provided that DEFMSTR_TYPE is set to fixed default master. Refer to Section 25.10.2 "Bus Matrix Slave Configuration Registers".

25.7.1 No Default Master

After the end of the current access, if no other request is pending, the slave is disconnected from all masters. No Default Master suits low-power mode.

This configuration incurs one latency clock cycle for the first access of a burst after bus Idle. Arbitration without default master may be used for masters that perform significant bursts or several transfers with no Idle in between, or if the slave bus bandwidth is widely used by one or more masters.

This configuration provides no benefit on access latency or bandwidth when reaching maximum slave bus throughput whatever is the number of requesting masters.

25.7.2 Last Access Master

After the end of the current access, if no other request is pending, the slave remains connected to the last master that performed an access request.

This allows the Bus Matrix to remove the one latency cycle for the last master that accessed the slave. Other non privileged masters still get one latency clock cycle if they want to access the same slave. This technique is useful for masters that mainly perform single accesses or short bursts with some Idle cycles in between.

This configuration provides no benefit on access latency or bandwidth when reaching maximum slave bus throughput whatever is the number of requesting masters.

25.7.3 Fixed Default Master

After the end of the current access, if no other request is pending, the slave connects to its fixed default master. Unlike last access master, the fixed master does not change unless the user modifies it by a software action (field FIXED_DEFMSTR of the related MATRIX_SCFG).

This allows the Bus Matrix arbiters to remove the one latency clock cycle for the fixed default master of the slave. Every request attempted by this fixed default master will not cause any arbitration latency whereas other non privileged masters will still get one latency cycle. This technique is useful for a master that mainly perform single accesses or short bursts with some Idle cycles in between.

This configuration provides no benefit on access latency or bandwidth when reaching maximum slave bus throughput whatever is the number of requesting masters.

25.8 Arbitration

The Bus Matrix provides an arbitration mechanism that reduces latency when conflict cases occur, i.e. when two or more masters try to access the same slave at the same time. One arbiter per AHB slave is provided, thus arbitrating each slave differently.

The Bus Matrix provides the user with the possibility of choosing between 2 arbitration types or mixing them for each slave:

1. Round-Robin Arbitration (default)
2. Fixed Priority Arbitration

The resulting algorithm may be complemented by selecting a default master configuration for each slave.

When a re-arbitration must be done, specific conditions apply. See Section 25.8.1 “Arbitration Scheduling”.

25.8.1 Arbitration Scheduling

Each arbiter has the ability to arbitrate between two or more different master requests. In order to avoid burst breaking and also to provide the maximum throughput for slave interfaces, arbitration may only take place during the following cycles:

1. Idle Cycles: When a slave is not connected to any master or is connected to a master which is not currently accessing it.
2. Single Cycles: When a slave is currently doing a single access.
3. End of Burst Cycles: When the current cycle is the last cycle of a burst transfer. For defined length burst, predicted end of burst matches the size of the transfer but is managed differently for undefined length burst. See Section 25.8.1.1 “Undefined Length Burst Arbitration”.
4. Slot Cycle Limit: When the slot cycle counter has reached the limit value indicating that the current master access is too long and must be broken. See Section 25.8.1.2 “Slot Cycle Limit Arbitration”.

© 2017 Microchip Technology Inc.

25.8.1.1 Undefined Length Burst Arbitration

In order to optimize AHB burst lengths and arbitration, it may be interesting to set a maximum for undefined length bursts (INCR). The Bus Matrix provides specific logic in order to re-arbitrate before the end of the INCR transfer. A predicted end of burst is used as a defined length burst transfer and can be selected from among the following Undefined Length Burst Type (ULBT) possibilities:

1. Unlimited: No predicted end of burst is generated and therefore INCR burst transfer will not be broken by this way, but will be able to complete unless broken at the Slot Cycle Limit. This is normally the default and should be let as is in order to be able to allow full 1 Kilobyte AHB intra-boundary 256-beat word bursts performed by some Microchip AHB masters.
2. 1-beat bursts: Predicted end of burst is generated at each single transfer inside the INCR transfer.
3. 4-beat bursts: Predicted end of burst is generated at the end of each 4-beat boundary inside INCR transfer.
4. 8-beat bursts: Predicted end of burst is generated at the end of each 8-beat boundary inside INCR transfer.
5. 16-beat bursts: Predicted end of burst is generated at the end of each 16-beat boundary inside INCR transfer.
6. 32-beat bursts: Predicted end of burst is generated at the end of each 32-beat boundary inside INCR transfer.
7. 64-beat bursts: Predicted end of burst is generated at the end of each 64-beat boundary inside INCR transfer.
8. 128-beat bursts: Predicted end of burst is generated at the end of each 128-beat boundary inside INCR transfer.

Use of undefined length 16-beat bursts or less is discouraged since this generally decreases significantly overall bus bandwidth due to arbitration and slave latencies at each first access of a burst.

If the master does not permanently and continuously request the same slave or has an intrinsically limited average throughput, the ULBT should be let at its default unlimited value, knowing that the AHB specification natively limits all word bursts to 256 beats and double-word bursts to 128 beats because of its 1 Kbyte address boundaries.

Unless duly needed the ULBT should be let to its default 0 value for power saving.

This selection can be done through the field ULBT of the Master Configuration Registers (MATRIX_MCFG).

25.8.1.2 Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break long accesses, such as back to back undefined length bursts or very long bursts on a very slow slave (e.g., an external low speed memory). At each arbitration time a counter is loaded with the value previously written in the SLOT_CYCLE field of the related Slave Configuration Register (MATRIX_SCFG) and decreased at each clock cycle. When the counter elapses, the arbiter has the ability to re-arbitrate at the end of the current AHB bus access cycle.

Unless some master has a very tight access latency constraint which could lead to data overflow or underflow due to a badly undersized internal fifo with respect to its throughput, the Slot Cycle Limit should be disabled (SLOT_CYCLE = 0) or let to its default maximum value in order not to inefficiently break long bursts performed by some Microchip masters.

However, the Slot Cycle Limit should not be disabled in the very particular case of a master capable of accessing the slave by performing back to back undefined length bursts shorter than the number of ULBT beats with no Idle cycle in between, since in this case the arbitration could be frozen all along the bursts sequence.

In most cases this feature is not needed and should be disabled for power saving.

Warning: This feature cannot prevent any slave from locking its access indefinitely.

25.8.2 Arbitration Priority Scheme

The bus Matrix arbitration scheme is organized in priority pools.

Round-Robin priority is used inside the highest and lowest priority pools, whereas fix level priority is used between priority pools and inside the intermediate priority pools.

For each slave, each master x is assigned to one of the slave priority pools through the Priority Registers for Slaves (MxPR fields of MATRIX_PRAS and MATRIX_PRBS). When evaluating masters requests, this programmed priority level always takes precedence.

After reset, all the masters are belonging to the lowest priority pool (MxPR = 0) and so are granted bus access in a true Round-Robin fashion.

The highest priority pool must be specifically reserved for masters requiring very low access latency. If more than one master belong to this pool, these will be granted bus access in a biased Round-Robin fashion which allow tight and deterministic maximum access latency from AHB bus request. In fact, at worst, any currently high priority master request will be granted after the current bus master access is ended and the other high priority pool masters, if any, have been granted once each.

The lowest priority pool shares the remaining bus bandwidth between AHB Masters.

Intermediate priority pools allow fine priority tuning. Typically, a moderately latency critical master or a bandwidth only critical master will use such a priority level. The higher the priority level (MxPR value), the higher the master priority.

© 2017 Microchip Technology Inc.

All combination of MxPR values are allowed for all masters and slaves. For example some masters might be assigned to the highest priority pool (round-robin) and the remaining masters to the lowest priority pool (round-robin), with no master for intermediate fix priority levels.

If more than one master is requesting the slave bus, whatever are the respective masters priorities, no master will be granted the slave bus for two consecutive runs. A master can only get back to back grants as long as it is the only requesting master.

25.8.2.1 Fixed Priority Arbitration

This arbitration algorithm is the first and only applied between masters from distinct priority pools. It is also used inside priority pools other than the highest and lowest ones (intermediate priority pools).

It allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave by using the fixed priority defined by the user in the MxPR field for each master inside the MATRIX_PRAS and MATRIX_PRBS Priority Registers. If two or more master requests are active at the same time, the master with the highest priority number MxPR is serviced first.

Inside intermediate priority pools, if two or more master requests with the same priority are active at the same time, the master with the highest number is serviced first.

25.8.2.2 Round-Robin Arbitration

This algorithm is only used inside the highest and lowest priority pools. It allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave in a fair way. If two or more master requests are active at the same time inside the priority pool, they are serviced in a round-robin increasing master number order.

25.9 Write Protect Registers

To prevent any single software error that may corrupt MATRIX behavior, the entire MATRIX address space from address offset 0x000 to 0x1FC can be write protected by setting the WPEN bit in the MATRIX Write Protect Mode Register (MATRIX_WPMR).

If a write access to anywhere in the MATRIX address space from address offset 0x000 to 0x1FC is detected, then the WPVS flag in the MATRIX Write Protect Status Register (MATRIX_WPSR) is set and the field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is reset by writing the MATRIX Write Protect Mode Register (MATRIX_WPMR) with the appropriate access key WPKEY.

© 2017 Microchip Technology Inc.

25.10 Bus Matrix (MATRIX) User Interface

Table 25-4: Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Master Configuration Register 0	MATRIX_MCFG0	Read/Write	0x00000001
0x0004	Master Configuration Register 1	MATRIX_MCFG1	Read/Write	0x00000000
0x0008	Master Configuration Register 2	MATRIX_MCFG2	Read/Write	0x00000000
0x000C	Master Configuration Register 3	MATRIX_MCFG3	Read/Write	0x00000000
0x0010	Master Configuration Register 4	MATRIX_MCFG4	Read/Write	0x00000000
0x0014	Master Configuration Register 5	MATRIX_MCFG5	Read/Write	0x00000000
0x0018–0x003C	Reserved	–	–	–
0x0040	Slave Configuration Register 0	MATRIX_SCFG0	Read/Write	0x000001FF
0x0044	Slave Configuration Register 1	MATRIX_SCFG1	Read/Write	0x000001FF
0x0048	Slave Configuration Register 2	MATRIX_SCFG2	Read/Write	0x000001FF
0x004C	Slave Configuration Register 3	MATRIX_SCFG3	Read/Write	0x000001FF
0x0050	Slave Configuration Register 4	MATRIX_SCFG4	Read/Write	0x000001FF
0x0054–0x007C	Reserved	–	–	–
0x0080	Priority Register A for Slave 0	MATRIX_PRAS0	Read/Write	0x00000000
0x0084	Reserved	–	–	–
0x0088	Priority Register A for Slave 1	MATRIX_PRAS1	Read/Write	0x00000000
0x008C	Reserved	–	–	–
0x0090	Priority Register A for Slave 2	MATRIX_PRAS2	Read/Write	0x00000000
0x0094	Reserved	–	–	–
0x0098	Priority Register A for Slave 3	MATRIX_PRAS3	Read/Write	0x00000000
0x009C	Reserved	–	–	–
0x00A0	Priority Register A for Slave 4	MATRIX_PRAS4	Read/Write	0x00000000
0x00A4–0x00FC	Reserved	–	–	–
0x0100	Master Remap Control Register	MATRIX_MRCR	Read/Write	0x00000000
0x0104–0x010C	Reserved	–	–	–
0x0110–0x01E0	Chip Configuration Registers	–	–	–
0x01E4	Write Protect Mode Register	MATRIX_WPMR	Read/Write	0x00000000
0x01E8	Write Protect Status Register	MATRIX_WPSR	Read-only	0x00000000

© 2017 Microchip Technology Inc.

25.10.1 Bus Matrix Master Configuration Registers

Name:MATRIX_MCFG0...MATRIX_MCFG5

Addresses:0xFFFFDE00 [0], 0xFFFFDE04 [1], 0xFFFFDE08 [2], 0xFFFFDE0C [3], 0xFFFFDE10 [4], 0xFFFFDE14 [5]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	ULBT		

ULBT: Undefined Length Burst Type

0: Unlimited Length Burst

No predicted end of burst is generated and therefore INCR bursts coming from this master can only be broken if the Slave Slot Cycle Limit is reached. If the Slot Cycle Limit is not reached, the burst is normally completed by the master, at the latest, on the next AHB 1 Kbyte address boundary, allowing up to 256-beat word bursts or 128-beat double-word bursts.

1: Single Access

The undefined length burst is treated as a succession of single accesses, allowing re-arbitration at each beat of the INCR burst.

2: 4-beat Burst

The undefined length burst is split into 4-beat bursts, allowing re-arbitration at each 4-beat burst end.

3: 8-beat Burst

The undefined length burst is split into 8-beat bursts, allowing re-arbitration at each 8-beat burst end.

4: 16-beat Burst

The undefined length burst is split into 16-beat bursts, allowing re-arbitration at each 16-beat burst end.

5: 32-beat Burst

The undefined length burst is split into 32-beat bursts, allowing re-arbitration at each 32-beat burst end.

6: 64-beat Burst

The undefined length burst is split into 64-beat bursts, allowing re-arbitration at each 64-beat burst end.

7: 128-beat Burst

The undefined length burst is split into 128-beat bursts, allowing re-arbitration at each 128-beat burst end.

Unless duly needed the ULBT should be let to its default 0 value for power saving.

© 2017 Microchip Technology Inc.

25.10.2 Bus Matrix Slave Configuration Registers

Name:MATRIX_SCFG0...MATRIX_SCFG4

Addresses:0xFFFFDE40 [0], 0xFFFFDE44 [1], 0xFFFFDE48 [2], 0xFFFFDE4C [3], 0xFFFFDE50 [4]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	FIXED_DEFMSTR				DEFMSTR_TYPE	

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	SLOT_CYCLE

7	6	5	4	3	2	1	0
SLOT_CYCLE							

SLOT_CYCLE: Maximum Bus Grant Duration for Masters

When SLOT_CYCLE AHB clock cycles have elapsed since the last arbitration, a new arbitration takes place so as to let an other master access this slave. If an other master is requesting the slave bus, then the current master burst is broken.

If SLOT_CYCLE = 0, the Slot Cycle Limit feature is disabled and bursts always complete unless broken according to the ULBT.

This limit has been placed in order to enforce arbitration so as to meet potential latency constraints of masters waiting for slave access or in the particular case of a master performing back to back undefined length bursts indefinitely freezing the arbitration.

This limit must not be small. Unreasonably small values break every burst and the Bus Matrix arbitrates without performing any data transfer. The default maximum value is usually an optimal conservative choice.

In most cases this feature is not needed and should be disabled for power saving. See Section 25.8.1.2 “Slot Cycle Limit Arbitration”.

DEFMSTR_TYPE: Default Master Type

0: No Default Master

At the end of the current slave access, if no other master request is pending, the slave is disconnected from all masters.

This results in a one clock cycle latency for the first access of a burst transfer or for a single access.

1: Last Default Master

At the end of the current slave access, if no other master request is pending, the slave stays connected to the last master having accessed it.

This results in not having one clock cycle latency when the last master tries to access the slave again.

2: Fixed Default Master

At the end of the current slave access, if no other master request is pending, the slave connects to the fixed master the number that has been written in the FIXED_DEFMSTR field.

This results in not having one clock cycle latency when the fixed master tries to access the slave again.

FIXED_DEFMSTR: Fixed Default Master

This is the number of the Default Master for this slave. Only used if DEFMSTR_TYPE is 2. Specifying the number of a master which is not connected to the selected slave is equivalent to setting DEFMSTR_TYPE to 0.

© 2017 Microchip Technology Inc.

25.10.3 Bus Matrix Priority Registers A For Slaves

Name:MATRIX_PRAS0...MATRIX_PRAS4

Addresses:0xFFFFDE80 [0], 0xFFFFDE88 [1], 0xFFFFDE90 [2], 0xFFFFDE98 [3], 0xFFFFDEA0 [4]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	M5PR		–	–	M4PR	

15	14	13	12	11	10	9	8
–	–	M3PR		–	–	M2PR	

7	6	5	4	3	2	1	0
–	–	M1PR		–	–	M0PR	

MxPR: Master x Priority

Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

All the masters programmed with the same MxPR value for the slave make up a priority pool.

Round-Robin arbitration is used inside the lowest (MxPR = 0) and highest (MxPR = 3) priority pools.

Fixed priority is used inside intermediate priority pools (MxPR = 1) and (MxPR = 2).

See Section 25.8.2 “Arbitration Priority Scheme” for details.

© 2017 Microchip Technology Inc.

25.10.4 Bus Matrix Master Remap Control Register

Name:MATRIX_MRCR

Address:0xFFFFDF00

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	RCB5	RCB4	RCB3	RCB2	RCB1	RCB0

RCBx: Remap Command Bit for Master x

0: Disables remapped address decoding for the selected Master.

1: Enables remapped address decoding for the selected Master.

25.10.5 Chip Configuration User Interface

Table 25-5: Chip Configuration User Interface

Offset	Register	Name	Access	Reset Value
0x0110–0x0114	Reserved	–	–	–
0x0118	EBI Chip Select Assignment Register	CCFG_EBICSA	Read/Write	0x00000000
0x011C–0x01FC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

25.10.5.1 EBI Chip Select Assignment Register

Name:CCFG_EBICSA

Access: Read/Write

Reset:0x0000_0000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	NFD0_ON_D16
23	22	21	20	19	18	17	16
–	–	–	–	–	–	EBI_DRIVE	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	EBI_DBPDC	EBI_DBPUC
7	6	5	4	3	2	1	0
–	–	–	–	EBI_CS3A	–	EBI_CS1A	–

EBI_CS1A: EBI Chip Select 1 Assignment

0: EBI Chip Select 1 is assigned to the Static Memory Controller.

1: EBI Chip Select 1 is assigned to the DDR2SDR Controller.

EBI_CS3A: EBI Chip Select 3 Assignment

0: EBI Chip Select 3 is only assigned to the Static Memory Controller and EBI_NCS3 behaves as defined by the SMC.

1: EBI Chip Select 3 is assigned to the Static Memory Controller and the NAND Flash Logic is activated.

EBI_DBPUC: EBI Data Bus Pull-Up Configuration

0: EBI D0–D15 Data Bus bits are internally pulled-up to the VDDIOM power supply.

1: EBI D0–D15 Data Bus bits are not internally pulled-up.

EBI_DBPDC: EBI Data Bus Pull-Down Configuration

0: EBI D0–D15 Data Bus bits are internally pulled-down to the GND.

1: EBI D0–D15 Data Bus bits are not internally pulled-down.

EBI_DRIVE: EBI I/O Drive Configuration

0: LOW drive(1)

1: HIGH drive (default)(1)

Note 1: Load capacitance defined in Table 47-18 ”I/O Characteristics”

NFD0_ON_D16: NAND Flash databus selection

0: NAND Flash I/Os are connected to D0–D15. VDDNF must be equal to VDDIOM (default).

1: NAND Flash I/Os are connected to D16–D31. VDDNF can be different from or equal to VDDIOM.

This can be used if the SMC connects to the NAND Flash only. Using this function with another device on the SMC will lead to an unpredictable behavior of that device. In that case, the default value must be selected.

Table 25-6: Connection Examples with Various VDDNF and VDDIOM

NFD0_ON_D16	Signals	VDDIOM	VDDNF	External Memory
0	NFD0 = D0, ..., NFD15 = D15	1.8V	1.8V	DDR2 or LPDDR or LPSDR + NAND Flash 1.8V
0	NFD0 = D0, ..., NFD15 = D15	3.3V	3.3V	32-bit SDR + NAND Flash 3.3V

© 2017 Microchip Technology Inc.

Table 25-6: Connection Examples with Various VDDNF and VDDIOM (Continued)

1	NFD0 = D16, ..., NFD15 = D31	1.8V	1.8V	DDR2 or LPDDR or LPSDR + NAND Flash 1.8V
1	NFD0 = D16, ..., NFD15 = D31	1.8V	3.3V	DDR2 or LPDDR or LPSDR + NAND Flash 3.3V
1	NFD0 = D16, ..., NFD15 = D31	3.3V	1.8V	16-bit SDR + NAND Flash 1.8V

© 2017 Microchip Technology Inc.

25.10.6 Write Protect Mode Register

Name:MATRIX_WPMR

Address:0xFFFFDFE4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

For more details on MATRIX_WPMR, refer to Section 25.9 “Write Protect Registers”.

WPEN: Write Protect Enable

0: Disables the Write Protect if WPKEY corresponds to 0x4D4154 (“MAT” in ASCII).

1: Enables the Write Protect if WPKEY corresponds to 0x4D4154 (“MAT” in ASCII).

Protects the entire MATRIX address space from address offset 0x000 to 0x1FC.

WPKEY: Write Protect Key (Write-only)

Value	Name	Description
0x4D4154	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

25.10.7 Write Protect Status Register

Name:MATRIX_WPSR

Address:0xFFFFDFE8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

For more details on MATRIX_WPSR, refer to Section 25.9 “Write Protect Registers”.

WPVS: Write Protect Violation Status

0: No Write Protect Violation has occurred since the last write of the MATRIX_WPMR.

1: At least one Write Protect Violation has occurred since the last write of the MATRIX_WPMR.

WPVSRC: Write Protect Violation Source

When WPVS is active, this field indicates the register address offset in which a write access has been attempted.

Otherwise it reads as 0.

© 2017 Microchip Technology Inc.

26. External Bus Interface (EBI)

26.1 Description

The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the embedded Memory Controller of an Arm-based device.

The Static Memory, DDR, SDRAM and ECC Controllers are all featured external Memory Controllers on the EBI. These external Memory Controllers are capable of handling several types of external memory and peripheral devices, such as SRAM, PROM, EPROM, EEPROM, Flash, DDR2 and SDRAM. The EBI operates with 1.8V or 3.3V Power Supply (VDDIOM).

The EBI also supports the NAND Flash protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI handles data transfers with up to six external devices, each assigned to six address spaces defined by the embedded Memory Controller. Data transfers are performed through a 16-bit or 32-bit data bus, an address bus of up to 26 bits, up to six chip select lines (NCS[5:0]) and several control pins that are generally multiplexed between the different external Memory Controllers.

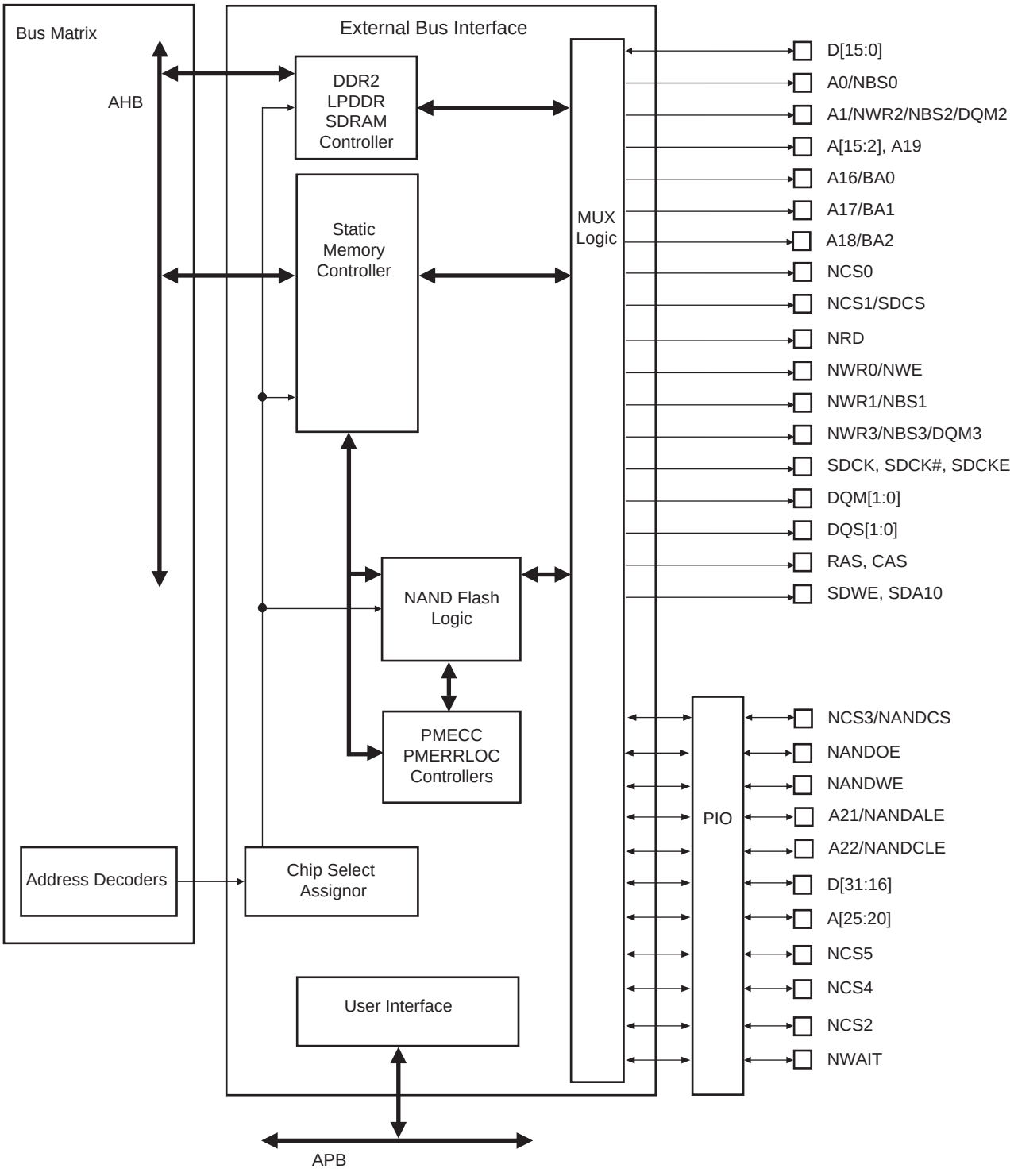
26.2 Embedded Characteristics

32-bit Wide Interface, Supporting:

- 16-bit DDR2/LPDDR, 32-bit SDRAM/LPSDR
- Static Memories
- NAND Flash with Multi-bit ECC

26.3 EBI Block Diagram

Figure 26-1: Organization of the External Bus Interface



© 2017 Microchip Technology Inc.

26.4 I/O Lines Description

Table 26-1: EBI I/O Lines Description

Name	Function	Type	Active Level
EBI			
EBI_D0–EBI_D31	Data Bus	I/O	–
EBI_A0–EBI_A25	Address Bus	Output	–
EBI_NWAIT	External Wait Signal	Input	Low
SMC			
EBI_NCS0–EBI_NCS5	Chip Select Lines	Output	Low
EBI_NWR0–EBI_NWR3	Write Signals	Output	Low
EBI_NRD	Read Signal	Output	Low
EBI_NWE	Write Enable	Output	Low
EBI_NBS0–EBI_NBS3	Byte Mask Signals	Output	Low
EBI for NAND Flash Support			
EBI_NANDCS	NAND Flash Chip Select Line	Output	Low
EBI_NANDOE	NAND Flash Output Enable	Output	Low
EBI_NANDWE	NAND Flash Write Enable	Output	Low
DDR2/SDRAM Controller			
EBI_SDCK, EBI_SDCK#	DDR2/SDRAM Differential Clock	Output	–
EBI_SDCKE	DDR2/SDRAM Clock Enable	Output	High
EBI_SDCS	DDR2/SDRAM Controller Chip Select Line	Output	Low
EBI_BA0–2	Bank Select	Output	–
EBI_SDWE	DDR2/SDRAM Write Enable	Output	Low
EBI_RAS - EBI_CAS	Row and Column Signal	Output	Low
EBI_SDA10	SDRAM Address 10 Line	Output	–

The connection of some signals through the MUX logic is not direct and depends on the memory controller currently in use. Table 26-2 details the connections between the two Memory Controllers and the EBI pins.

Table 26-2: EBI Pins and Memory Controllers I/O Lines Connections

EBIx Pins	SDRAM I/O Lines	SMC I/O Lines
EBI_NWR1/NBS1/CFIOR	NBS1	NWR1
EBI_A0/NBS0	Not Supported	SMC_A0
EBI_A1/NBS2/NWR2	Not Supported	SMC_A1
EBI_A[11:2]	SDRAMC_A[9:0]	SMC_A[11:2]
EBI_SDA10	SDRAMC_A10	Not Supported
EBI_A12	Not Supported	SMC_A12
EBI_A[14:13]	SDRAMC_A[12:11]	SMC_A[14:13]
EBI_A[25:15]	Not Supported	SMC_A[25:15]
EBI_D[31:0]	D[31:0]	D[31:0]

© 2017 Microchip Technology Inc.

26.5 Application Example

26.5.1 Hardware Interface

Table 26-3 details the connections to be applied between the EBI pins and the external devices for each memory controller.

Table 26-3: EBI Pins and External Static Device Connections

	Pins of the SMC Interfaced Device					
Signals: EBI_	**8-bit Static Device**	**2 x 8-bit Static Devices**	**16-bit Static Device**	**4 x 8-bit Static Devices**	**2 x 16-bit Static Devices**	**32-bit Static Device**
D0–D7	D0–D7	D0–D7	D0–D7	D0–D7	D0–D7	D0–D7
D8–D15	–	D8–D15	D8–D15	D8–D15	D8–15	D8–15
D16–D24	–	–	–	D16–D23	D16–D23	D16–D23
D25–D31(5))	–	–	–	D24–D31	D24–D31	D24–D31
A0/NBS0	A0	–	NLB	–	NLB(3)	BE0
A1/NWR2/NBS2/DQM2	A1	A0	A0	WE(2)	NLB(4)	BE2
A2–A22(5)	A[2:22]	A[1:21]	A[1:21]	A[0:20]	A[0:20]	A[0:20]
A23–A25(5)	A[23:25]	A[22:24]	A[22:24]	A[21:23]	A[21:23]	A[21:23]
NCS0	CS	CS	CS	CS	CS	CS
NCS1/DDRSDCS	CS	CS	CS	CS	CS	CS
NCS2(5)	CS	CS	CS	CS	CS	CS
NCS3/NANDCS	CS	CS	CS	CS	CS	CS
NCS4(5)	CS	CS	CS	CS	CS	CS
NCS5(5)	CS	CS	CS	CS	CS	CS
NRD	OE	OE	OE	OE	OE	OE
NWR0/NWE	WE	WE(1)	WE	WE(2)	WE	WE
NWR1/NBS1	–	WE(1)	NUB	WE(2)	NUB(3)	BE1
NWR3/NBS3/DQM3	–	–	–	WE(2)	NUB(4)	BE3

Note 1: NWR1 enables upper byte writes. NWR0 enables lower byte writes.

2: NWRx enables corresponding byte x writes. (x = 0, 1, 2 or 3).

3: NBS0 and NBS1 enable respectively lower and upper bytes of the lower 16-bit word.

4: NBS2 and NBS3 enable respectively lower and upper bytes of the upper 16-bit word.

5: D25–D31 and A20, A23-A25, NCS2, NCS4, NCS5 are multiplexed on PD15–PD31.

© 2017 Microchip Technology Inc.

Table 26-4: EBI Pins and External Device Connections

Signals: EBI_	Power supply	Pins of the Interfaced Device		
		DDRSDRC		SMC
		DDR2/LPDDR	SDR/LPSDR	NAND Flash
D0–D15	VDDIOM	D0–D15	D0–D15	NFD0–NFD15(1)
D16–D31	VDDNF	–	D16–D31	NFD0–NFD15(1)
A0/NBS0	VDDIOM	–	–	–
A1/NWR2/NBS2/DQM2	VDDIOM	–	DQM2	–
DQM0–DQM1	VDDIOM	DQM0–DQM1	DQM0–DQM1	–
DQS0–DQS1	VDDIOM	DQS0–DQS1	–	–
A2–A10	VDDIOM	A[0:8]	A[0:8]	–
A11	VDDIOM	A9	A9	–
SDA10	VDDIOM	A10	A10	–
A12	VDDIOM	–	–	–
A13–A14	VDDIOM	A[11:12]	A[11:12]	–
A15	VDDIOM	A13	–	–
A16/BA0	VDDIOM	BA0	BA0	–
A17/BA1	VDDIOM	BA1	BA1	–
A18/BA2	VDDIOM	BA2	BA2	–
A19	VDDIOM	–	–	–
A20	VDDIOM	–	–	–
A21/NANDALE	VDDNF	–	–	ALE
A22/NANDCLE	VDDNF	–	–	CLE
A23–A24	VDDIOM	–	–	–
A25	VDDIOM	–	–	–
NCS0	VDDIOM	–	–	–
NCS1/DDRSDCS	VDDIOM	DDRCS	SDCS	–
NCS2	VDDIOM	–	–	–
NCS3/NANDCS	VDDNF	–	–	CE
NCS4	VDDIOM	–	–	–
NCS5	VDDIOM	–	–	–
NANDOE	VDDNF	–	–	OE
NANDWE	VDDNF	–	–	WE
NRD	VDDIOM	–	–	–
NWR0/NWE	VDDIOM	–	–	–
NWR1/NBS1	VDDIOM	–	–	–
NWR3/NBS3/DQM3	VDDIOM	–	DQM3	–

Table 26-4: EBI Pins and External Device Connections (Continued)

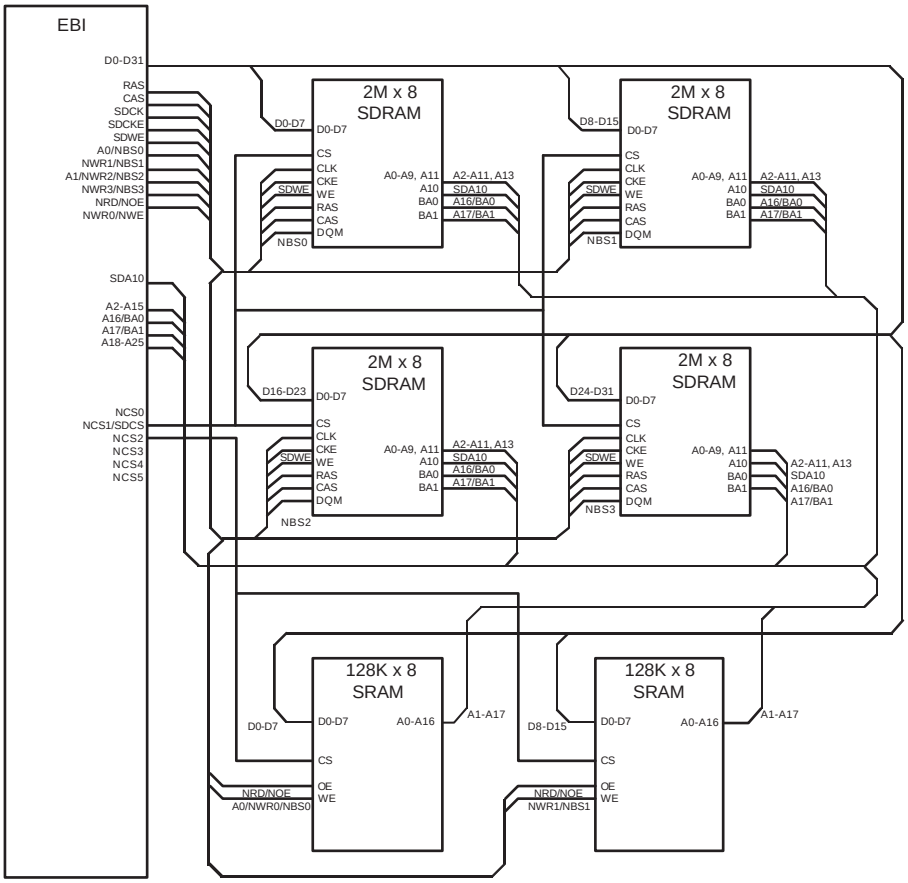
Signals: EBI_	Power supply	Pins of the Interfaced Device		
		DDRSDRC		SMC
		DDR2/LPDDR	SDR/LPSDR	NAND Flash
SDCK	VDDIOM	CK	CK	–
SDCK#	VDDIOM	CK#	–	–
SDCKE	VDDIOM	CKE	CKE	–
RAS	VDDIOM	RAS	RAS	–
CAS	VDDIOM	CAS	CAS	–
SDWE	VDDIOM	WE	WE	–
Pxx	VDDNF	–	–	CE
Pxx	VDDNF	–	–	RDY

Note 1: A switch, NFD0_ON_D16, enables the user to select NAND Flash path on D0–D7 or D16–D24 depending on memory power supplies. This switch is located in the EBI Chip Select Assignment Register (CCFG_EBICSA) in the Bus Matrix user interface.

26.5.2 Connection Examples

Figure 26-2 shows an example of connections between the EBI and external devices.

Figure 26-2: EBI Connections to Memory Devices



© 2017 Microchip Technology Inc.

26.6 Product Dependencies

26.6.1 I/O Lines

The pins used for interfacing the External Bus Interface may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the External Bus Interface pins to their peripheral function. If I/O lines of the External Bus Interface are not used by the application, they can be used for other purposes by the PIO Controller.

26.7 Functional Description

The EBI transfers data between the internal AHB Bus (handled by the Bus Matrix) and the external memories or peripheral devices. It controls the waveforms and the parameters of the external address, data and control buses and is composed of the following elements:

- the Static Memory Controller (SMC)
- the DDR2/SDRAM Controller (DDR2SDRC)
- the Programmable Multibit Error Correction Code Controller (PMECC)
- a chip select assignment feature that assigns an AHB address space to the external devices
- a multiplex controller circuit that shares the pins between the different Memory Controllers
- programmable NAND Flash support logic

26.7.1 Bus Multiplexing

The EBI offers a complete set of control signals that share the 32-bit data lines, the address lines of up to 26 bits and the control signals through a multiplex logic operating in function of the memory area requests.

Multiplexing is specifically organized in order to guarantee the maintenance of the address and output control lines at a stable state while no external access is being performed. Multiplexing is also designed to respect the data float times defined in the memory controllers. Furthermore, refresh cycles of the DDR2 and SDRAM are executed independently by the DDR2SDR Controller without delaying the other external memory controller accesses.

26.7.2 Pull-up Control

The EBI Chip Select Assignment Register (CCFG_EBICSA) in the Chip Configuration User Interface permits enabling of on-chip pull-up resistors on the data bus lines not multiplexed with the PIO Controller lines. The pull-up resistors are enabled after reset. Setting the EBIx_DBPUC bit disables the pull-up resistors on the lines D0–D15. Enabling the pull-up resistor on the lines D16–D31 can be performed by programming the appropriate PIO controller.

26.7.3 Drive level

The EBI I/Os accept two drive level, HIGH and LOW. This allows to avoid overshoots and give the best performances according to the bus load and external memories.

The voltage ranges and the slew rates are determined by programming the EBI_DRIVE field in the CCFG_EBICSA register.

At reset the selected current drive is HIGH.

26.7.4 Power Supplies

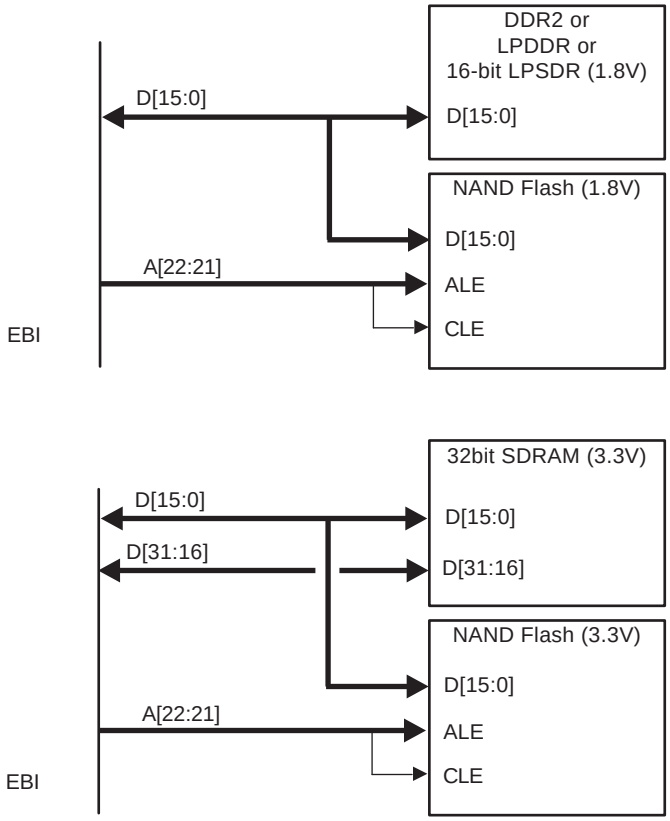
The product embeds a dual power supply for EBI, VDDNF for NAND Flash signals, and VDDIOM for others. This allows to use an 1.8V or 3.3V NAND Flash independently of SDRAM power supply.

A switch, NFD0_ON_D16, enables the user to select NAND Flash path on D0-D15 or D16-D32 depending on memory power supplies. This switch is located in the CCFG_EBICSA register.

In the following example the NAND Flash and the external RAM (DDR2 or LPDDR or 16-bit LPSDR) are in the same power supply range, (NFD0_ON_D16 = 0).

© 2017 Microchip Technology Inc.

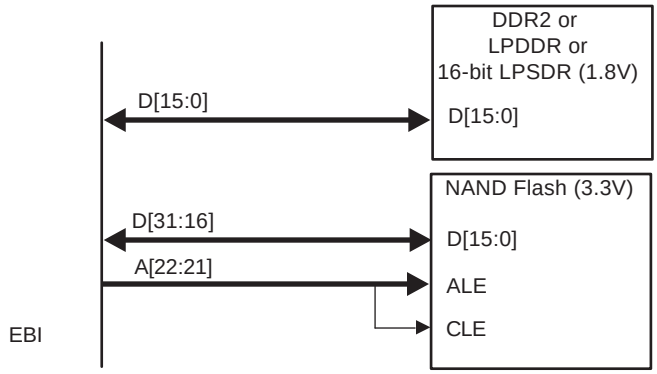
Figure 26-3: NAND Flash and the external RAM power supply (NFD0_ON_D16 = 0)



In the following example the NAND Flash and the external RAM (DDR2 or LPDDR or 16-bit LPSDR) are NOT in the same power supply range (NFD0_ON_D16 = 1).

This can be used if the SMC connects to the NAND Flash only. Using this function with another device on the SMC will lead to an unpredictable behavior of that device. In that case, the default value must be selected.

Figure 26-4: NAND Flash and the external RAM power supply (NFD0_ON_D16 = 1)



At reset NFD0_ON_D16 = 1 and NAND Flash bus is connected to D16–D31.

26.7.5 Static Memory Controller

For information on the Static Memory Controller, refer to Section 29. “Static Memory Controller (SMC)”.

26.7.6 DDR2SDRAM Controller

For information on the DDR2SDR Controller, refer to Section 30. “DDR SDR SDRAM Controller (DDRSDRC)”.

© 2017 Microchip Technology Inc.

26.7.7 Programmable Multi-bit ECC Controller

For information on the PMECC Controller, refer to Section 27. “Programmable Multibit Error Correction Code Controller (PMECC)”.

26.7.8 NAND Flash Support

External Bus Interfaces 1 integrate circuitry that interfaces to NAND Flash devices.

26.7.8.1 External Bus Interface

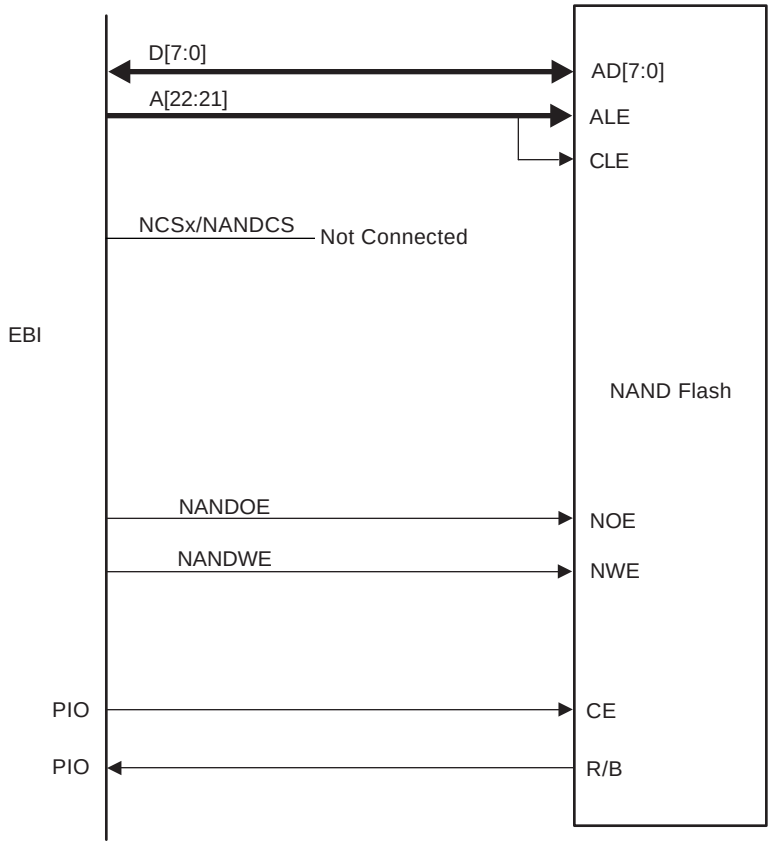
The NAND Flash logic is driven by the SMC on the NCS3 address space. Programming the EBI_CS3A field in the CCFG_EBICSA register in the Chip Configuration User Interface to the appropriate value enables the NAND Flash logic. For details on this register, refer to Section 25. “Bus Matrix (MATRIX)”. Access to an external NAND Flash device is then made by accessing the address space reserved to NCS3 (i.e., between 0x4000 0000 and 0x4FFF FFFF).

The NAND Flash Logic drives the read and write command signals of the SMC on the NANDOE and NANDWE signals when the NCS3 signal is active. NANDOE and NANDWE are invalidated as soon as the transfer address fails to lie in the NCS3 address space. See Figure 26-5 for more information. For details on these waveforms, refer to Section 29. “Static Memory Controller (SMC)”.

26.7.8.2 NAND Flash Signals

The address latch enable and command latch enable signals on the NAND Flash device are driven by address bits A22 and A21 of the EBI address bus. The command, address or data words on the data bus of the NAND Flash device are distinguished by using their address within the NCSx address space. The chip enable (CE) signal of the device and the ready/busy (R/B) signals are connected to PIO lines. The CE signal then remains asserted even when NCSx is not selected, preventing the device from returning to standby mode.

Figure 26-5: NAND Flash Application Example



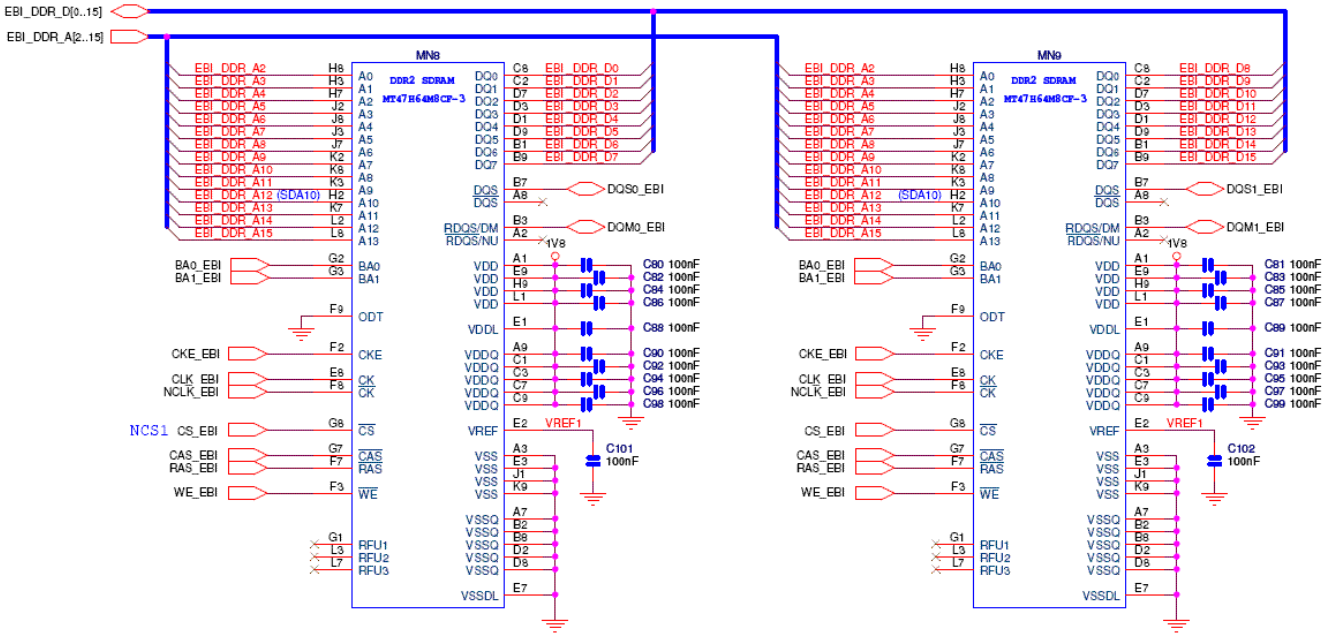
26.8 Implementation Examples

The following hardware configurations are given for illustration only. The user should refer to the memory manufacturer web site to check current device availability.

26.8.1 2x8-bit DDR2 on EBI

26.8.1.1 Hardware Configuration

Figure 26-6: 2x8-bit DDR2 on EBI Configuration



26.8.1.2 Software Configuration

- Assign EBI_CS1 to the DDR2 controller by setting the EBI_CS1A bit in the EBI Chip Select Assignment Register.
- Initialize the DDR2 Controller depending on the DDR2 device and system bus frequency.

The DDR2 initialization sequence is described in Section 30.4.3 “DDR2-SDRAM Initialization”.

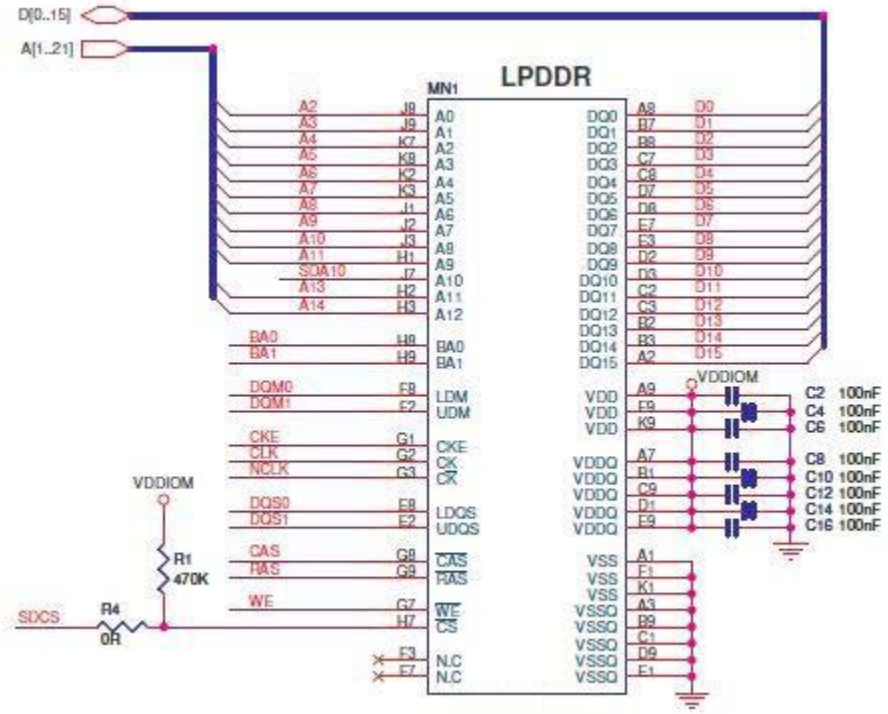
In this case VDDNF can be different from VDDIOM. The NAND Flash device can be 3.3V or 1.8V and wired on D16–D31 data bus. NFD0_ON_D16 is to be set to 1.

© 2017 Microchip Technology Inc.

26.8.2 16-bit LPDDR on EBI

26.8.2.1 Hardware Configuration

Figure 26-7: 16-bit LPDDR on EBI Configuration



26.8.2.2 Software Configuration

The following configuration has to be performed:

- Assign EBI_CS1 to the DDR2 controller by setting the bit EBI_CS1A in the EBI Chip Select Assignment Register.
- Initialize the DDR2 Controller depending on the LPDDR device and system bus frequency.

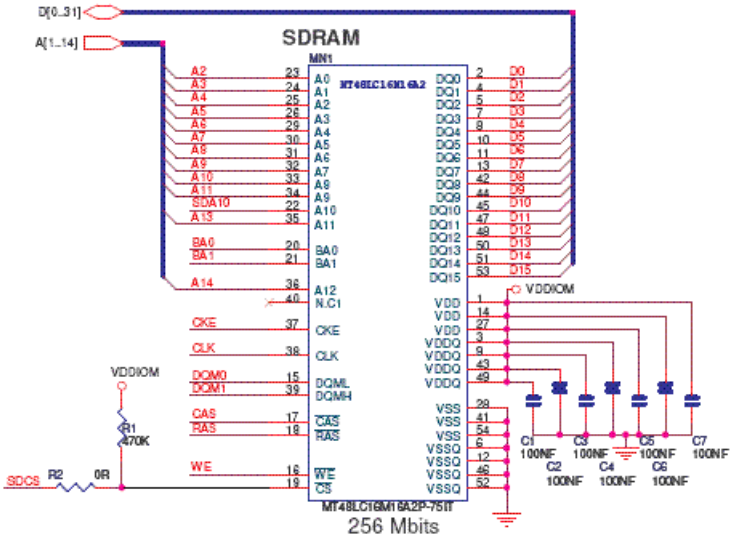
The LPDDR initialization sequence is described in Section 30.4.2 “Low-power DDR1-SDRAM Initialization”.

In this case VDDNF can be different from VDDIOM. The NAND Flash device can be 3.3V or 1.8V and wired on D16–D31 data bus. NFD0_ON_D16 is to be set to 1.

26.8.3 16-bit SDRAM

26.8.3.1 Hardware Configuration

Figure 26-8: 16-bit SDRAM Configuration



26.8.3.2 Software Configuration

The following configuration has to be performed:

- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI_CS1A in the EBI Chip Select Assignment Register.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

The Data Bus Width is to be programmed to 16 bits.

The SDRAM initialization sequence is described in Section 30.4.1 “SDR-SDRAM Initialization”.

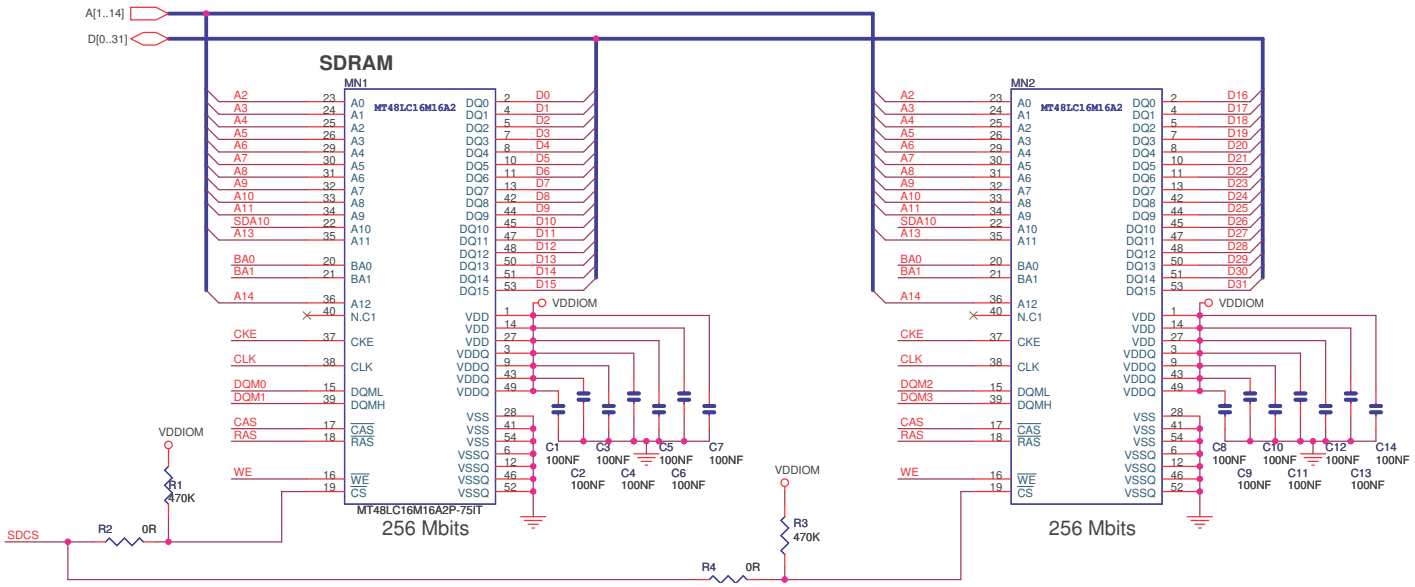
In this case VDDNF can be different from VDDIOM. The NAND Flash device can be 3.3V or 1.8V and wired on D16–D31 data bus. NFD0_ON_D16 is to be set to 1.

© 2017 Microchip Technology Inc.

26.8.4 2x16-bit SDRAM

26.8.4.1 Hardware Configuration

Figure 26-9: 2x16-bit SDRAM Configuration



26.8.4.2 Software Configuration

The following configuration has to be performed:

- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI_CS1A in the EBI Chip Select Assignment Register.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

The Data Bus Width is to be programmed to 32 bits. The data lines D[16..31] are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.

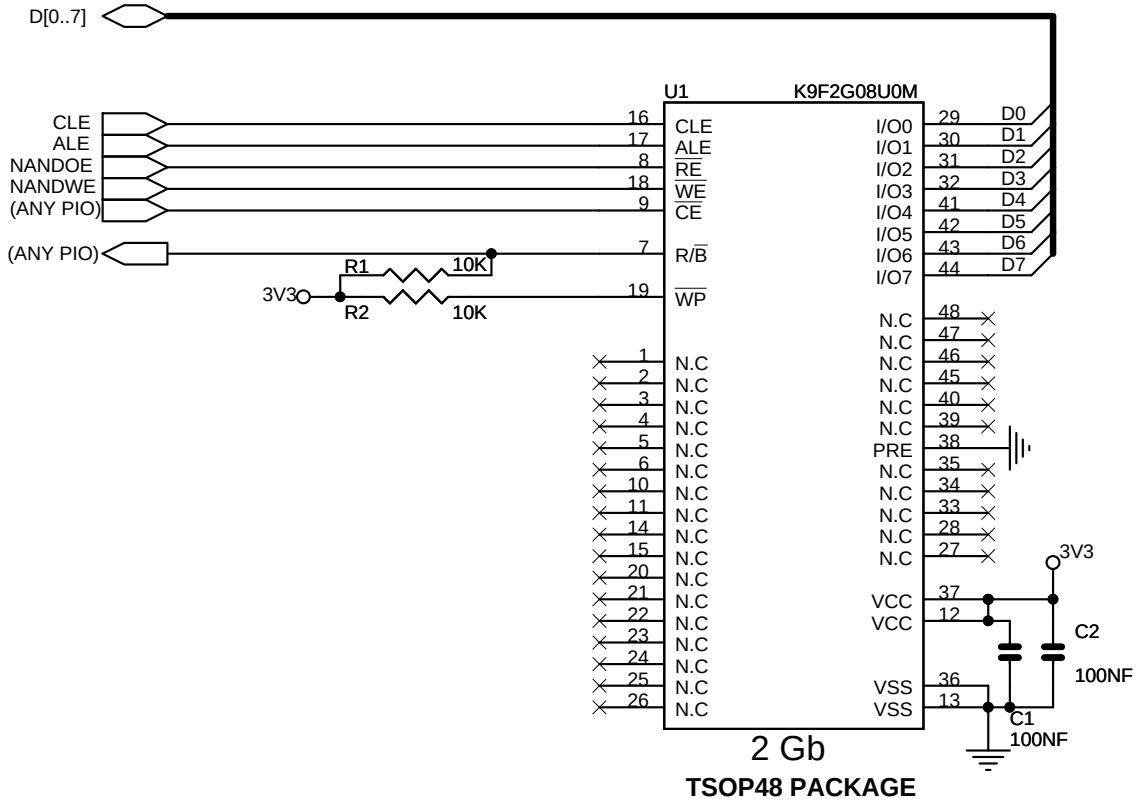
The SDRAM initialization sequence is described in Section 30.4.1 “SDR-SDRAM Initialization”.

In this case, VDDNF must be equal to VDDIOM. The NAND Flash device must be 3.3V and wired on D0–D15 data bus. NFD0_ON_D16 must be set to 0.

26.8.5 8-bit NAND Flash with NFD0_ON_D16 = 0

26.8.5.1 Hardware Configuration

Figure 26-10: 8-bit NAND Flash with NFD0_ON_D16 = 0



26.8.5.2 Software Configuration

The following configuration has to be performed:

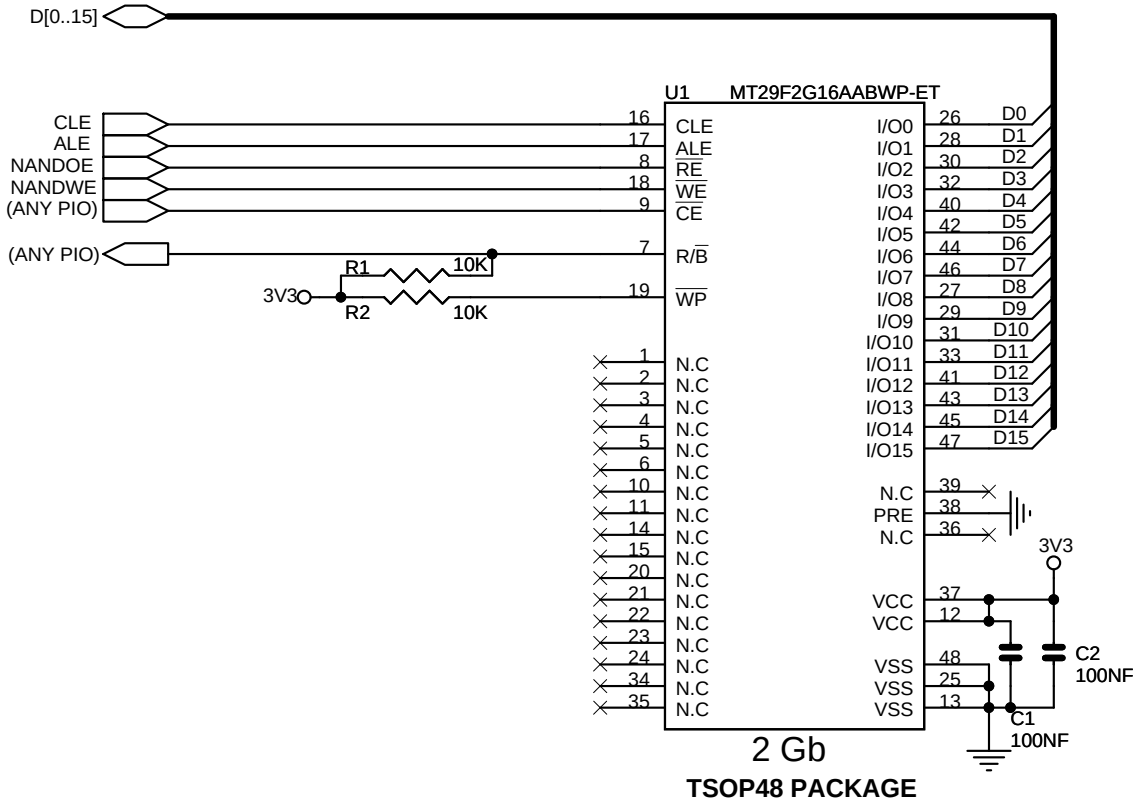
- Set NFD0_ON_D16 = 0 in the EBI Chip Select Assignment Register.
- Assign the EBI CS3 to the NAND Flash by setting the bit EBI_CS3A in the EBI Chip Select Assignment Register.
- Reserve A21/A22 for ALE/CLE functions. Address and Command Latches are controlled respectively by setting to 1 the address bits A21 and A22 during accesses.
- Configure a PIO line as an input to manage the Ready/Busy signal.
- Configure Static Memory Controller CS3 Setup, Pulse, Cycle and Mode accordingly to NAND Flash timings, the data bus width and the system bus frequency.

© 2017 Microchip Technology Inc.

26.8.6 16-bit NAND Flash with NFD0_ON_D16 = 0

26.8.6.1 Hardware Configuration

Figure 26-11: 16-bit NAND Flash with NFD0_ON_D16 = 0



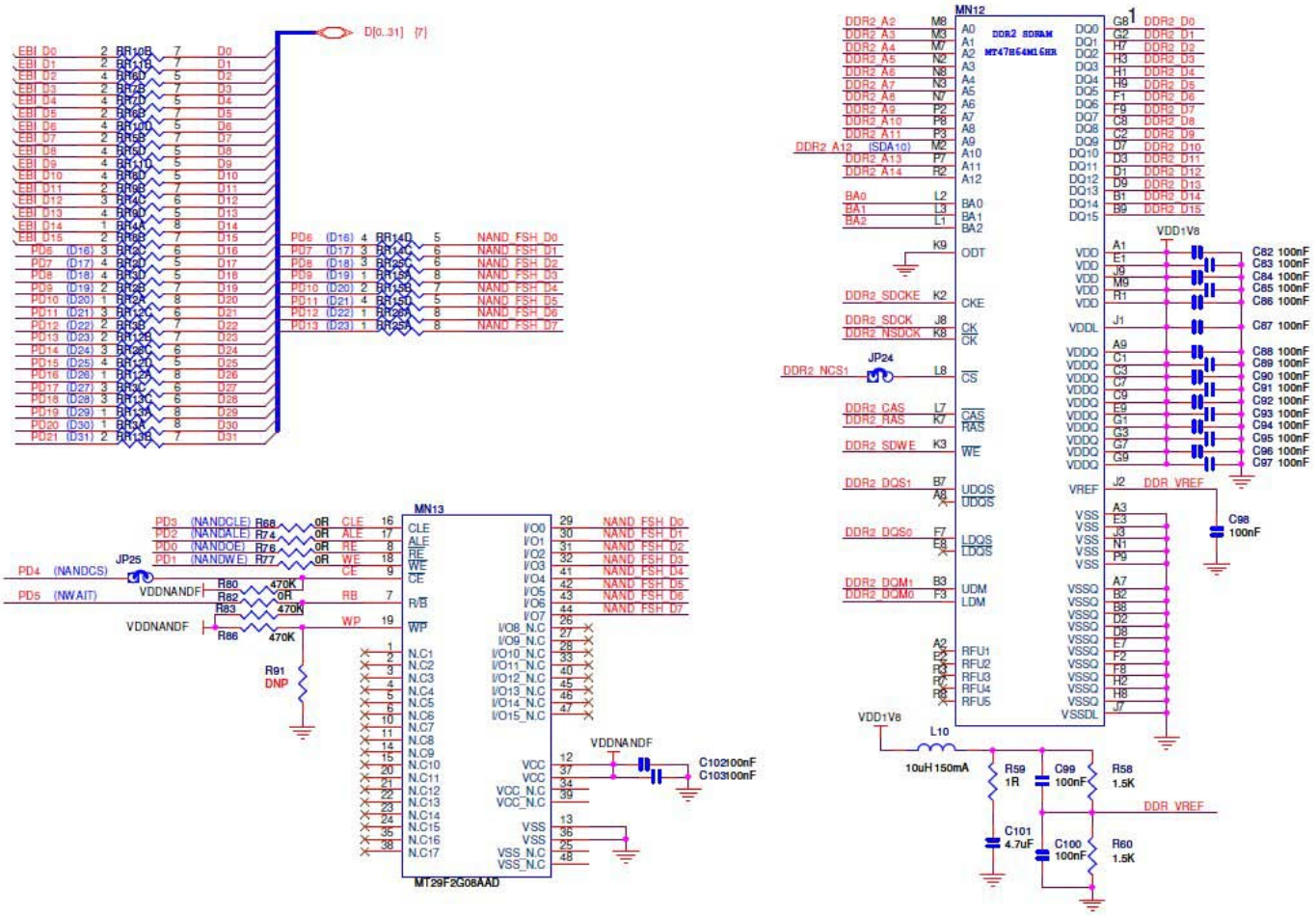
26.8.6.2 Software Configuration

The software configuration is the same as for an 8-bit NAND Flash except for the data bus width programmed in the SMC Mode Register.

26.8.7 8-bit NAND Flash with NFD0_ON_D16 = 1

26.8.7.1 Hardware Configuration

Figure 26-12: 8-bit NAND Flash with NFD0_ON_D16 = 1



26.8.7.2 Software Configuration

The following configuration has to be performed:

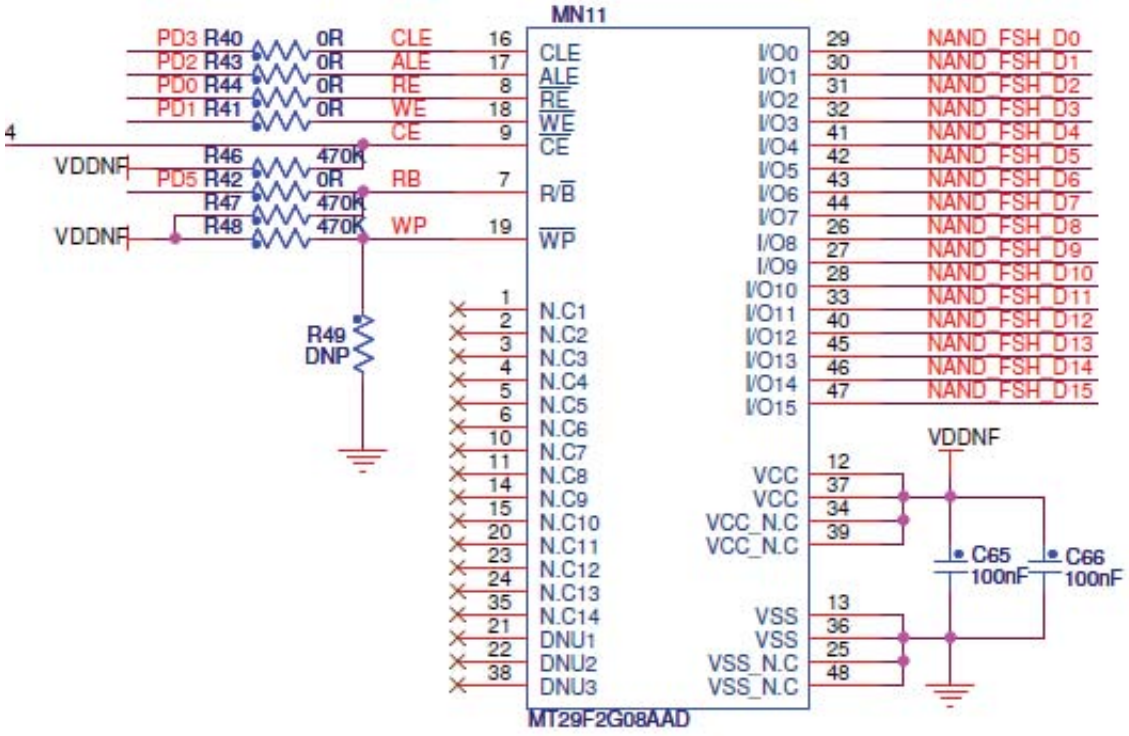
- Set NFD0_ON_D16 = 1 in the EBI Chip Select Assignment Register.
- Assign the EBI CS3 to the NAND Flash by setting the bit EBI_CS3A in the EBI Chip Select Assignment Register.
- Configure the PIOD controller to assign the required PIOD[23..0] to EBI function.
- Reserve A21 / A22 for ALE / CLE functions. Address and Command Latches are controlled respectively by setting to 1 the address bit A21 and A22 during accesses.
- Configure a PIO line as an input to manage the Ready/Busy signal.
- Configure Static Memory Controller CS3 Setup, Pulse, Cycle and Mode accordingly to NAND Flash timings, the data bus width and the system bus frequency.

© 2017 Microchip Technology Inc.

26.8.8 16-bit NAND Flash with NFD0_ON_D16 = 1

26.8.8.1 Hardware Configuration

Figure 26-13: 16-bit NAND Flash with NFD0_ON_D16 = 1



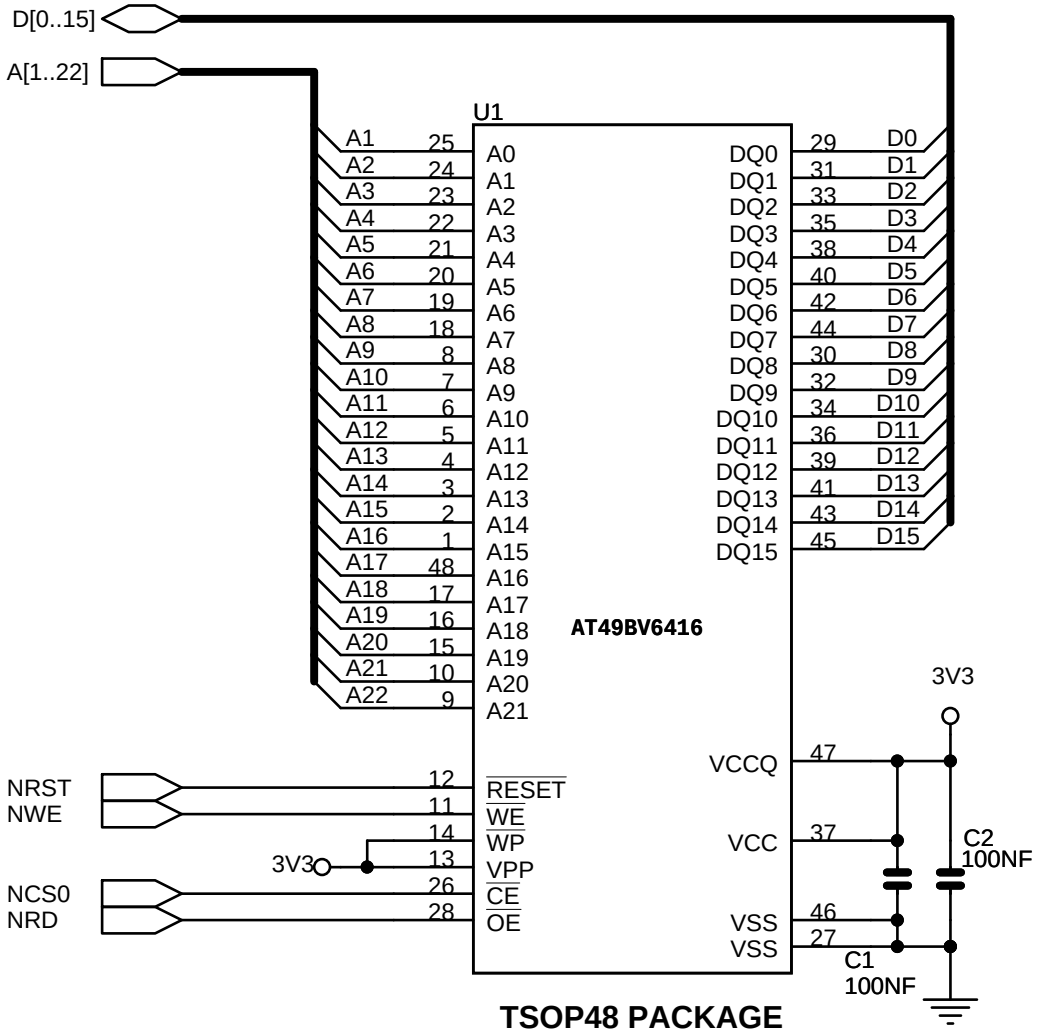
26.8.8.2 Software Configuration

The software configuration is the same as for an 8-bit NAND Flash except for the data bus width programmed in the SMC Mode Register.

26.8.9 NOR Flash on NCS0

26.8.9.1 Hardware Configuration

Figure 26-14: NOR Flash on NCS0



26.8.9.2 Software Configuration

The default configuration for the Static Memory Controller, byte select mode, 16-bit data bus, Read/Write controlled by Chip Select, allows boot on 16-bit non-volatile memory at slow clock.

For another configuration, configure the Static Memory Controller CS0 Setup, Pulse, Cycle and Mode depending on Flash timings and system bus frequency.

© 2017 Microchip Technology Inc.

27. Programmable Multibit Error Correction Code Controller (PMECC)

27.1 Description

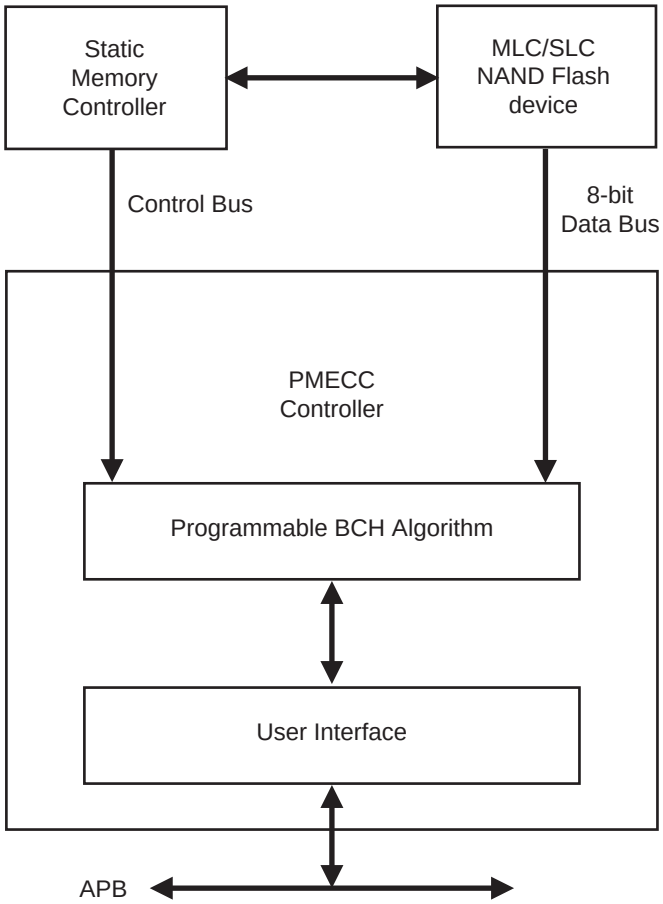
The Programmable Multibit Error Correction Code Controller (PMECC) is a programmable binary BCH (Bose, Chaudhuri and Hocquenghem) encoder/decoder. This controller can be used to generate redundancy information for both Single-Level Cell (SLC) and Multi-level Cell (MLC) NAND Flash devices. It supports redundancy for correction of 2, 4, 8, 12 or 24 bits of error per sector of data.

27.2 Embedded Characteristics

- 8-bit Nand Flash Data Bus Support
- Multibit Error Correcting Code
- Algorithm based on binary shortened Bose, Chaudhuri and Hocquenghem (BCH) codes
- Programmable Error Correcting Capability: 2, 4, 8, 12 and 24 bit of errors per sector
- Programmable Sector Size: 512 bytes or 1024 bytes
- Programmable Number of Sectors per page: 1, 2, 4 or 8 sectors of data per page
- Programmable Spare Area Size
- Supports Spare Area ECC Protection
- Supports 8 Kbytes page size using 1024 bytes per sector and 4 Kbytes page size using 512 bytes per sector
- Configurable through APB interface
- Multibit Error Detection is Interrupt Driven

27.3 Block Diagram

Figure 27-1: Block Diagram



27.4 Functional Description

The NAND Flash sector size is programmable and can be set to 512 bytes or 1024 bytes. The PMECC module generates redundancy at encoding time, when a NAND write page operation is performed. The redundancy is appended to the page and written in the spare area. This operation is performed by the processor. It moves the content of the PMECCx registers into the NAND Flash memory. The number of registers depends on the selected error correction capability, refer to Table 27-1 “Relevant Redundancy Registers”. This operation is executed for each sector. At decoding time, the PMECC module generates the remainder of the received codeword by minimal polynomials. When all polynomial remainders for a given sector are set to zero, no error occurred. When the polynomial remainders are other than zero, the codeword is corrupted and further processing is required.

The PMECC module generates an interrupt indicating that an error occurred. The processor must read the PMECC Interrupt Status Register (PMECC_ISR). This register indicates which sector is corrupted.

To find the error location within a sector, the processor must execute the following decoding steps:

1. Syndrome computation
2. Find the error locator polynomials
3. Find the roots of the error locator polynomial

All decoding steps involve finite field computation. It means that a library of finite field arithmetic must be available to perform addition, multiplication and inversion. The finite field arithmetic operations can be performed through the use of a memory mapped lookup table, or direct software implementation. The software implementation presented is based on lookup tables. Two tables named gf_log and gf_antilog are used. If alpha is the primitive element of the field, then a power of alpha is in the field. Assume beta = alpha ^ index, then beta belongs to the field, and gf_log(beta) = gf_log(alpha ^ index) = index. The gf_antilog tables provide exponent inverse of the element, if beta = alpha ^ index, then gf_antilog(index) = beta.

The first step consists of the syndrome computation. The PMECC module computes the remainders and software must substitute the power of the primitive element.

The procedure implementation is given in Section 27.5.1 “Remainder Substitution Procedure”.

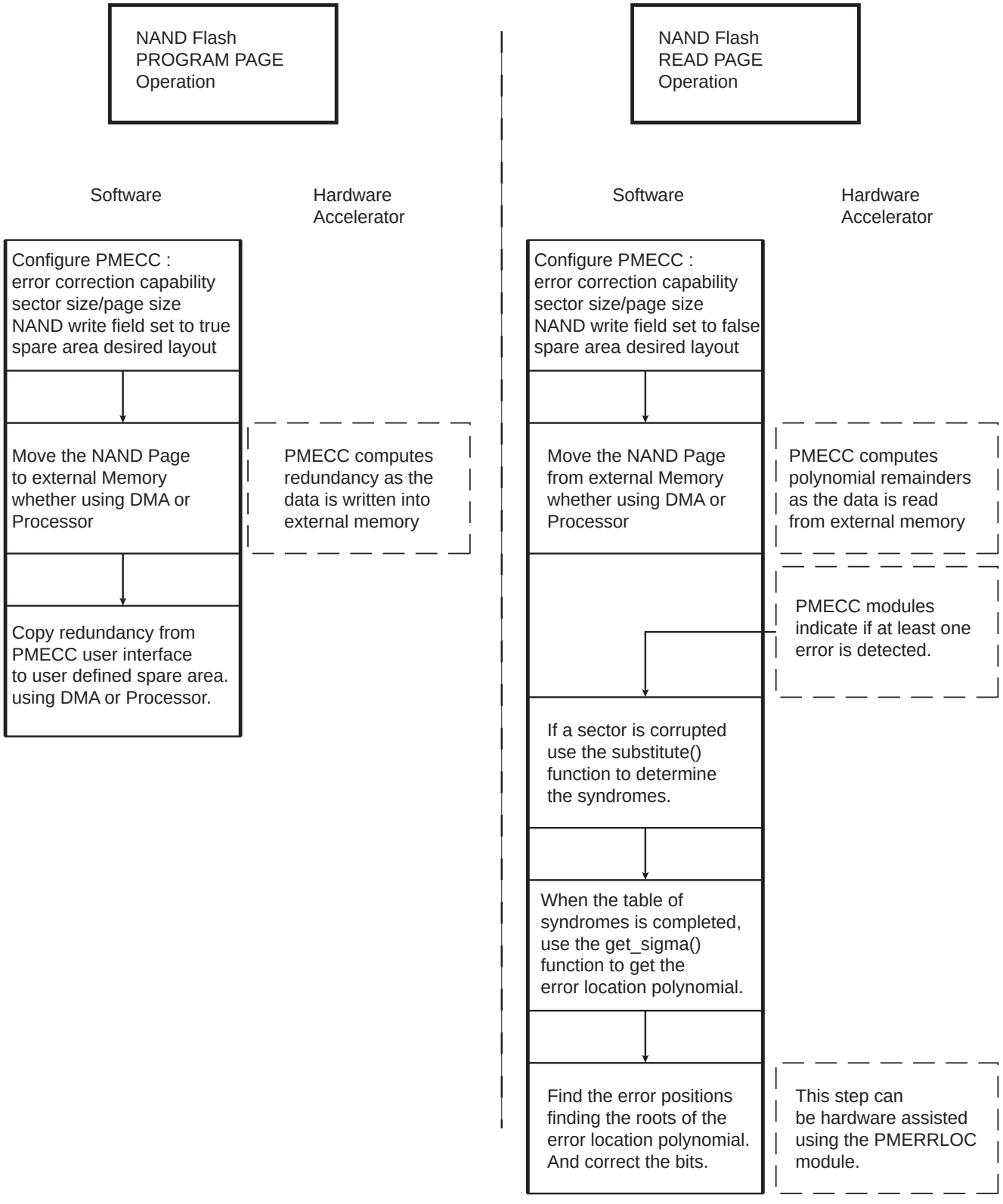
The second step is the most software intensive. It is the Berlekamp’s iterative algorithm for finding the error-location polynomial.

The procedure implementation is given in Section 27.5.2 “Find the Error Location Polynomial Sigma(x)”.

The Last step is finding the root of the error location polynomial. This step can be very software intensive. Indeed, there is no straightforward method of finding the roots, except by evaluating each element of the field in the error location polynomial. However a hardware accelerator can be used to find the roots of the polynomial. The Programmable Multibit Error Correction Code Location (PMERRLOC) module provides this kind of hardware acceleration.

© 2017 Microchip Technology Inc.

Figure 27-2: Software/Hardware Multibit Error Correction Dataflow



27.4.1 MLC/SLC Write Page Operation using PMECC

When an MLC write page operation is performed, the PMECC controller is configured with the NANDWR bit in the PMECC Configuration Register (PMECC_CFG) set to one. When the NAND spare area contains file system information and redundancy (PMECCx), the spare area is error protected, then the SPAREEN in PMECC_CFG is set to one. When the NAND spare area contains only redundancy information, the SPAREEN bit is set to zero.

When the write page operation is terminated, the user writes the redundancy in the NAND spare area. This operation can be done with DMA assistance.

Table 27-1: Relevant Redundancy Registers

BCH_ERR field	Sector size set to 512 bytes	Sector size set to 1024 bytes
0	PMECC_ECC0	PMECC_ECC0
1	PMECC_ECC0, PMECC_ECC1	PMECC_ECC0, PMECC_ECC1
2	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3
3	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3, PMECC_ECC4, PMECC_ECC5, PMECC_ECC6	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3, PMECC_ECC4, PMECC_ECC5, PMECC_ECC6
4	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3, PMECC_ECC4, PMECC_ECC5, PMECC_ECC6, PMECC_ECC7, PMECC_ECC8, PMECC_ECC9	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3, PMECC_ECC4, PMECC_ECC5, PMECC_ECC6, PMECC_ECC7, PMECC_ECC8, PMECC_ECC9, PMECC_ECC10

Table 27-2: Number of relevant ECC bytes per sector, copied from LSbyte to MSbyte

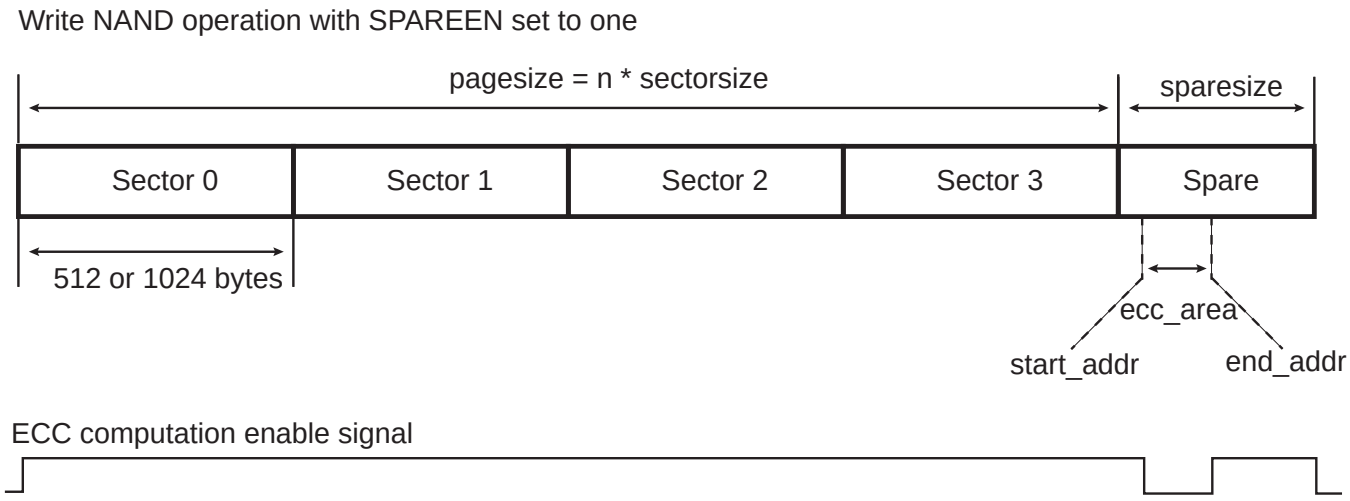
BCH_ERR field	Sector size set to 512 bytes	Sector size set to 1024 bytes
0	4 bytes	4 bytes
1	7 bytes	7 bytes
2	13 bytes	14 bytes
3	20 bytes	21 bytes
4	39 bytes	42 bytes

27.4.1.1 SLC/MLC Write Operation with Spare Enable Bit Set

When the SPAREEN bit in PMECC_CFG is set to one, the spare area of the page is encoded with the stream of data of the last sector of the page. This mode is entered by writing one in the DATA bit in the PMECC Control Register (PMECC_CTRL). When the encoding process is over, the redundancy is written to the spare area in user mode, USER bit in PMECC_CTRL must be set to one.

© 2017 Microchip Technology Inc.

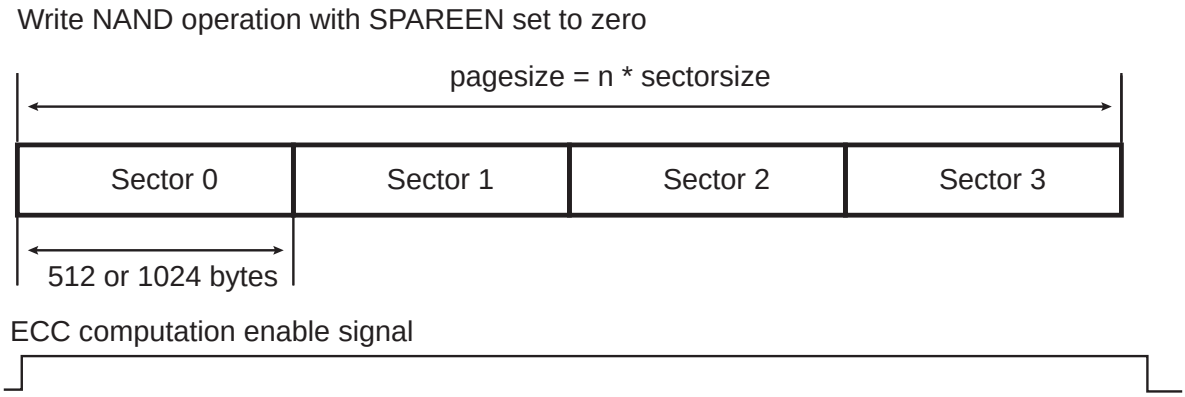
Figure 27-3: NAND Write Operation with Spare Encoding



27.4.1.2 MLC/SLC Write Operation with Spare Area Disabled

When the SPAREEN bit in PMECC_CFG is set to zero the spare area is not encoded with the stream of data. This mode is entered by writing one to the DATA bit in PMECC_CTRL.

Figure 27-4: NAND Write Operation



27.4.2 MLC/SLC Read Page Operation using PMECC

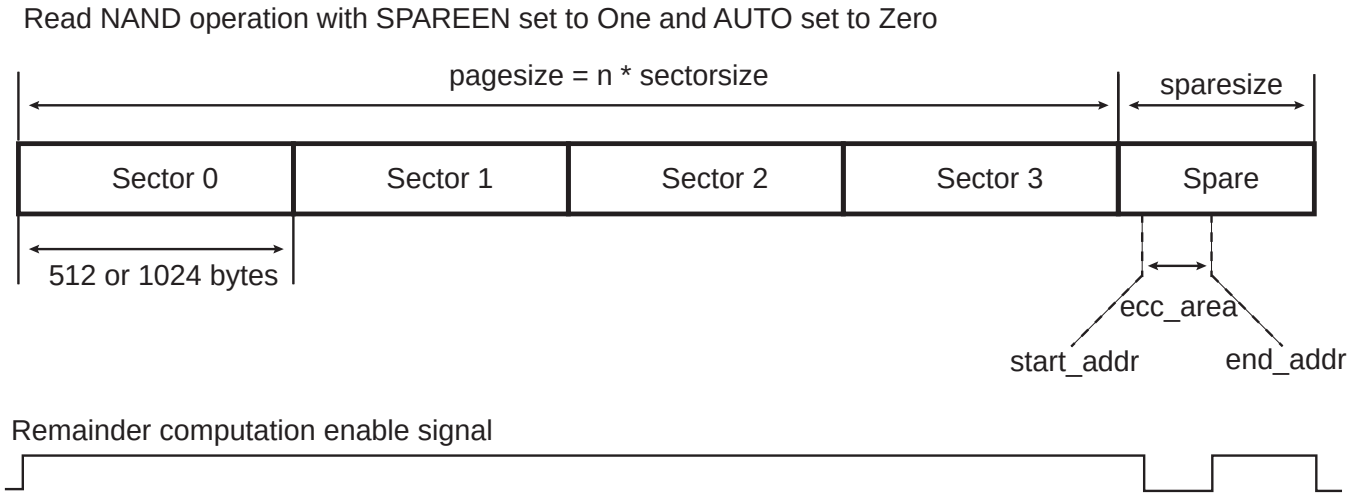
Table 27-3: Relevant Remainders Registers

BCH_ERR field	Sector size set to 512 bytes	Sector size set to 1024 bytes
0	PMECC_REM0	PMECC_REM0
1	PMECC_REM0, PMECC_REM1	PMECC_REM0, PMECC_REM1
2	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3,	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3
3	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3, PMECC_REM4, PMECC_REM5, PMECC_REM6, PMECC_REM7	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3, PMECC_REM4, PMECC_REM5, PMECC_REM6, PMECC_REM7
4	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3, PMECC_REM4, PMECC_REM5, PMECC_REM6, PMECC_REM7, PMECC_REM8, PMECC_REM9, PMECC_REM10, PMECC_REM11	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3, PMECC_REM4, PMECC_REM5, PMECC_REM6, PMECC_REM7, PMECC_REM8, PMECC_REM9, PMECC_REM10, PMECC_REM11

27.4.2.1 MLC/SLC Read Operation with Spare Decoding

When the spare area is protected, the spare area contains valid data. As the redundancy may be included in the middle of the information stream, the user programs the start address and the end address of the ECC area. The controller will automatically skip the ECC area. This mode is entered by writing one in the DATA bit in PMECC_CTRL. When the page has been fully retrieved from NAND, the ECC area is read using the user mode by writing one to the USER bit in PMECC_CTRL.

Figure 27-5: Read Operation with Spare Decoding

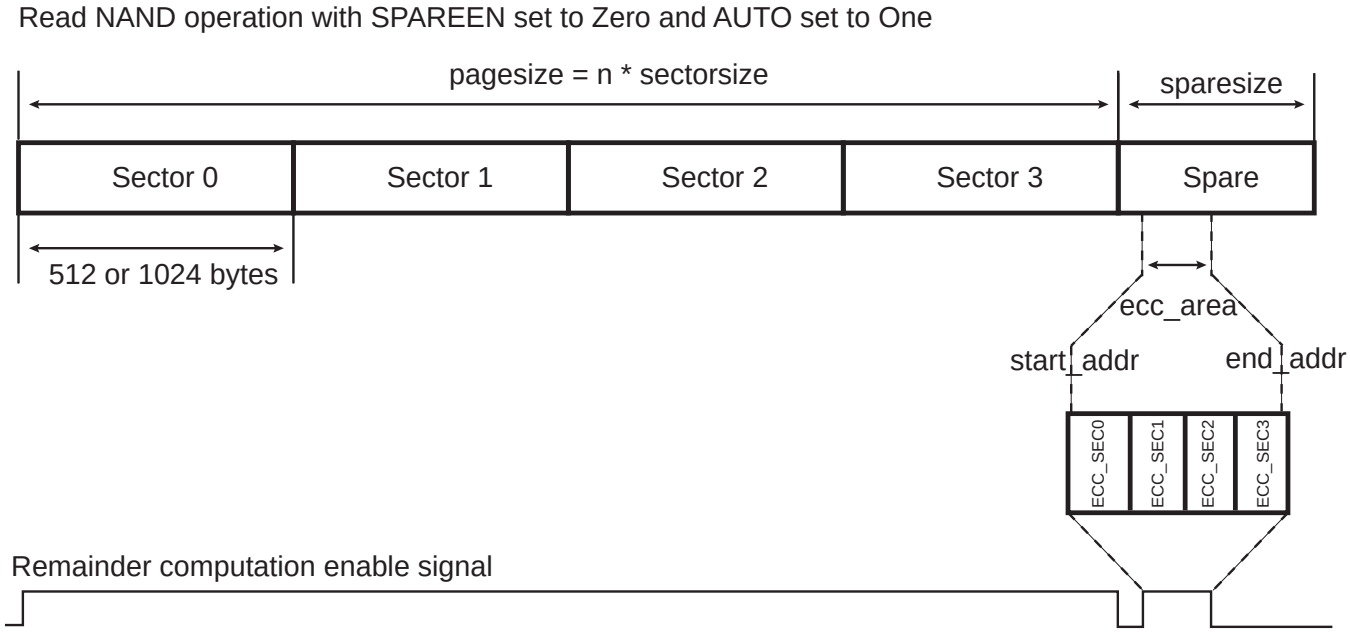


27.4.2.2 MLC/SLC Read Operation

If the spare area is not protected with the error correcting code, the redundancy area is retrieved directly. This mode is entered by writing one in the DATA bit in PMECC_CTRL. When the AUTO bit in PMECC_CFG is set to one the ECC is retrieved automatically, otherwise the ECC must be read using user mode.

© 2017 Microchip Technology Inc.

Figure 27-6: Read Operation

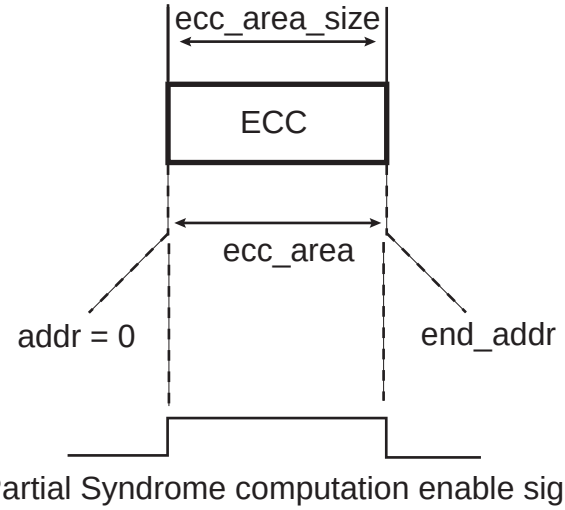


27.4.2.3 MLC/SLC User Read ECC Area

This mode allows a manual retrieve of the ECC.

This mode is entered writing one in the USER bit in PMECC_CTRL.

Figure 27-7: User Read Mode



27.5 Software Implementation

27.5.1 Remainder Substitution Procedure

The substitute function evaluates the polynomial remainder, with different values of the field primitive elements. The finite field arithmetic addition operation is performed with the Exclusive or. The finite field arithmetic multiplication operation is performed through the gf_log, gf_antilog lookup tables.

The REM2NP1 and REMN2NP3 fields of the PMECC Remainder x registers (PMECC_REMx) contain only odd remainders. Each bit indicates whether the coefficient of the polynomial remainder is set to zero or not.

NB_ERROR_MAX defines the maximum value of the error correcting capability.

NB_ERROR defines the error correcting capability selected at encoding/decoding time.

NB_FIELD_ELEMENTS defines the number of elements in the field.

si[] is a table that holds the current syndrome value, an element of that table belongs to the field. This is also a shared variable for the next step of the decoding operation.

oo[] is a table that contains the degree of the remainders.

```
int substitute()
{
int i;
int j;
for (i = 1; i < 2 * NB_ERROR_MAX; i++)
{
    si[i] = 0;
}
for (i = 1; i < 2*NB_ERROR; i++)
{
    for (j = 0; j < oo[i]; j++)
    {
        if (REM2NPX[i][j])
        {
            si[i] = gf_antilog[(i * j)%NB_FIELD_ELEMENTS] ^ si[i];
        }
    }
}
return 0;
}
```


© 2017 Microchip Technology Inc.

27.5.2 Find the Error Location Polynomial Sigma(x)

The sample code below gives a Berlekamp iterative procedure for finding the value of the error location polynomial.

The input of the procedure is the si[] table defined in the remainder substitution procedure.

The output of the procedure is the error location polynomial named smu (sigma mu). The polynomial coefficients belong to the field. The smu[NB_ERROR+1][] is a table that contains all these coefficients.

NB_ERROR_MAX defines the maximum value of the error correcting capability.

NB_ERROR defines the error correcting capability selected at encoding/decoding time.

NB_FIELD_ELEMENTS defines the number of elements in the field.

```
int get_sigma()
{
int i;
int j;
int k;
/* mu          */
int mu[NB_ERROR_MAX+2];
/* sigma ro   */
int sro[2*NB_ERROR_MAX+1];
/* discrepancy */
int dmu[NB_ERROR_MAX+2];
/* delta order   */
int delta[NB_ERROR_MAX+2];
/* index of largest delta */
int ro;
int largest;
int diff;
/*                    */
/*     First Row      */
/*                    */
/* Mu */
mu[0]  = -1; /* Actually -1/2 */
/* Sigma(x) set to 1 */
for (i = 0; i < (2*NB_ERROR_MAX+1); i++)
    smu[0][i] = 0;
smu[0][0] = 1;
/* discrepancy set to 1 */
dmu[0] = 1;
/* polynom order set to 0 */
lmu[0] = 0;
/* delta set to -1 */
delta[0]  = (mu[0] * 2 - lmu[0]) >> 1;
/*                    */
/*     Second Row     */
/*                    */
/* Mu */
mu[1]  = 0;
/* Sigma(x) set to 1 */
for (i = 0; i < (2*NB_ERROR_MAX+1); i++)
    smu[1][i] = 0;
smu[1][0] = 1;
/* discrepancy set to Syndrome 1 */
dmu[1] = si[1];
/* polynom order set to 0 */
lmu[1] = 0;
/* delta set to 0 */
delta[1]  = (mu[1] * 2 - lmu[1]) >> 1;
for (i=1; i <= NB_ERROR; i++)
{
    mu[i+1] = i << 1;
    /*********************************************/
    /*                                           */
    /*                                           */
    /*          Compute Sigma (Mu+1)             */
```

```
    /*          And L(mu)                               */
    /* check if discrepancy is set to 0 */
    if (dmu[i] == 0)
    {
    /* copy polynom */
    for (j=0; j<2*NB_ERROR_MAX+1; j++)
    {
        smu[i+1][j] = smu[i][j];
    }
    /* copy previous polynom order to the next */
    lmu[i+1] = lmu[i];
    }
    else
    {
    ro      = 0;
    largest = -1;
    /* find largest delta with dmu != 0 */
    for (j=0; j<i; j++)
        {
        if (dmu[j])
            {
            if (delta[j] > largest)
                {
                largest = delta[j];
                ro   = j;
                }
            }
        }
    /* initialize signal ro */
    for (k = 0; k < 2*NB_ERROR_MAX+1; k ++)
    {
        sro[k] = 0;
    }
    /* compute difference */
    diff = (mu[i] - mu[ro]);
    /* compute X ^ (2(mu-ro)) */
    for (k = 0; k < (2*NB_ERROR_MAX+1); k ++)
    {
        sro[k+diff] = smu[ro][k];
    }
    /* multiply by dmu * dmu[ro]^-1 */
    for (k = 0; k < 2*NB_ERROR_MAX+1; k ++)
    {
        /* dmu[ro] is not equal to zero by definition */
        /* check that operand are different from 0    */
        if (sro[k] && dmu[i])
        {
            /* galois inverse */
            sro[k] = gf_antilog[(gf_log[dmu[i]] + (NB_FIELD_ELEMENTS-gf_log[dmu[ro]]) + gf_log[sro[k]]) %
NB_FIELD_ELEMENTS];
        }
    }
    /* multiply by dmu * dmu[ro]^-1 */
    for (k = 0; k < 2*NB_ERROR_MAX+1; k++)
    {
        smu[i+1][k] = smu[i][k] ^ sro[k];
        if (smu[i+1][k])
        {
            /* find the order of the polynom */
            lmu[i+1] = k << 1;
        }
    }
}
/*                                                  */
/*                                                  */
```


© 2017 Microchip Technology Inc.

```
/*      End Compute Sigma (Mu+1)                */
/*      And L(mu)                               */
/*********************************************/
/* In either case compute delta */
delta[i+1]  = (mu[i+1] * 2 - lmu[i+1]) >> 1;
/* In either case compute the discrepancy */
for (k = 0 ; k <= (lmu[i+1]>>1); k++)
{
    if (k == 0)
    dmu[i+1] = si[2*(i-1)+3];
    /* check if one operand of the multiplier is null, its index is -1 */
    else if (smu[i+1][k] && si[2*(i-1)+3-k])
    dmu[i+1] = gf_antilog[(gf_log[smu[i+1][k]] + gf_log[si[2*(i-1)+3-k]])%nn] ^ dmu[i+1];
}
}
return 0;
}
```

27.5.3 Find the Error Position

The output of the get_sigma() procedure is a polynomial stored in the smu[NB_ERROR+1][] table. The error position is the roots of that polynomial. The degree of this polynomial is very important information, as it gives the number of errors. The PMERRLOC module provides a hardware accelerator for this step.

27.6 Programmable Multibit Error Correction Code Controller (PMECC) User Interface

Table 27-4: Register Mapping

Offset	Register	Name	Access	Reset
0x000	PMECC Configuration Register	PMECC_CFG	Read/Write	0x00000000
0x004	PMECC Spare Area Size Register	PMECC_SAREA	Read/Write	0x00000000
0x008	PMECC Start Address Register	PMECC_SADDR	Read/Write	0x00000000
0x00C	PMECC End Address Register	PMECC_EADDR	Read/Write	0x00000000
0x010	PMECC Clock Control Register	PMECC_CLK	Read/Write	0x00000000
0x014	PMECC Control Register	PMECC_CTRL	Write-only	–
0x018	PMECC Status Register	PMECC_SR	Read-only	0x00000000
0x01C	PMECC Interrupt Enable register	PMECC_IER	Write-only	–
0x020	PMECC Interrupt Disable Register	PMECC_IDR	Write-only	–
0x024	PMECC Interrupt Mask Register	PMECC_IMR	Read-only	0x00000000
0x028	PMECC Interrupt Status Register	PMECC_ISR	Read-only	0x00000000
0x02C–0x03C	Reserved	–	–	–
0x040+sec_num*(0x40)+0x00	PMECC ECC 0 Register	PMECC_ECC0	Read-only	0x00000000
0x040+sec_num*(0x40)+0x04	PMECC ECC 1 Register	PMECC_ECC1	Read-only	0x00000000
0x040+sec_num*(0x40)+0x08	PMECC ECC 2 Register	PMECC_ECC2	Read-only	0x00000000
0x040+sec_num*(0x40)+0x0C	PMECC ECC 3 Register	PMECC_ECC3	Read-only	0x00000000
0x040+sec_num*(0x40)+0x10	PMECC ECC 4 Register	PMECC_ECC4	Read-only	0x00000000
0x040+sec_num*(0x40)+0x14	PMECC ECC 5 Register	PMECC_ECC5	Read-only	0x00000000
0x040+sec_num*(0x40)+0x18	PMECC ECC 6 Register	PMECC_ECC6	Read-only	0x00000000
0x040+sec_num*(0x40)+0x1C	PMECC ECC 7 Register	PMECC_ECC7	Read-only	0x00000000
0x040+sec_num*(0x40)+0x20	PMECC ECC 8 Register	PMECC_ECC8	Read-only	0x00000000
0x040+sec_num*(0x40)+0x24	PMECC ECC 9 Register	PMECC_ECC9	Read-only	0x00000000
0x040+sec_num*(0x40)+0x28	PMECC ECC 10 Register	PMECC_ECC10	Read-only	0x00000000
0x240+sec_num*(0x40)+0x00	PMECC REM 0 Register	PMECC_REM0	Read-only	0x00000000
0x240+sec_num*(0x40)+0x04	PMECC REM 1 Register	PMECC_REM1	Read-only	0x00000000
0x240+sec_num*(0x40)+0x08	PMECC REM 2 Register	PMECC_REM2	Read-only	0x00000000
0x240+sec_num*(0x40)+0x0C	PMECC REM 3 Register	PMECC_REM3	Read-only	0x00000000
0x240+sec_num*(0x40)+0x10	PMECC REM 4 Register	PMECC_REM4	Read-only	0x00000000
0x240+sec_num*(0x40)+0x14	PMECC REM 5 Register	PMECC_REM5	Read-only	0x00000000
0x240+sec_num*(0x40)+0x18	PMECC REM 6 Register	PMECC_REM6	Read-only	0x00000000
0x240+sec_num*(0x40)+0x1C	PMECC REM 7 Register	PMECC_REM7	Read-only	0x00000000
0x240+sec_num*(0x40)+0x20	PMECC REM 8 Register	PMECC_REM8	Read-only	0x00000000
0x240+sec_num*(0x40)+0x24	PMECC REM 9 Register	PMECC_REM9	Read-only	0x00000000
0x240+sec_num*(0x40)+0x28	PMECC REM 10 Register	PMECC_REM10	Read-only	0x00000000
0x240+sec_num*(0x40)+0x2C	PMECC REM 11 Register	PMECC_REM11	Read-only	0x00000000

© 2017 Microchip Technology Inc.

Table 27-4: Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x440–0x5FC	Reserved	–	–	–

27.6.1 PMECC Configuration Register

Name: PMECC_CFG

Address:0xFFFFE000

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	AUTO	–	–	–	SPAREEN
15	14	13	12	11	10	9	8
–	–	–	NANDWR	–	–	PAGESIZE	
7	6	5	4	3	2	1	0
–	–	–	SECTORSZ	–	BCH_ERR		

BCH_ERR: Error Correct Capability

Value	Name	Description
0	BCH_ERR2	2 errors
1	BCH_ERR4	4 errors
2	BCH_ERR8	8 errors
3	BCH_ERR12	12 errors
4	BCH_ERR24	24 errors

SECTORSZ: Sector Size

0: The ECC computation is based on a sector of 512 bytes.

1: The ECC computation is based on a sector of 1024 bytes.

PAGESIZE: Number of Sectors in the Page

Value	Name	Description
0	PAGESIZE_1SEC	1 sector for main area (512 or 1024 bytes)
1	PAGESIZE_2SEC	2 sectors for main area (1024 or 2048 bytes)
2	PAGESIZE_4SEC	4 sectors for main area (2048 or 4096 bytes)
3	PAGESIZE_8SEC	8 errors for main area (4096 or 8192 bytes)

NANDWR: NAND Write Access

:0: NAND read access

1: NAND write access

SPAREEN: Spare Enable

– for NAND write access:

0: The spare area is skipped

1: The spare area is protected with the last sector of data.

– for NAND read access:

0: The spare area is skipped.

1: The spare area contains protected data or only redundancy information.

© 2017 Microchip Technology Inc.

AUTO: Automatic Mode Enable

This bit is only relevant in NAND Read Mode, when spare enable is activated.

0: Indicates that the spare area is not protected. In that case the ECC computation takes into account the ECC area located in the spare area. (within the start address and the end address).

1: Indicates that the spare is error protected. In this case, the ECC computation takes into account the whole spare area minus the ECC area in the ECC computation operation.

27.6.2 PMECC Spare Area Size Register

Name: PMECC_SAREA

Address:0xFFFFE004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	SPARESIZE
7	6	5	4	3	2	1	0
SPARESIZE							

SPARESIZE: Spare Area Size

The spare area size is equal to (SPARESIZE+1) bytes.

© 2017 Microchip Technology Inc.

27.6.3 PMECC Start Address Register

Name: PMECC_SADDR

Address:0xFFFFE008

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	STARTADDR
7	6	5	4	3	2	1	0
STARTADDR							

STARTADDR: ECC Area Start Address (byte oriented address)

This field indicates the first byte address of the ECC area. Location 0 matches the first byte of the spare area.

27.6.4 PMECC End Address Register

Name: PMECC_EADDR

Address:0xFFFFE00C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	ENDADDR
7	6	5	4	3	2	1	0
ENDADDR							

ENDADDR: ECC Area End Address (byte oriented address)

This field indicates the last byte address of the ECC area.

© 2017 Microchip Technology Inc.

27.6.5 PMECC Clock Control Register

Name: PMECC_CLK

Address:0xFFFFE010

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	CLKCTRL		

CLKCTRL: Clock Control Register

The PMECC Module data path Setup Time is set to CLKCTRL+1.

This field indicates the database setup times in number of clock cycles. At 133 MHz, this field must be programmed with 2, indicating that the setup time is 3 clock cycles.

27.6.6 PMECC Control Register

Name: PMECC_CTRL

Address:0xFFFFE014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	DISABLE	ENABLE	–	USER	DATA	RST

RST: Reset the PMECC Module

When set to one, this bit resets the PMECC controller; configuration registers remain unaffected.

DATA: Start a Data Phase

USER: Start a User Mode Phase

ENABLE: PMECC Module Enable

PMECC module must always be configured before being activated.

DISABLE: PMECC Module Disable

PMECC module must always be configured after being deactivated.

© 2017 Microchip Technology Inc.

27.6.7 PMECC Status Register

Name: PMECC_SR

Address:0xFFFFE018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	ENABLE	–	–	–	BUSY

BUSY: The Kernel of the PMECC is Busy

ENABLE: PMECC Module Status

0: The PMECC module is disabled and can be configured.

1: The PMECC module is enabled and the configuration registers cannot be written.

27.6.8 PMECC Interrupt Enable Register

Name: PMECC_IER

Address:0xFFFFE01C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	ERRIE

ERRIE: Error Interrupt Enable

© 2017 Microchip Technology Inc.

27.6.9 PMECC Interrupt Disable Register

Name: PMECC_IDR

Address:0xFFFFE020

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	ERRID

ERRID: Error Interrupt Disable

27.6.10 PMECC Interrupt Mask Register

Name: PMECC_IMR

Address:0xFFFFE024

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	ERRIM

ERRIM: Error Interrupt Mask

© 2017 Microchip Technology Inc.

27.6.11 PMECC Interrupt Status Register

Name: PMECC_ISR

Address:0xFFFFE028

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ERRIS							

ERRIS: Error Interrupt Status

When set to one, bit *i* of the PMECC_ISR indicates that sector *i* is corrupted.

27.6.12 PMECC ECC x Register

Name: PMECC_ECCx [x=0..10] [sec_num=0..7]

Address:0xFFFFE040 [0][0] .. 0xFFFFE068 [10][0]
0xFFFFE080 [0][1] .. 0xFFFFE0A8 [10][1]
0xFFFFE0C0 [0][2] .. 0xFFFFE0E8 [10][2]
0xFFFFE100 [0][3] .. 0xFFFFE128 [10][3]
0xFFFFE140 [0][4] .. 0xFFFFE168 [10][4]
0xFFFFE180 [0][5] .. 0xFFFFE1A8 [10][5]
0xFFFFE1C0 [0][6] .. 0xFFFFE1E8 [10][6]
0xFFFFE200 [0][7] .. 0xFFFFE228 [10][7]

Access: Read-only

31	30	29	28	27	26	25	24
ECC							
23	22	21	20	19	18	17	16
ECC							
15	14	13	12	11	10	9	8
ECC							
7	6	5	4	3	2	1	0
ECC							

ECC: BCH Redundancy

This register contains the remainder of the division of the codeword by the generator polynomial.

© 2017 Microchip Technology Inc.

27.6.13 PMECC Remainder x Register

Name: PMECC_REMx [x=0..11] [sec_num=0..7]

Address:0xFFFFE240 [0][0] .. 0xFFFFE26C [11][0]
0xFFFFE280 [0][1] .. 0xFFFFE2AC [11][1]
0xFFFFE2C0 [0][2] .. 0xFFFFE2EC [11][2]
0xFFFFE300 [0][3] .. 0xFFFFE32C [11][3]
0xFFFFE340 [0][4] .. 0xFFFFE36C [11][4]
0xFFFFE380 [0][5] .. 0xFFFFE3AC [11][5]
0xFFFFE3C0 [0][6] .. 0xFFFFE3EC [11][6]
0xFFFFE400 [0][7] .. 0xFFFFE42C [11][7]

Access: Read-only

31	30	29	28	27	26	25	24
–	–	REM2NP3					
23	22	21	20	19	18	17	16
REM2NP3							
15	14	13	12	11	10	9	8
–	–	REM2NP1					
7	6	5	4	3	2	1	0
REM2NP1							

REM2NP1: BCH Remainder 2 * N + 1

When sector size is set to 512 bytes, bit REM2NP1[13] is not used and read as zero.

If bit *i* of the REM2NP1 field is set to one then the coefficient of the X ^ *i* is set to one, otherwise the coefficient is zero.

REM2NP3: BCH Remainder 2 * N + 3

When sector size is set to 512 bytes, bit REM2NP3[29] is not used and read as zero.

If bit *i* of the REM2NP3 field is set to one then the coefficient of the X ^ *i* is set to one, otherwise the coefficient is zero.

© 2017 Microchip Technology Inc.

28. Programmable Multibit ECC Error Location Controller (PMERRLOC)

28.1 Description

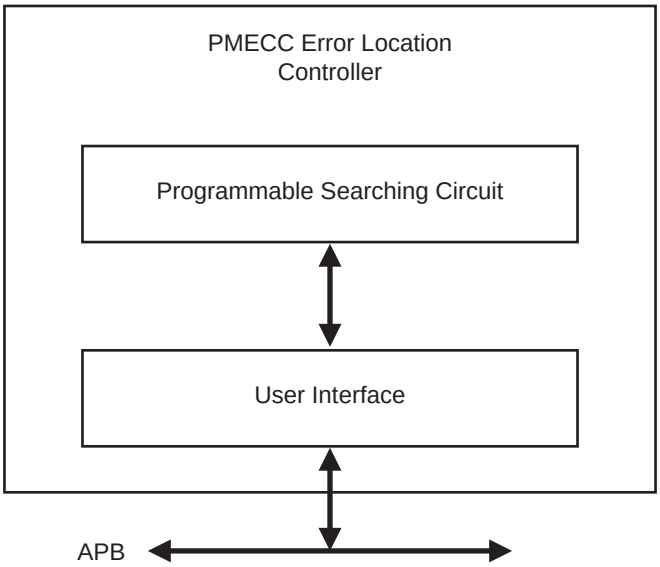
The PMECC Error Location Controller (PMERRLOC) provides hardware acceleration for determining roots of polynomials over two finite fields: GF(2^13) and GF(2^14). It integrates 24 fully programmable coefficients. These coefficients belong to GF(2^13) or GF(2^14). The coefficient programmed in the PMERRLOC_SIGMAx register is the coefficient of degree x in the polynomial.

28.2 Embedded Characteristics

- Provides Hardware Acceleration for determining roots of polynomials defined over a finite field
- Programmable Finite Field GF(2^13) or GF(2^14)
- Finds Roots of Error Locator Polynomial
- Programmable Number of Roots

28.3 Block Diagram

Figure 28-1: Block Diagram



© 2017 Microchip Technology Inc.

28.4 Functional Description

The PMERRLOC search operation is started as soon as a write access is detected in the ELEN register and can be disabled by writing to the ELDIS register. The ENINIT field of the ELEN register shall be initialized with the number of Galois field elements to test. The set of the roots can be limited to a valid range.

Table 28-1: ENINIT field value for a sector size of 512 bytes

Error Correcting Capability	ENINIT Value
2	4122
4	4148
8	4200
12	4252
24	4408

Table 28-2: ENINIT field value for a sector size of 1024 bytes

Error Correcting Capability	ENINIT Value
2	8220
4	8248
8	8304
12	8360
24	8528

When the PMEERRLOC engine is searching for roots the BUSY field of the ELSR remains asserted. An interrupt is asserted at the end of the computation, and the DONE bit of the ELSIR register is set. The ERR_CNT field of the ELISR indicates the number of errors. The error position can be read in the PMERRLOCx registers.

28.5 Programmable Multibit ECC Error Location (PMERRLOC) User Interface

Table 28-3: Register Mapping

Offset	Register	Name	Access	Reset
0x000	Error Location Configuration Register	PMERRLOC_ELCFG	Read/Write	0x00000000
0x004	Error Location Primitive Register	PMERRLOC_ELPRIM	Read-only	0x00000000
0x008	Error Location Enable Register	PMERRLOC_ELEN	Read/Write	0x00000000
0x00C	Error Location Disable Register	PMERRLOC_ELDIS	Read/Write	0x00000000
0x010	Error Location Status Register	PMERRLOC_ELSR	Read/Write	0x00000000
0x014	Error Location Interrupt Enable register	PMERRLOC_ELIER	Read-only	0x00000000
0x018	Error Location Interrupt Disable Register	PMERRLOC_ELIDR	Read-only	0x00000000
0x01C	Error Location Interrupt Mask Register	PMERRLOC_ELIMR	Read-only	0x00000000
0x020	Error Location Interrupt Status Register	PMERRLOC_ELISR	Read-only	0x00000000
0x024	Reserved	–	–	–
0x028	PMECC SIGMA 0 Register	PMERRLOC_SIGMA0	Read/Write	0x00000000
...	...	...	...	...
0x088	PMECC SIGMA 24 Register	PMERRLOC_SIGMA24	Read/Write	0x00000000
0x08C	PMECC Error Location 0 Register	PMERRLOC_EL0	Read-only	0x00000000
...	...	...	...	...
0x0E4	PMECC Error Location 23 Register	PMERRLOC_EL23	Read-only	0x00000000
0xE8–0x1FC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

28.5.1 Error Location Configuration Register

Name: PMERRLOC_ELCFG

Address:0xFFFFE600

Access: Read/Write

Reset:0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	ERRNUM				
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SECTORSZ

ERRNUM: Number of Errors

SECTORSZ: Sector Size

0: The ECC computation is based on a 512-byte sector.

1: The ECC computation is based on a 1024-byte sector.

28.5.2 Error Location Primitive Register

Name: PMERRLOC_ELPRIM

Address:0xFFFFE604

Access: Read-only

Reset:0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
PRIMITIV							
7	6	5	4	3	2	1	0
PRIMITIV							

PRIMITIV: Primitive Polynomial

© 2017 Microchip Technology Inc.

28.5.3 Error Location Enable Register

Name: PMERRLOC_ELEN

Address:0xFFFFE608

Access: Read/Write

Reset:0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	ENINIT					
7	6	5	4	3	2	1	0
ENINIT							

ENINIT: Initial Number of Bits in the Codeword

28.5.4 Error Location Disable Register

Name: PMERRLOC_ELDIS

Address:0xFFFFE60C

Access: Read/Write

Reset:0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DIS

DIS: Disable Error Location Engine

© 2017 Microchip Technology Inc.

28.5.5 Error Location Status Register

Name: PMERRLOC_ELSR

Address:0xFFFFE610

Access: Read/Write

Reset:0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	BUSY

BUSY: Error Location Engine Busy

28.5.6 Error Location Interrupt Enable Register

Name: PMERRLOC_ELIER

Address:0xFFFFE614

Access: Read-only

Reset:0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DONE

DONE: Computation Terminated Interrupt Enable

© 2017 Microchip Technology Inc.

28.5.7 Error Location Interrupt Disable Register

Name: PMERRLOC_ELIDR

Address:0xFFFFE618

Access: Read-only

Reset:0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DONE

DONE: Computation Terminated Interrupt Disable

28.5.8 Error Location Interrupt Mask Register

Name: PMERRLOC_ELIMR

Address:0xFFFFE61C

Access: Read-only

Reset:0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DONE

DONE: Computation Terminated Interrupt Mask

© 2017 Microchip Technology Inc.

28.5.9 Error Location Interrupt Status Register

Name: PMERRLOC_ELISR

Address:0xFFFFE620

Access: Read-only

Reset:0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	ERR_CNT				
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DONE

DONE: Computation Terminated Interrupt Status

ERR_CNT: Error Counter Value

28.5.10 Error Location SIGMAx Register

Name: PMERRLOC_SIGMAx [x=0..24]

Address:0xFFFFE628 [0] .. 0xFFFFE688 [24]

Access: Read/Write

Reset:0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	SIGMAx					
7	6	5	4	3	2	1	0
SIGMAx							

SIGMAx: Coefficient of Degree x in the SIGMA Polynomial.

SIGMAx belongs to the finite field GF(2^13) when the sector size is set to 512 bytes.

SIGMAx belongs to the finite field GF(2^14) when the sector size is set to 1024 bytes.

© 2017 Microchip Technology Inc.

28.5.11 PMECC Error Locationx Register

Name: PMERRLOC_ELx [x=0..23]

Address:0xFFFFE68C

Access: Read-only

Reset:0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	ERRLOCN					
7	6	5	4	3	2	1	0
ERRLOCN							

ERRLOCN: Error Position within the Set {sector area, spare area}.

ERRLOCN points to 0 when the first bit of the main area is corrupted.

If the sector size is set to 512 bytes, the ERRLOCN points to 4096 when the last bit of the sector area is corrupted.

If the sector size is set to 1024 bytes, the ERRLOCN points to 8192 when the last bit of the sector area is corrupted.

If the sector size is set to 512 bytes, the ERRLOCN points to 4097 when the first bit of the spare area is corrupted.

If the sector size is set to 1024 bytes, the ERRLOCN points to 8193 when the first bit of the spare area is corrupted.

29. Static Memory Controller (SMC)

29.1 Description

The Static Memory Controller (SMC) generates the signals that control the access to the external memory devices or peripheral devices. It has 6 Chip Selects and a 26-bit address bus. The 32-bit data bus can be configured to interface with 8-, 16-, or 32-bit external devices. Separate read and write control signals allow for direct memory and peripheral interfacing. Read and write signal waveforms are fully parametrizable.

The SMC can manage wait requests from external devices to extend the current access. The SMC is provided with an automatic slow clock mode. In slow clock mode, it switches from user-programmed waveforms to slow-rate specific waveforms on read and write signals. The SMC supports asynchronous burst read in page mode access for page size up to 32 bytes.

29.2 Embedded Characteristics

- 6 Chip Selects Available
- 64-Mbyte Address Space per Chip Select
- 8-, 16- or 32-bit Data Bus
- Word, Halfword, Byte Transfers
- Byte Write or Byte Select Lines
- Programmable Setup, Pulse And Hold Time for Read Signals per Chip Select
- Programmable Setup, Pulse And Hold Time for Write Signals per Chip Select
- Programmable Data Float Time per Chip Select
- Compliant with LCD Module
- External Wait Request
- Automatic Switch to Slow Clock Mode
- Asynchronous Read in Page Mode Supported: Page Size Ranges from 4 to 32 Bytes

© 2017 Microchip Technology Inc.

29.3 I/O Lines Description

Table 29-1: I/O Line Description

Name	Description	Type	Active Level
NCS[7:0]	Static Memory Controller Chip Select Lines	Output	Low
NRD	Read Signal	Output	Low
NWR0/NWE	Write 0/Write Enable Signal	Output	Low
A0/NBS0	Address Bit 0/Byte 0 Select Signal	Output	Low
NWR1/NBS1	Write 1/Byte 1 Select Signal	Output	Low
A1/NWR2/NBS2	Address Bit 1/Write 2/Byte 2 Select Signal	Output	Low
NWR3/NBS3	Write 3/Byte 3 Select Signal	Output	Low
A[25:2]	Address Bus	Output	–
D[31:0]	Data Bus	I/O	–
NWAIT	External Wait Signal	Input	Low

29.4 Multiplexed Signals

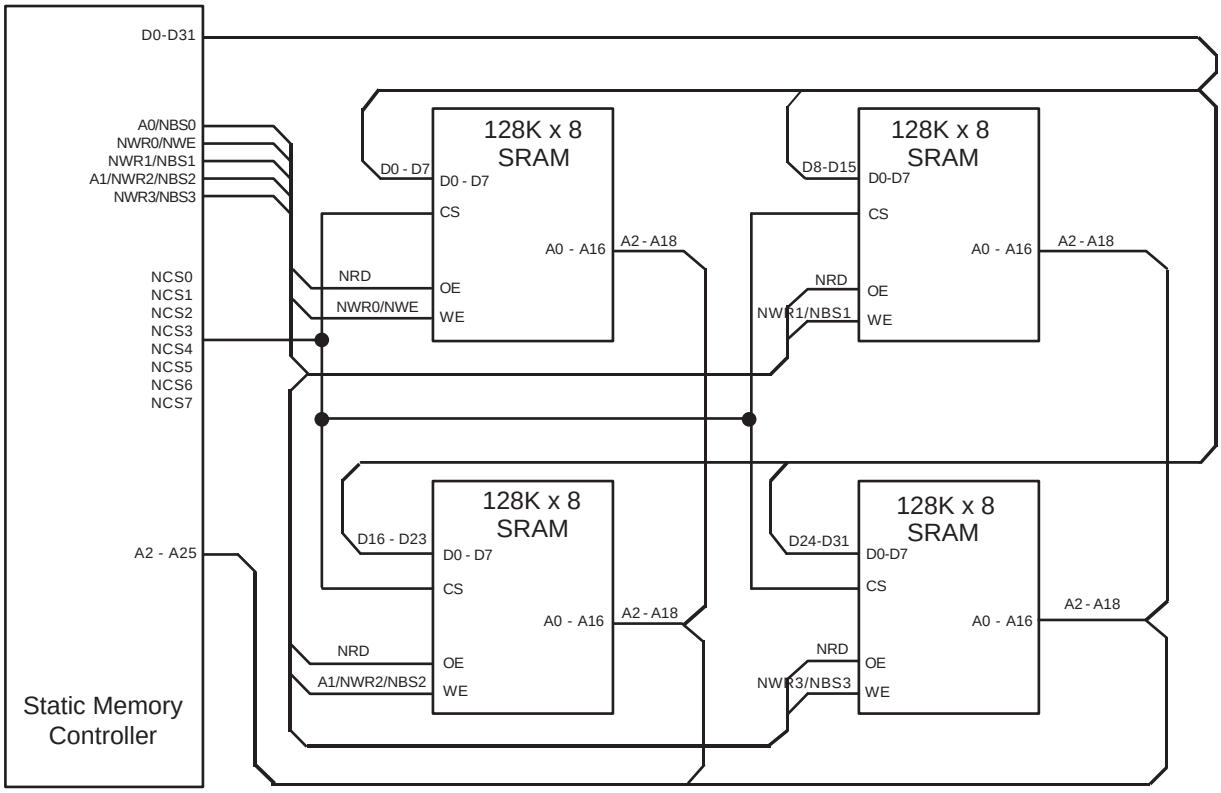
Table 29-2: Static Memory Controller (SMC) Multiplexed Signals

Multiplexed Signals			Related Function
NWR0	NWE	–	Byte-write or byte-select access, see “Byte Write or Byte Select Access”
A0	NBS0	–	8-bit or 16-/32-bit data bus, see “Data Bus Width”
NWR1	NBS1	–	Byte-write or byte-select access see “Byte Write or Byte Select Access”
A1	NWR2	NBS2	8-/16-bit or 32-bit data bus, see “Data Bus Width” . Byte-write or byte-select access, see “Byte Write or Byte Select Access”
NWR3	NBS3	–	Byte-write or byte-select access see “Byte Write or Byte Select Access”

29.5 Application Example

29.5.1 Hardware Interface

Figure 29-1: SMC Connections to Static Memory Devices



29.6 Product Dependencies

29.6.1 I/O Lines

The pins used for interfacing the Static Memory Controller may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the Static Memory Controller pins to their peripheral function. If I/O Lines of the SMC are not used by the application, they can be used for other purposes by the PIO Controller.

Table 29-3: I/O Lines

Instance	Signal	I/O Line	Peripheral
SMC	A20	PD15	B
SMC	A21/NANDALE	PD2	A
SMC	A22/NANDCLE	PD3	A
SMC	A23	PD16	B
SMC	A24	PD17	B
SMC	A25	PD18	B
SMC	D16	PD6	A
SMC	D17	PD7	A
SMC	D18	PD8	A
SMC	D19	PD9	A
SMC	D20	PD10	A
SMC	D21	PD11	A

© 2017 Microchip Technology Inc.

Table 29-3: I/O Lines (Continued)

SMC	D22	PD12	A
SMC	D23	PD13	A
SMC	D24	PD14	A
SMC	D25	PD15	A
SMC	D26	PD16	A
SMC	D27	PD17	A
SMC	D28	PD18	A
SMC	D29	PD19	A
SMC	D30	PD20	A
SMC	D31	PD21	A
SMC	NANDOE	PD0	A
SMC	NANDWE	PD1	A
SMC	NCS2	PD19	B
SMC	NCS3	PD4	A
SMC	NCS4	PD20	B
SMC	NCS5	PD21	B
SMC	NWAIT	PD5	A

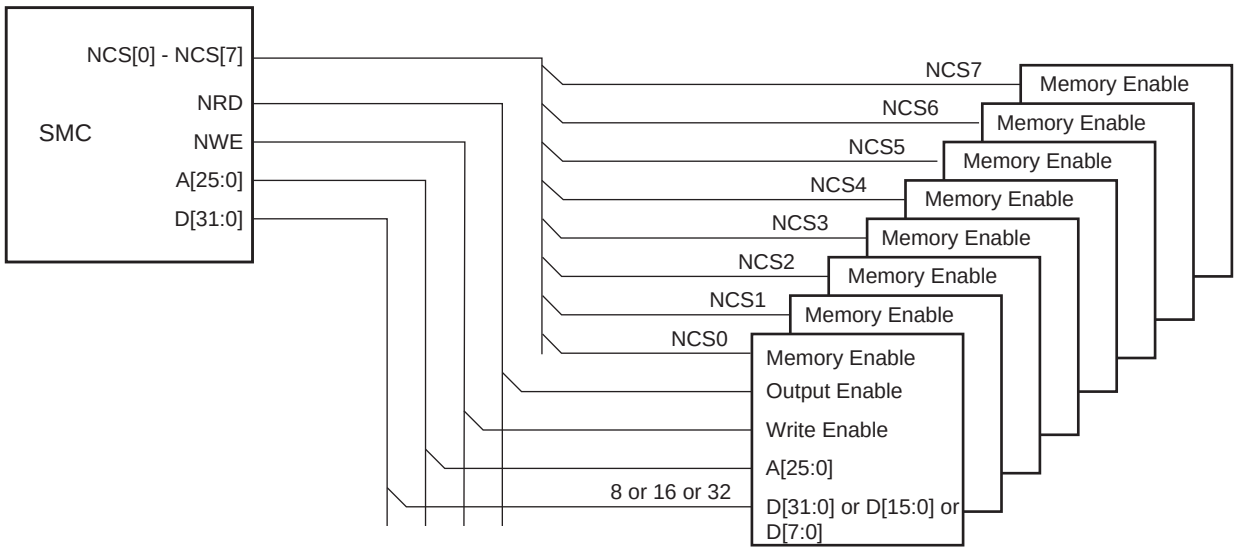
29.7 External Memory Mapping

The SMC provides up to 26 address lines, A[25:0]. This allows each chip select line to address up to 64 Mbytes of memory.

If the physical memory device connected on one chip select is smaller than 64 Mbytes, it wraps around and appears to be repeated within this space. The SMC correctly handles any valid access to the memory device within the page (see Figure 29-2).

A[25:0] is only significant for 8-bit memory, A[25:1] is used for 16-bit memory, A[25:2] is used for 32-bit memory.

Figure 29-2: Memory Connections for Eight External Devices



29.8 Connection to External Devices

29.8.1 Data Bus Width

A data bus width of 8, 16, or 32 bits can be selected for each chip select. This option is controlled by the field DBW in SMC_MODE (Mode Register) for the corresponding chip select.

Figure 29-3 shows how to connect a 512K x 8-bit memory on NCS2. Figure 29-4 shows how to connect a 512K x 16-bit memory on NCS2. Figure 29-5 shows two 16-bit memories connected as a single 32-bit memory

29.8.2 Byte Write or Byte Select Access

Each chip select with a 16-bit or 32-bit data bus can operate with one of two different types of write access: byte write or byte select access. This is controlled by the BAT field of the SMC_MODE register for the corresponding chip select.

Figure 29-3: Memory Connection for an 8-bit Data Bus

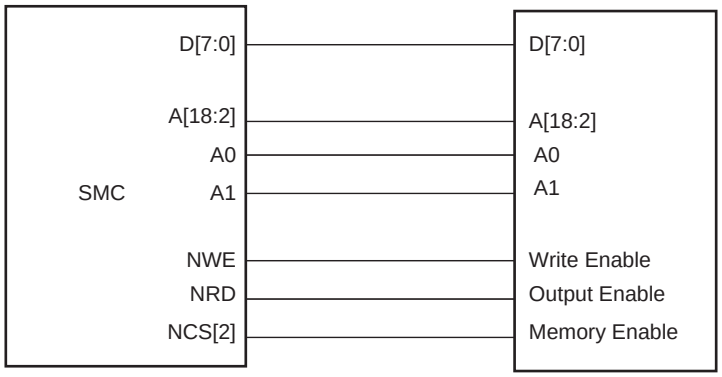
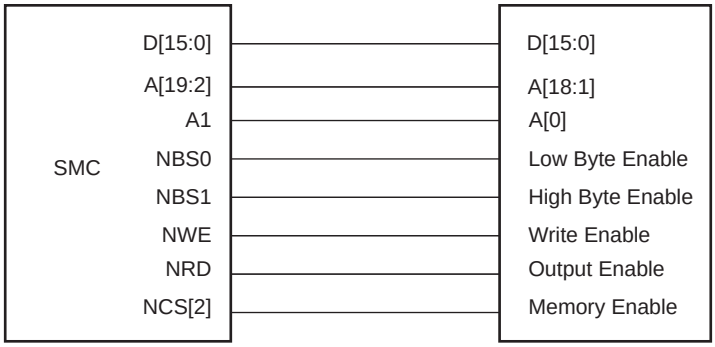
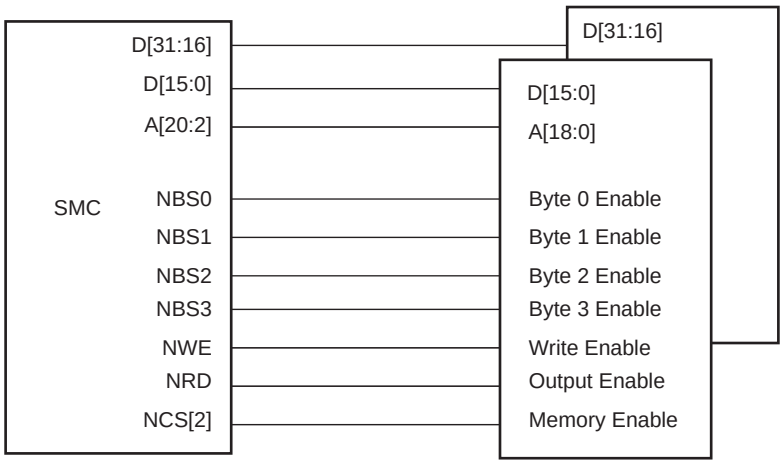


Figure 29-4: Memory Connection for a 16-bit Data Bus



© 2017 Microchip Technology Inc.

Figure 29-5: Memory Connection for a 32-bit Data Bus



29.8.2.1 Byte Write Access

Byte write access supports one byte write signal per byte of the data bus and a single read signal.

Note that the SMC does not allow boot in Byte Write Access mode.

- For 16-bit devices: the SMC provides NWR0 and NWR1 write signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus. One single read signal (NRD) is provided.

  Byte Write Access is used to connect 2 x 8-bit devices as a 16-bit memory.

- For 32-bit devices: NWR0, NWR1, NWR2 and NWR3, are the write signals of byte0 (lower byte), byte1, byte2 and byte 3 (upper byte) respectively. One single read signal (NRD) is provided.

  Byte Write Access is used to connect 4 x 8-bit devices as a 32-bit memory.

  Byte Write option is illustrated on Figure 29-6.

29.8.2.2 Byte Select Access

In this mode, read/write operations can be enabled/disabled at a byte level. One byte-select line per byte of the data bus is provided. One NRD and one NWE signal control read and write.

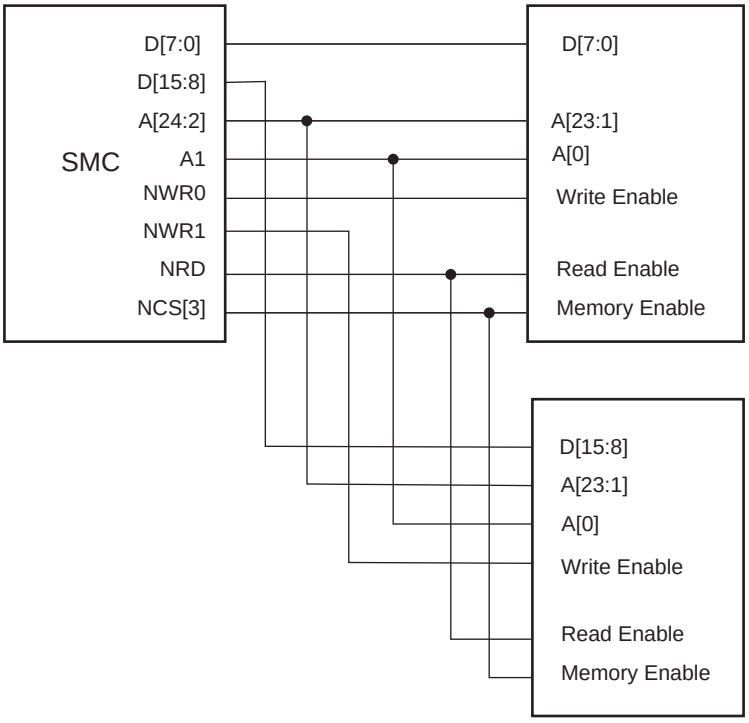
- For 16-bit devices: the SMC provides NBS0 and NBS1 selection signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus.

  Byte Select Access is used to connect one 16-bit device.

- For 32-bit devices: NBS0, NBS1, NBS2 and NBS3, are the selection signals of byte0 (lower byte), byte1, byte2 and byte 3 (upper byte) respectively. Byte Select Access is used to connect two 16-bit devices.

Figure 29-7 shows how to connect two 16-bit devices on a 32-bit data bus in Byte Select Access mode, on NCS3 (BAT = Byte Select Access).

Figure 29-6: Connection of 2 x 8-bit Devices on a 16-bit Bus: Byte Write Option



29.8.2.3 Signal Multiplexing

Depending on the byte access type (BAT), only the write signals or the byte select signals are used. To save IOs at the external bus interface, control signals at the SMC interface are multiplexed. Table 29-4 shows signal multiplexing depending on the data bus width and the byte access type.

For 32-bit devices, bits A0 and A1 are unused. For 16-bit devices, bit A0 of address is unused. When the Byte Select option is selected, NWR1 to NWR3 are unused. When teh Byte Write option is selected, NBS0 to NBS3 are unused.

© 2017 Microchip Technology Inc.

Figure 29-7: Connection of 2x16-bit Data Bus on a 32-bit Data Bus (Byte Select Option)

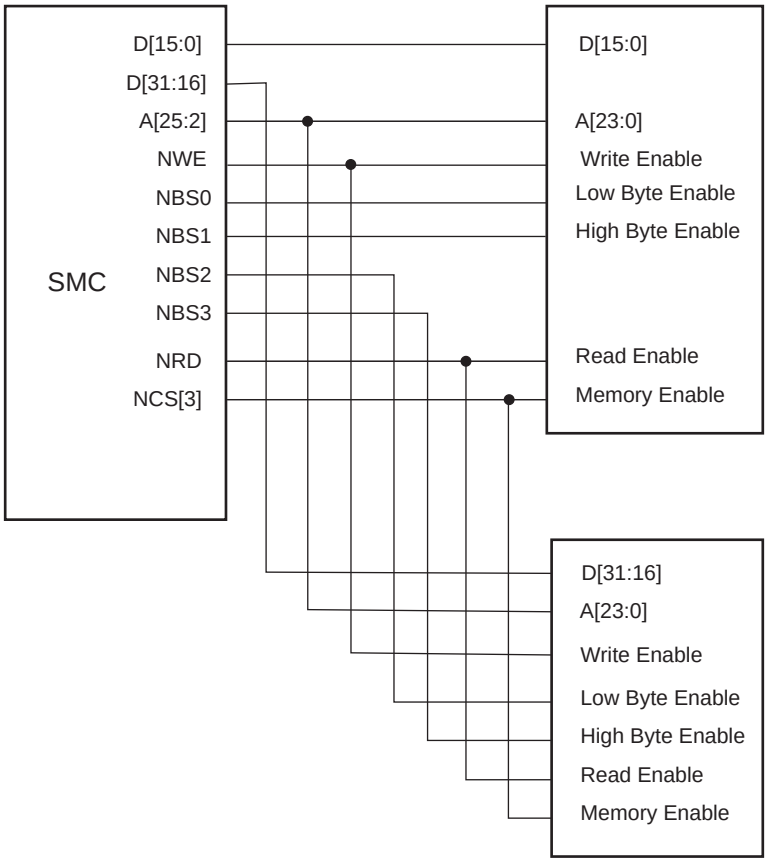


Table 29-4: SMC Multiplexed Signal Translation

Signal Name	32-bit Bus			16-bit Bus		8-bit Bus
Device Type	**1 x 32-bit**	**2 x 16-bit**	**4 x 8-bit**	**1 x 16-bit**	**2 x 8-bit**	**1 x 8-bit**
Byte Access Type (BAT)	Byte Select	Byte Select	Byte Write	Byte Select	Byte Write	–
NBS0_A0	NBS0	NBS0	–	NBS0	–	A0
NWE_NWR0	NWE	NWE	NWR0	NWE	NWR0	NWE
NBS1_NWR1	NBS1	NBS1	NWR1	NBS1	NWR1	–
NBS2_NWR2_A1	NBS2	NBS2	NWR2	A1	A1	A1
NBS3_NWR3	NBS3	NBS3	NWR3	–	–	–

29.9 Standard Read and Write Protocols

In the following sections, the byte access type is not considered. Byte select lines (NBS0 to NBS3) always have the same timing as the A address bus. NWE represents either the NWE signal in byte select access type or one of the byte write lines (NWR0 to NWR3) in byte write access type. NWR0 to NWR3 have the same timings and protocol as NWE. In the same way, NCS represents one of the NCS[0..5] chip select lines.

29.9.1 Read Waveforms

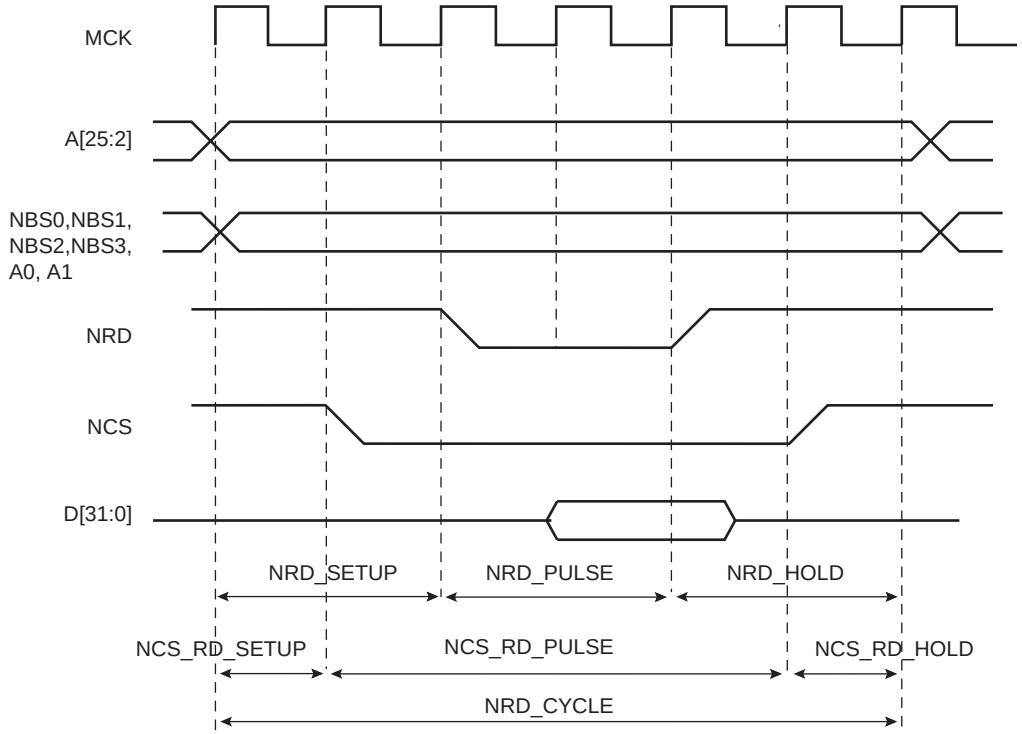
The read cycle is shown on Figure 29-8. The read cycle starts with the address setting on the memory address bus:

{A[25:2], A1, A0} for 8-bit devices

{A[25:2], A1} for 16-bit devices

A[25:2] for 32-bit devices

Figure 29-8: Standard Read Cycle



29.9.1.1 NRD Waveform

The NRD signal is characterized by a setup timing, a pulse width and a hold timing:

- NRD_SETUP—NRD setup time is defined as the setup of address before the NRD falling edge.
- NRD_PULSE—NRD pulse length is the time between NRD falling edge and NRD rising edge.
- NRD_HOLD—NRD hold time is defined as the hold time of address after the NRD rising edge.

29.9.1.2 NCS Waveform

Similar to the NRD signal, the NCS signal can be divided into a setup time, pulse length and hold time:

- NCS_RD_SETUP—NCS setup time is defined as the setup time of address before the NCS falling edge.
- NCS_RD_PULSE—NCS pulse length is the time between NCS falling edge and NCS rising edge;
- NCS_RD_HOLD—NCS hold time is defined as the hold time of address after the NCS rising edge.

29.9.1.3 Read Cycle

The NRD_CYCLE time is defined as the total duration of the read cycle, that is, from the time where address is set on the address bus to the point where address may change. The total read cycle time is defined as:

NRD_CYCLE = *NRD_SETUP* + *NRD_PULSE* + *NRD_HOLD*,

as well as

NRD_CYCLE = NCS_RD_SETUP + NCS_RD_PULSE + NCS_RD_HOLD

All NRD and NCS timings are defined separately for each chip select as an integer number of Master Clock cycles. The NRD_CYCLE field is common to both the NRD and NCS signals, thus the timing period is of the same duration.

© 2017 Microchip Technology Inc.

NRD_CYCLE, NRD_SETUP, and NRD_PULSE implicitly define the NRD_HOLD value as:

NRD_HOLD = NRD_CYCLE - NRD SETUP - NRD PULSE

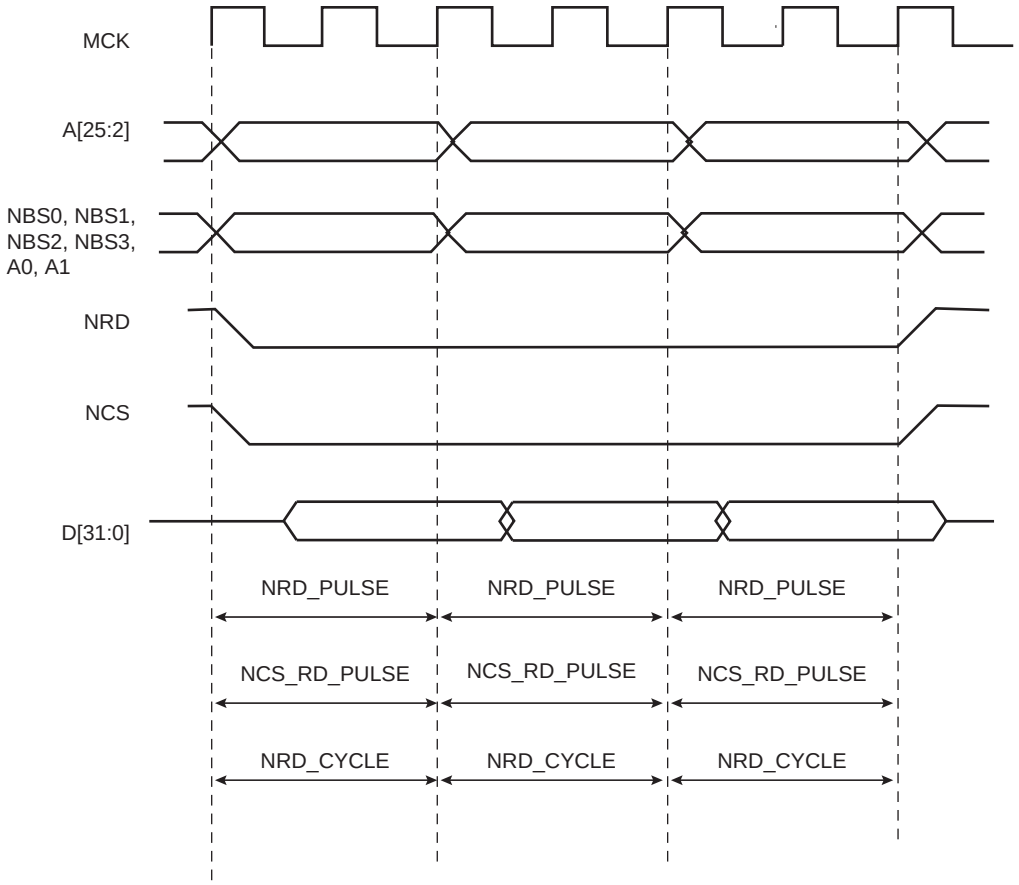
NRD_CYCLE, NCS_RD_SETUP, and NCS_RD_PULSE implicitly define the NCS_RD_HOLD value as:

NCS_RD_HOLD = NRD_CYCLE - NCS_RD_SETUP - NCS_RD_PULSE

29.9.1.4 Null Delay Setup and Hold

If null setup and hold parameters are programmed for NRD and/or NCS, NRD and NCS remain active continuously in case of consecutive read cycles in the same memory (see Figure 29-9).

Figure 29-9: No Setup, No Hold On NRD and NCS Read Signals



29.9.1.5 Null Pulse

Programming null pulse is not permitted. Pulse must be at least set to 1. A null value leads to unpredictable behavior.

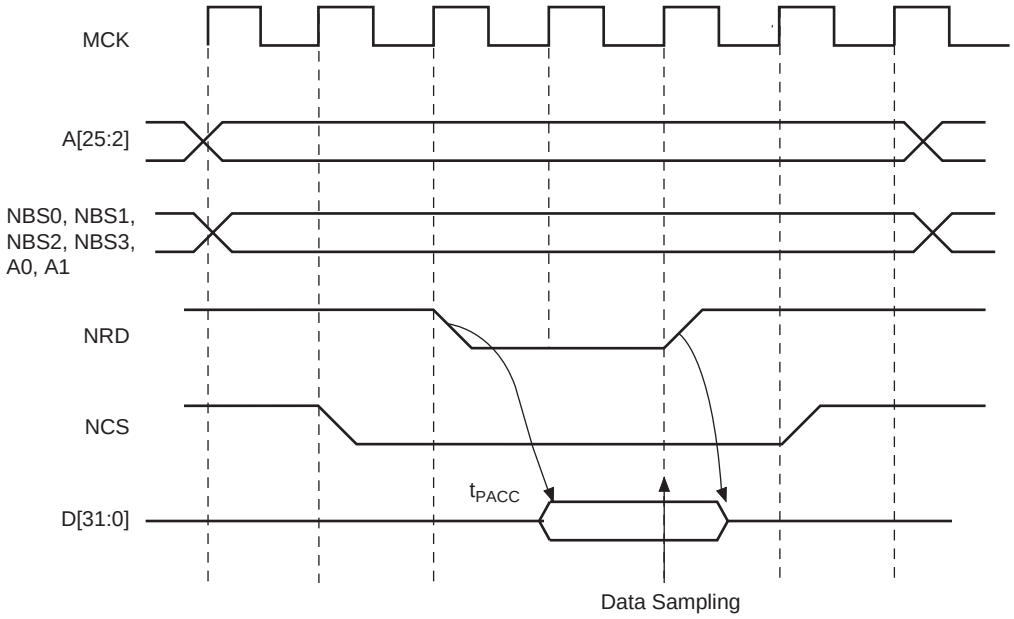
29.9.2 Read Mode

As NCS and NRD waveforms are defined independently of one other, the SMC needs to know when the read data is available on the data bus. The SMC does not compare NCS and NRD timings to know which signal rises first. The READ_MODE parameter in the SMC_MODE register of the corresponding chip select indicates which signal of NRD and NCS controls the read operation.

29.9.2.1 Read is Controlled by NRD (READ_MODE = 1)

Figure 29-10 shows the waveforms of a read operation of a typical asynchronous RAM. The read data is available t_{PACC} after the falling edge of NRD, and turns to ‘Z’ after the rising edge of NRD. In this case, the READ_MODE must be set to 1 (read is controlled by NRD), to indicate that data is available with the rising edge of NRD. The SMC samples the read data internally on the rising edge of Master Clock that generates the rising edge of NRD, whatever the programmed waveform of NCS may be.

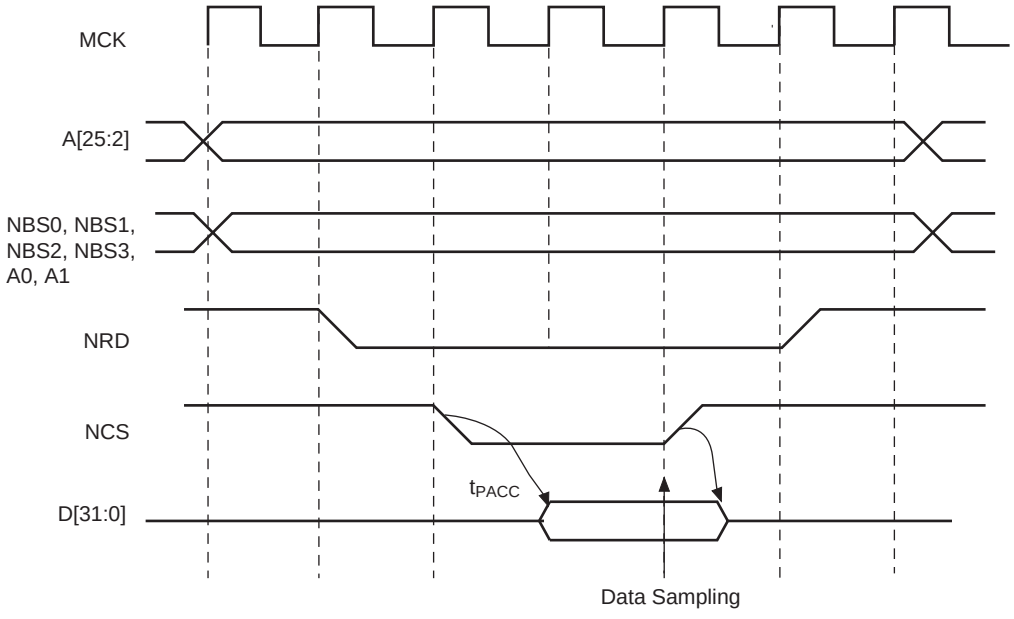
Figure 29-10: READ_MODE = 1 (Data sampled by SMC before rising edge of NRD)



29.9.2.2 Read is Controlled by NCS (READ_MODE = 0)

Figure 29-11 shows the typical read cycle of an LCD module. The read data is valid t_{PACC} after the falling edge of the NCS signal and remains valid until the rising edge of NCS. Data must be sampled when NCS is raised. In that case, the READ_MODE must be set to 0 (read is controlled by NCS): the SMC internally samples the data on the rising edge of Master Clock that generates the rising edge of NCS, whatever the programmed waveform of NRD may be.

Figure 29-11: READ_MODE = 0 (Data sampled by SMC before rising edge of NCS)



29.9.3 Write Waveforms

The write protocol (depicted in Figure 29-12) is similar to the read protocol. The write cycle starts with the address setting on the memory address bus.

© 2017 Microchip Technology Inc.

29.9.3.1 NWE Waveforms

The NWE signal is characterized by a setup timing, a pulse width and a hold timing.

- NWE_SETUP—NWE setup time is defined as the setup of address and data before the NWE falling edge.
- NWE_PULSE—NWE pulse length is the time between NWE falling edge and NWE rising edge.
- NWE_HOLD—NWE hold time is defined as the hold time of address and data after the NWE rising edge.

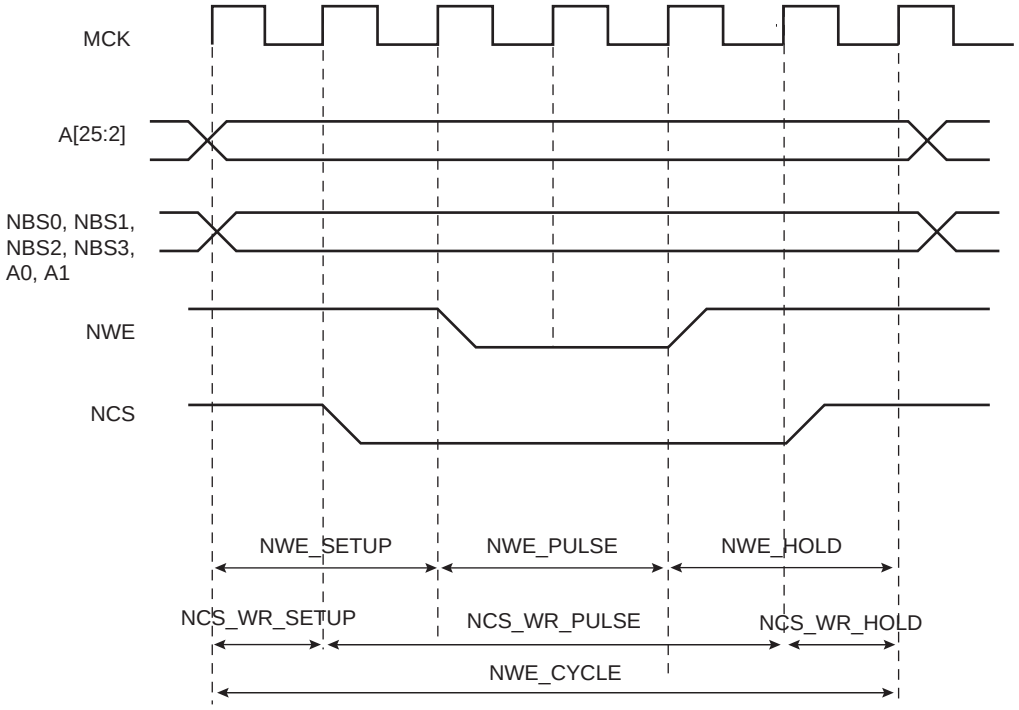
The NWE waveforms apply to all byte-write lines in Byte Write access mode: NWR0 to NWR3.

29.9.3.2 NCS Waveforms

The NCS signal waveforms in write operation are not the same that those applied in read operations, but are separately defined:

- NCS_WR_SETUP—NCS setup time is defined as the setup time of address before the NCS falling edge.
- NCS_WR_PULSE—NCS pulse length is the time between NCS falling edge and NCS rising edge.
- NCS_WR_HOLD—NCS hold time is defined as the hold time of address after the NCS rising edge.

Figure 29-12: Write Cycle



29.9.3.3 Write Cycle

The write_cycle time is defined as the total duration of the write cycle, that is, from the time where address is set on the address bus to the point where address may change. The total write cycle time is defined as:

NWE_CYCLE = *NWE_SETUP* + *NWE_PULSE* + *NWE_HOLD*,

as well as

NWE_CYCLE = NCS_WR_SETUP + NCS_WR_PULSE + NCS_WR_HOLD

All NWE and NCS (write) timings are defined separately for each chip select as an integer number of Master Clock cycles. The NWE_CYCLE field is common to both the NWE and NCS signals, thus the timing period is of the same duration.

NWE_CYCLE, NWE_SETUP, and NWE_PULSE implicitly define the NWE_HOLD value as:

NWE_HOLD = NWE_CYCLE - NWE_SETUP - NWE_PULSE

NWE_CYCLE, NCS_WR_SETUP, and NCS_WR_PULSE implicitly define the NCS_WR_HOLD value as:

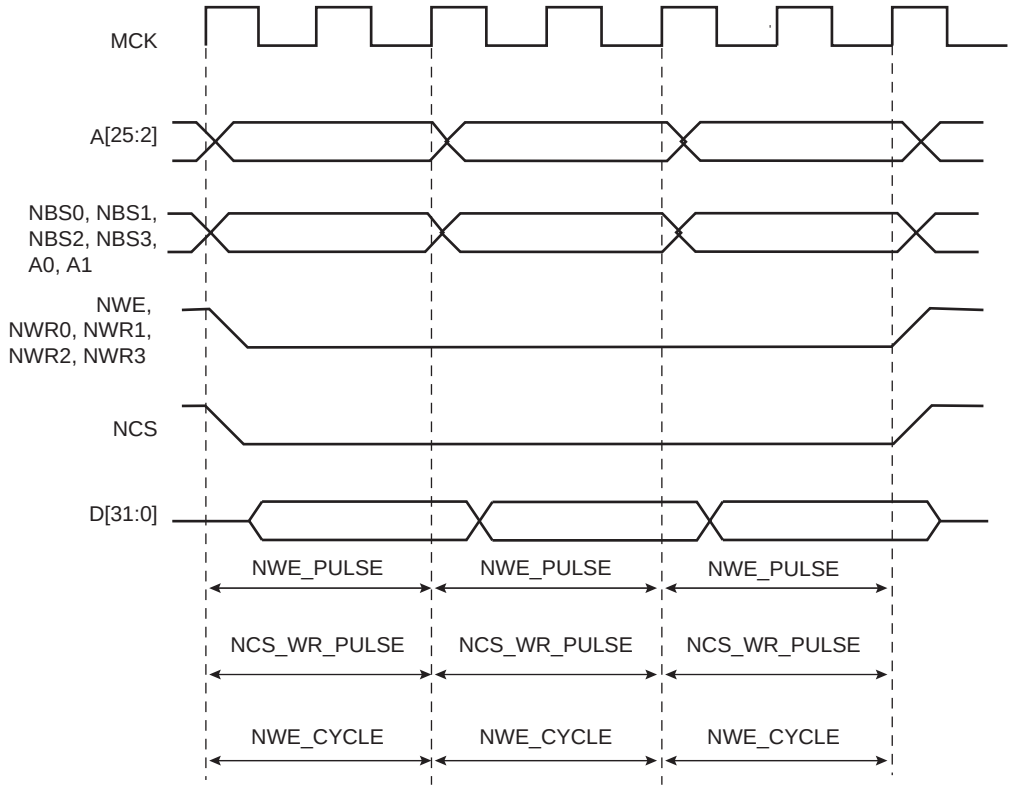
NCS_WR_HOLD = NWE_CYCLE - NCS_WR_SETUP - NCS_WR_PULSE

© 2017 Microchip Technology Inc.

29.9.3.4 Null Delay Setup and Hold

If null setup parameters are programmed for NWE and/or NCS, NWE and/or NCS remain active continuously in case of consecutive write cycles in the same memory (see Figure 29-13). However, for devices that perform write operations on the rising edge of NWE or NCS, such as SRAM, either a setup or a hold must be programmed.

Figure 29-13: Null Setup and Hold Values of NCS and NWE in Write Cycle



29.9.3.5 Null Pulse

Programming null pulse is not permitted. Pulse must be at least set to 1. A null value leads to unpredictable behavior.

29.9.4 Write Mode

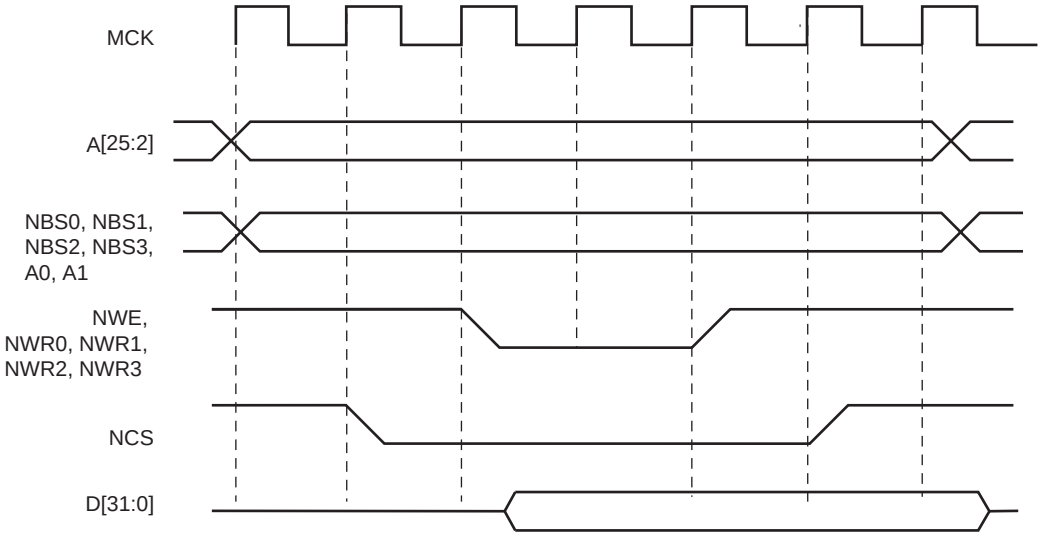
The WRITE_MODE parameter in the SMC_MODE register of the corresponding chip select indicates which signal controls the write operation.

29.9.4.1 Write is Controlled by NWE (WRITE_MODE = 1):

Figure 29-14 shows the waveforms of a write operation with WRITE_MODE set to 1. The data is put on the bus during the pulse and hold steps of the NWE signal. The internal data buffers are switched to output mode after the NWE_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NCS.

© 2017 Microchip Technology Inc.

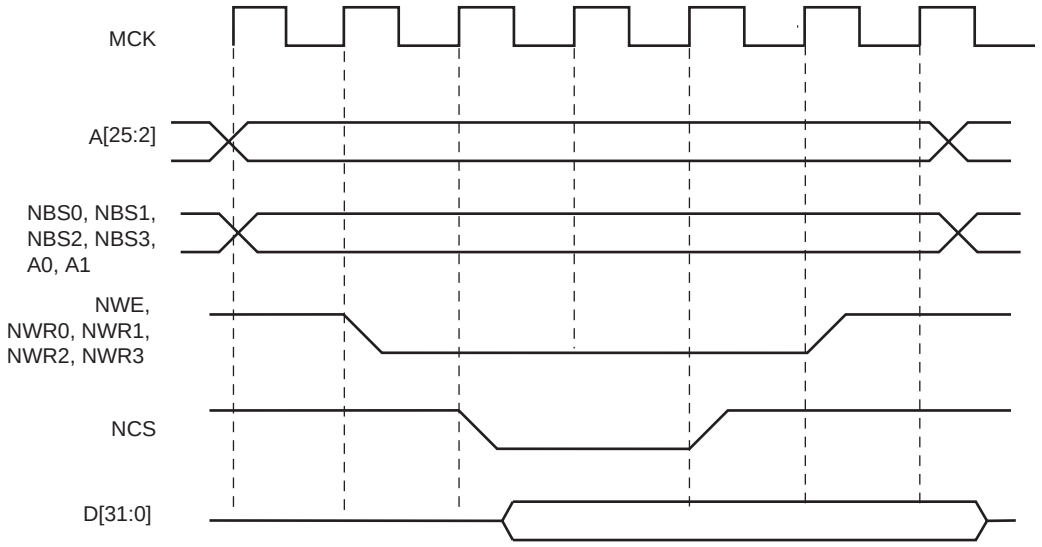
Figure 29-14: WRITE_MODE = 1 (Write Operation Controlled by NWE)



29.9.4.2 Write is Controlled by NCS (WRITE_MODE = 0)

Figure 29-15 shows the waveforms of a write operation with WRITE_MODE set to 0. The data is put on the bus during the pulse and hold steps of the NCS signal. The internal data buffers are switched to output mode after the NCS_WR_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NWE.

Figure 29-15: WRITE_MODE = 0 (Write Operation Controlled by NCS)



29.9.5 Coding Timing Parameters

All timing parameters are defined for one chip select and are grouped together in one SMC_REGISTER according to their type.

The SMC_SETUP register groups the definition of all setup parameters:

- NRD_SETUP, NCS_RD_SETUP, NWE_SETUP, NCS_WR_SETUP

The SMC_PULSE register groups the definition of all pulse parameters:

- NRD_PULSE, NCS_RD_PULSE, NWE_PULSE, NCS_WR_PULSE

The SMC_CYCLE register groups the definition of all cycle parameters:

- NRD_CYCLE, NWE_CYCLE

Table 29-5 shows how the timing parameters are coded and their permitted range.

Table 29-5: Coding and Range of Timing Parameters

Coded Value	Number of Bits	Effective Value	Permitted Range	
			Coded Value	Effective Value
setup [5:0]	6	128 x setup[5] + setup[4:0]	0 ≤ 31	0 ≤ 128+31
pulse [6:0]	7	256 x pulse[6] + pulse[5:0]	0 ≤ 63	0 ≤ 256+63
cycle [8:0]	9	256 x cycle[8:7] + cycle[6:0]	0 ≤ 127	0 ≤ 256+127 0 ≤ 512+127 0 ≤ 768+127

29.9.6 Reset Values of Timing Parameters

Table 29-9 gives the default value of timing parameters at reset.

29.9.7 Usage Restriction

The SMC does not check the validity of the user-programmed parameters. If the sum of SETUP and PULSE parameters is larger than the corresponding CYCLE parameter, this leads to unpredictable behavior of the SMC.

- For read operations:

  Null but positive setup and hold of address and NRD and/or NCS can not be guaranteed at the memory interface because of the propagation delay of theses signals through external logic and pads. If positive setup and hold values must be verified, then it is strictly recommended to program non-null values so as to cover possible skews between address, NCS and NRD signals.

- For write operations:

  If a null hold value is programmed on NWE, the SMC can guarantee a positive hold of address, byte select lines, and NCS signal after the rising edge of NWE. This is true for WRITE_MODE = 1 only. See “Early Read Wait State” .

- For read and write operations:

  A null value for pulse parameters is forbidden and may lead to unpredictable behavior.

  In read and write cycles, the setup and hold time parameters are defined in reference to the address bus. For external devices that require setup and hold time between NCS and NRD signals (read), or between NCS and NWE signals (write), these setup and hold times must be converted into setup and hold times in reference to the address bus.

29.10 Automatic Wait States

Under certain circumstances, the SMC automatically inserts idle cycles between accesses to avoid bus contention or operation conflict.

29.10.1 Chip Select Wait States

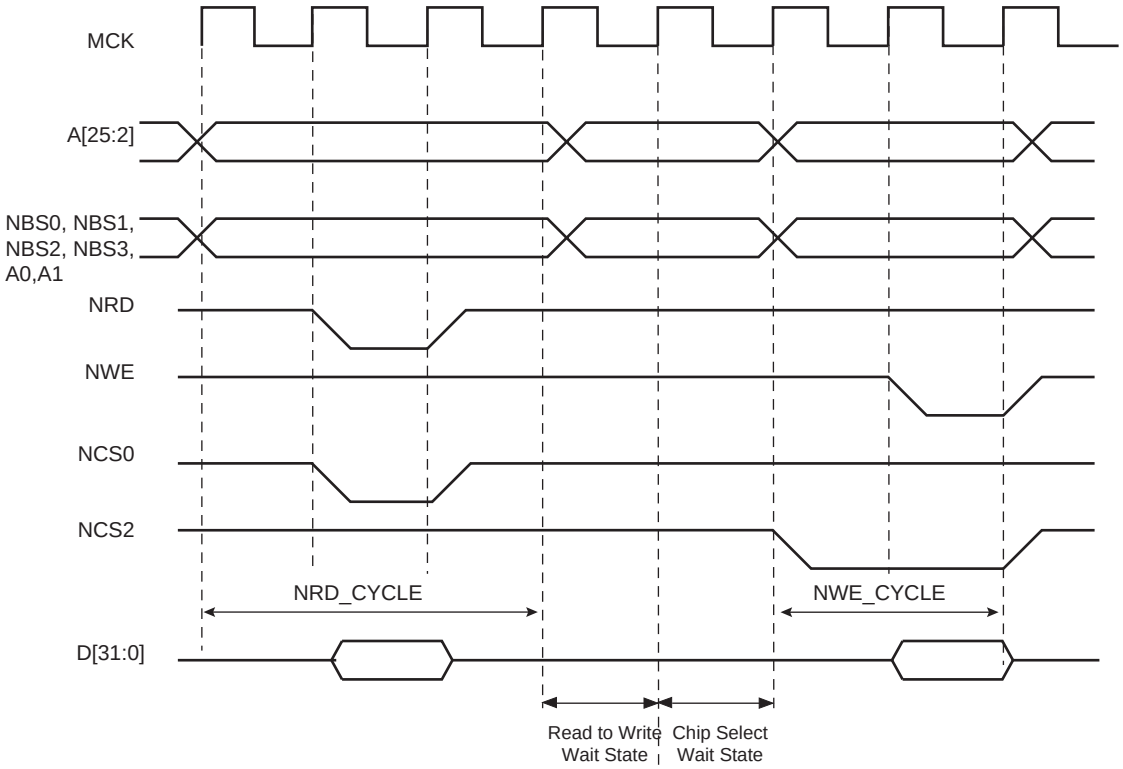
The SMC always inserts an idle cycle between 2 transfers on separate chip selects. This idle cycle ensures that there is no bus contention between the de-activation of one device and the activation of the next one.

During chip select wait state, all control lines are turned inactive: NBS0 to NBS3, NWR0 to NWR3, NCS[0..5], NRD lines are all set to 1.

Figure 29-16 illustrates a chip select wait state between access on Chip Select 0 and Chip Select 2.

© 2017 Microchip Technology Inc.

Figure 29-16: Chip Select Wait State between a Read Access on NCS0 and a Write Access on NCS2



29.10.2 Early Read Wait State

In some cases, the SMC inserts a wait state cycle between a write access and a read access to allow time for the write cycle to end before the subsequent read cycle begins. This wait state is not generated in addition to a chip select wait state. The early read cycle thus only occurs between a write and read access to the same memory device (same chip select).

An early read wait state is automatically inserted if at least one of the following conditions is valid:

- the write controlling signal has no hold time and the read controlling signal has no setup time (Figure 29-17).
- in NCS write controlled mode (WRITE_MODE = 0), there is no hold timing on the NCS signal and the NCS_RD_SETUP parameter is set to 0, regardless of the read mode (Figure 29-18). The write operation must end with a NCS rising edge. Without an Early Read Wait State, the write operation could not complete properly.
- in NWE controlled mode (WRITE_MODE = 1) and if there is no hold timing (NWE_HOLD = 0), the feedback of the write control signal is used to control address, data, chip select and byte select lines. If the external write control signal is not inactivated as expected due to load capacitances, an Early Read Wait State is inserted and address, data and control signals are maintained one more cycle. See Figure 29-19.

Figure 29-17: Early Read Wait State: Write with No Hold Followed by Read with No Setup

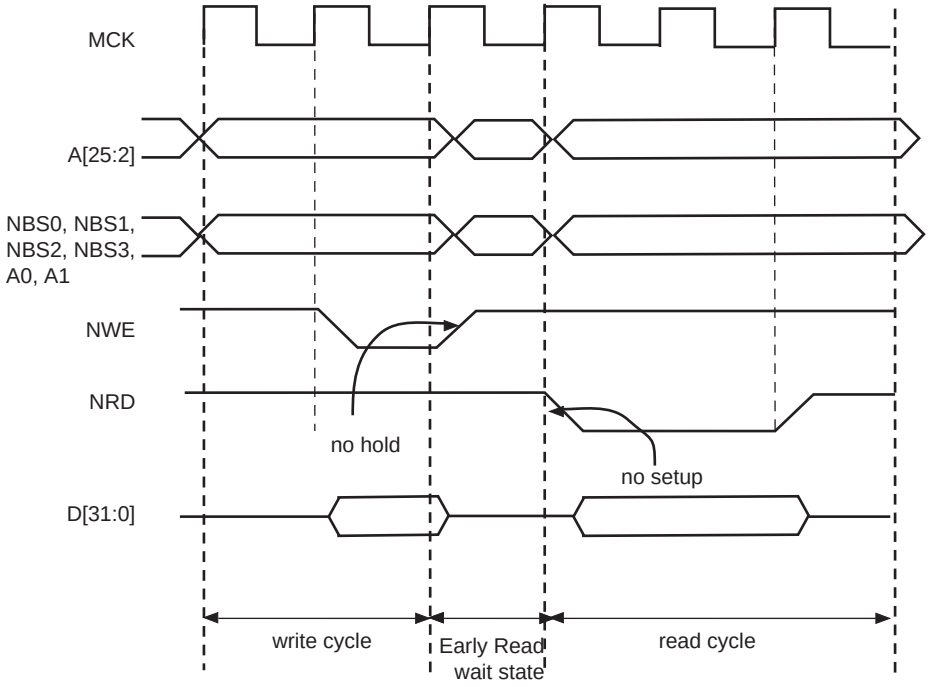
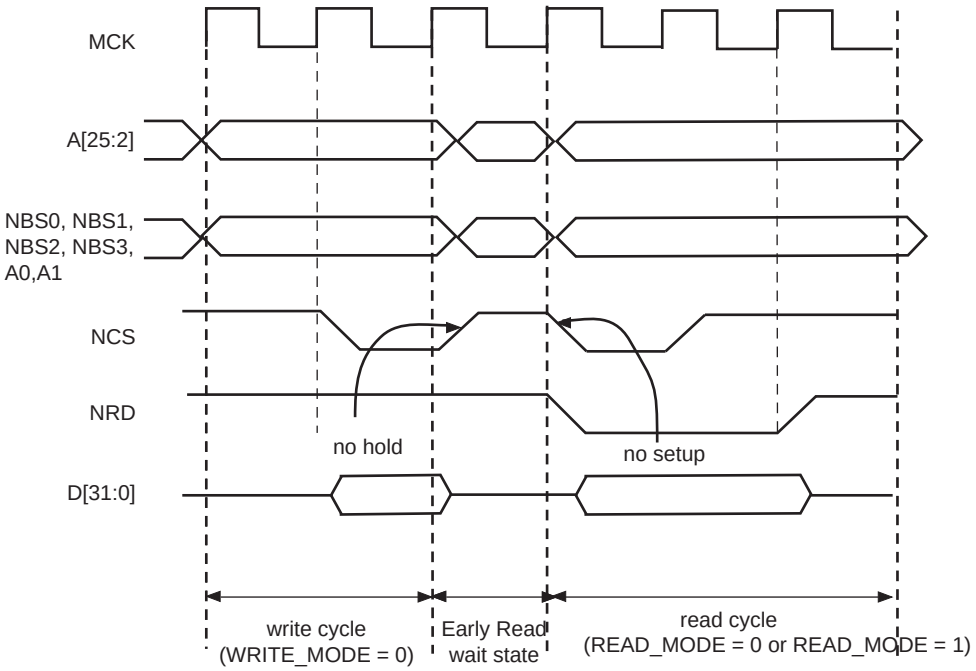
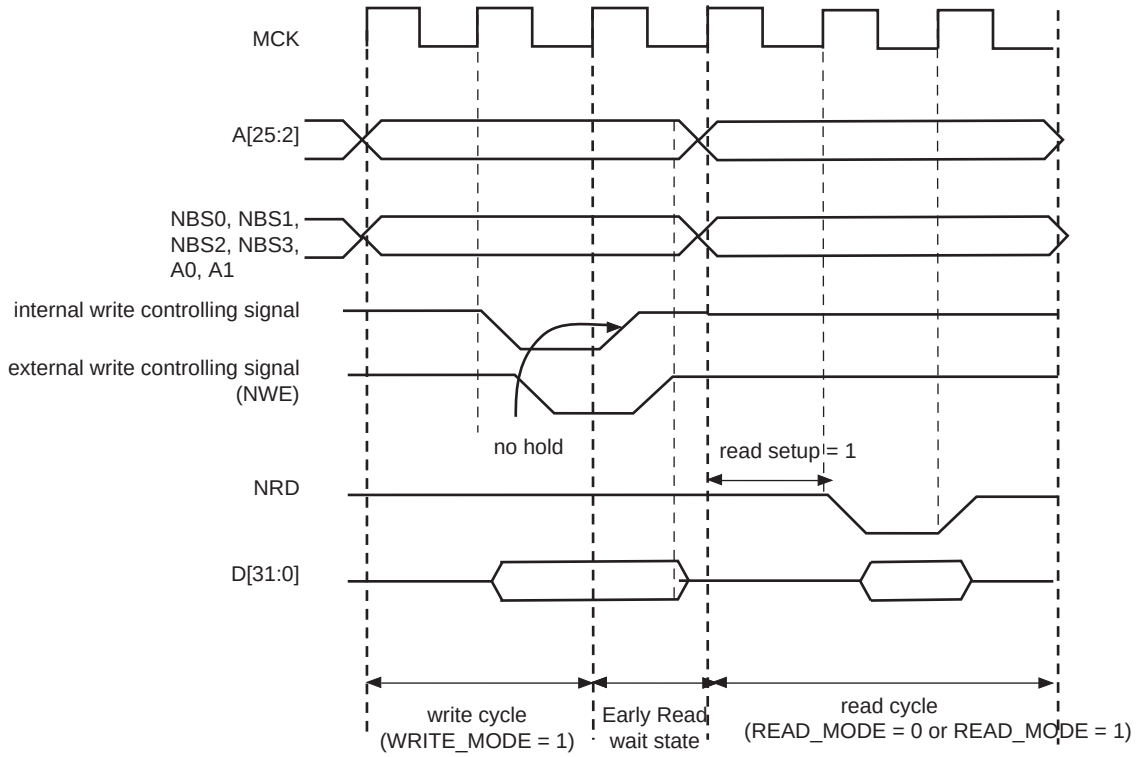


Figure 29-18: Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read with No NCS Setup



© 2017 Microchip Technology Inc.

Figure 29-19: Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read with one Set-up Cycle



29.10.3 Reload User Configuration Wait State

The user may change any of the configuration parameters by writing the SMC user interface.

When detecting that a new user configuration has been written in the user interface, the SMC inserts a wait state before starting the next access. The so called “Reload User Configuration Wait State” is used by the SMC to load the new set of parameters to apply to next accesses.

The Reload Configuration Wait State is not applied in addition to the Chip Select Wait State. If accesses before and after re-programming the user interface are made to different devices (Chip Selects), then one single Chip Select Wait State is applied.

On the other hand, if accesses before and after writing the user interface are made to the same device, a Reload Configuration Wait State is inserted, even if the change does not concern the current Chip Select.

29.10.3.1 User Procedure

To insert a Reload Configuration Wait State, the SMC detects a write access to any SMC_MODE register of the user interface. If the user only modifies timing registers (SMC_SETUP, SMC_PULSE, SMC_CYCLE registers) in the user interface, he must validate the modification by writing the SMC_MODE, even if no change was made on the mode parameters.

The user must not change the configuration parameters of an SMC Chip Select (Setup, Pulse, Cycle, Mode) if accesses are performed on this CS during the modification. Any change of the Chip Select parameters, while fetching the code from a memory connected on this CS, may lead to unpredictable behavior. The instructions used to modify the parameters of an SMC Chip Select can be executed from the internal RAM or from a memory connected to another CS.

29.10.3.2 Slow Clock Mode Transition

A Reload Configuration Wait State is also inserted when the Slow Clock Mode is entered or exited, after the end of the current transfer (see Section 29.13 ”Slow Clock Mode”).

29.10.4 Read to Write Wait State

Due to an internal mechanism, a wait cycle is always inserted between consecutive read and write SMC accesses.

This wait cycle is referred to as a read to write wait state in this document.

This wait cycle is applied in addition to chip select and reload user configuration wait states when they are to be inserted. See Figure 29-16.

29.11 Data Float Wait States

Some memory devices are slow to release the external bus. For such devices, it is necessary to add wait states (data float wait states) after a read access:

- before starting a read access to a different external memory
- before starting a write access to the same device or to a different external one.

The Data Float Output Time (t_{DF}) for each external memory device is programmed in the TDF_CYCLES field of the SMC_MODE register for the corresponding chip select. The value of TDF_CYCLES indicates the number of data float wait cycles (between 0 and 15) before the external device releases the bus, and represents the time allowed for the data output to go to high impedance after the memory is disabled.

Data float wait states do not delay internal memory accesses. Hence, a single access to an external memory with long t_{DF} will not slow down the execution of a program from internal memory.

The data float wait states management depends on the READ_MODE and the TDF_MODE fields of the SMC_MODE register for the corresponding chip select.

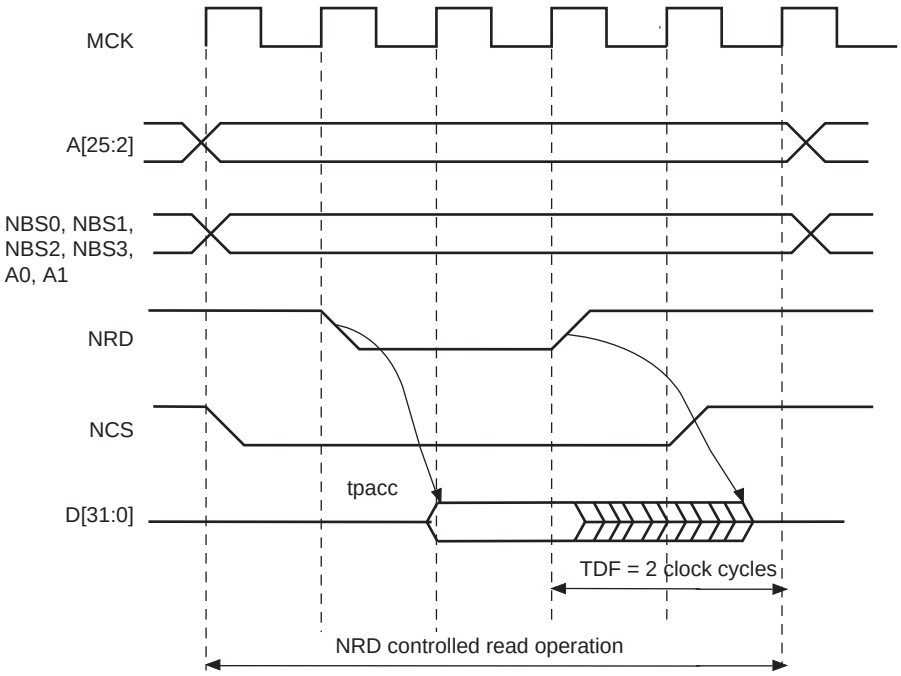
29.11.1 READ_MODE

Setting the READ_MODE to 1 indicates to the SMC that the NRD signal is responsible for turning off the tri-state buffers of the external memory device. The Data Float Period then begins after the rising edge of the NRD signal and lasts TDF_CYCLES MCK cycles.

When the read operation is controlled by the NCS signal (READ_MODE = 0), the TDF field gives the number of MCK cycles during which the data bus remains busy after the rising edge of NCS.

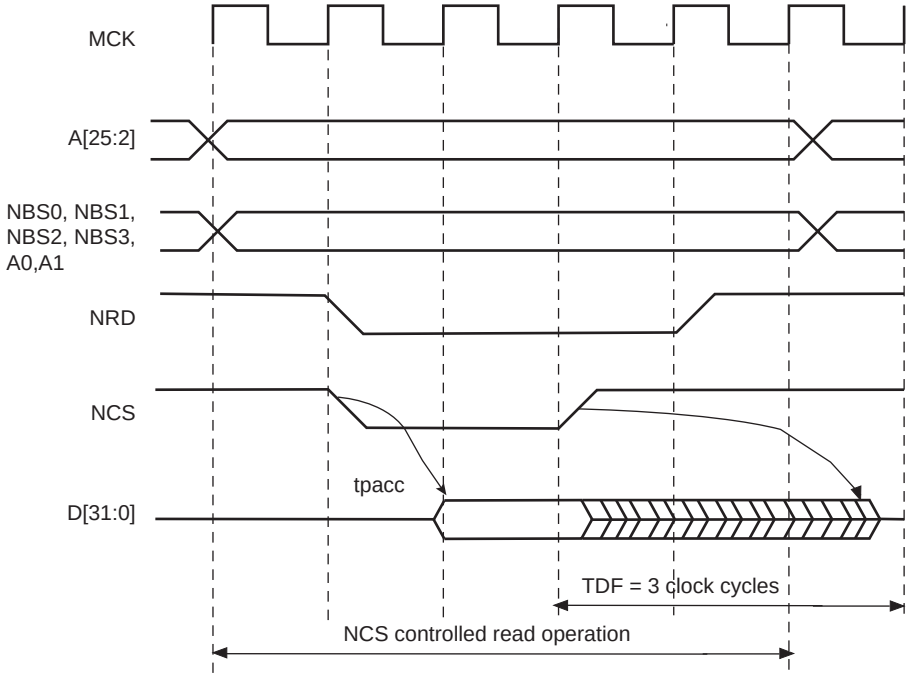
Figure 29-20 illustrates the Data Float Period in NRD-controlled mode (READ_MODE =1), assuming a data float period of 2 cycles (TDF_CYCLES = 2). Figure 29-21 shows the read operation when controlled by NCS (READ_MODE = 0) and the TDF_CYCLES parameter equals 3.

Figure 29-20: TDF Period in NRD Controlled Read Access (TDF = 2)



© 2017 Microchip Technology Inc.

Figure 29-21: TDF Period in NCS Controlled Read Operation (TDF = 3)



29.11.2 TDF Optimization Enabled (TDF_MODE = 1)

When the TDF_MODE of the SMC_MODE register is set to 1 (TDF optimization is enabled), the SMC takes advantage of the setup period of the next access to optimize the number of wait states cycle to insert.

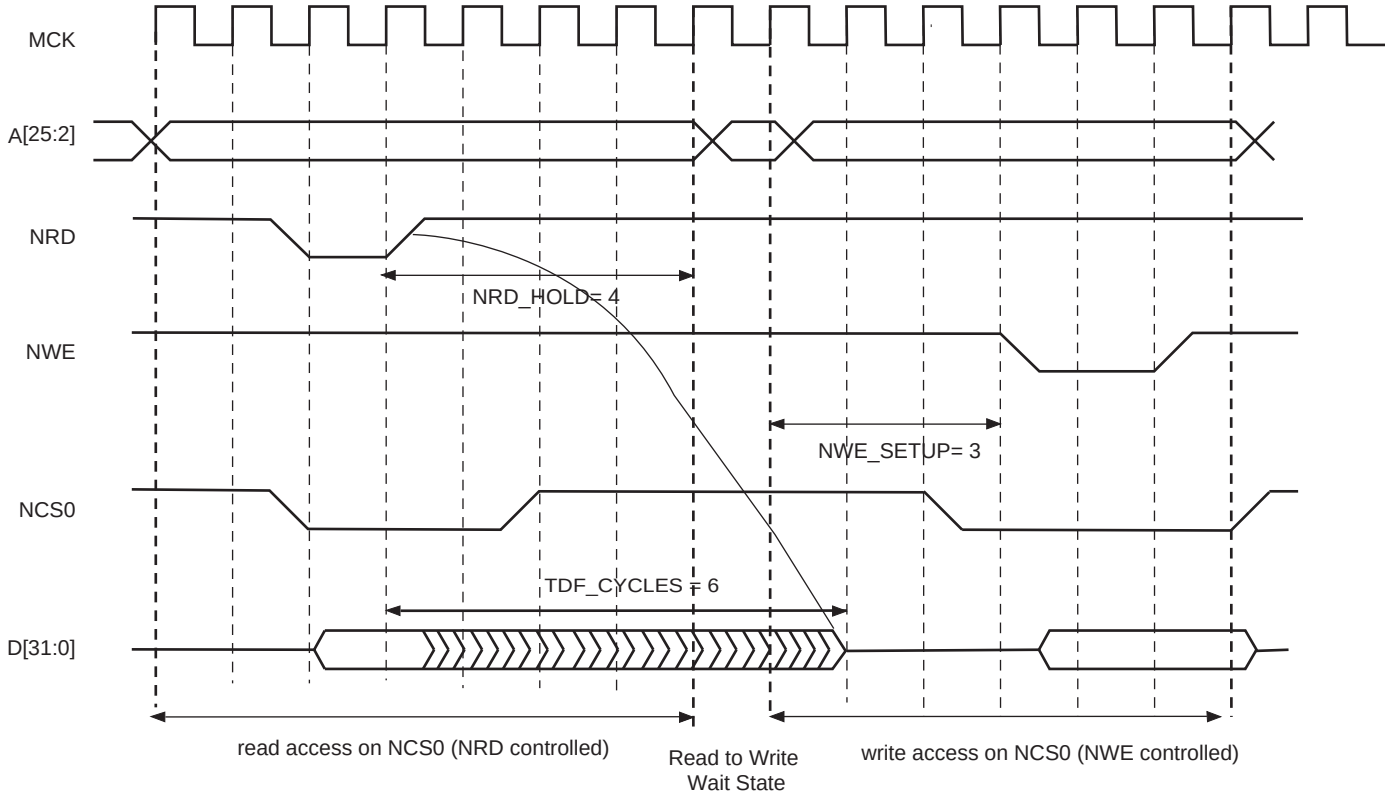
Figure 29-22 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

NRD_HOLD = 4; READ_MODE = 1 (NRD controlled)

NWE_SETUP = 3; WRITE_MODE = 1 (NWE controlled)

TDF_CYCLES = 6; TDF_MODE = 1 (optimization enabled).

Figure 29-22: TDF Optimization: No TDF wait states are inserted if the TDF period is over when the next access begins



29.11.3 TDF Optimization Disabled (TDF_MODE = 0)

When optimization is disabled, tdf wait states are inserted at the end of the read transfer, so that the data float period is ended when the second access begins. If the hold period of the read1 controlling signal overlaps the data float period, no additional tdf wait states will be inserted.

Figure 29-23, Figure 29-24 and Figure 29-25 illustrate the cases:

- read access followed by a read access on another chip select,
- read access followed by a write access on another chip select,
- read access followed by a write access on the same chip select,

with no TDF optimization.

© 2017 Microchip Technology Inc.

Figure 29-23: TDF Optimization Disabled (TDF Mode = 0). TDF wait states between 2 read accesses on different chip selects

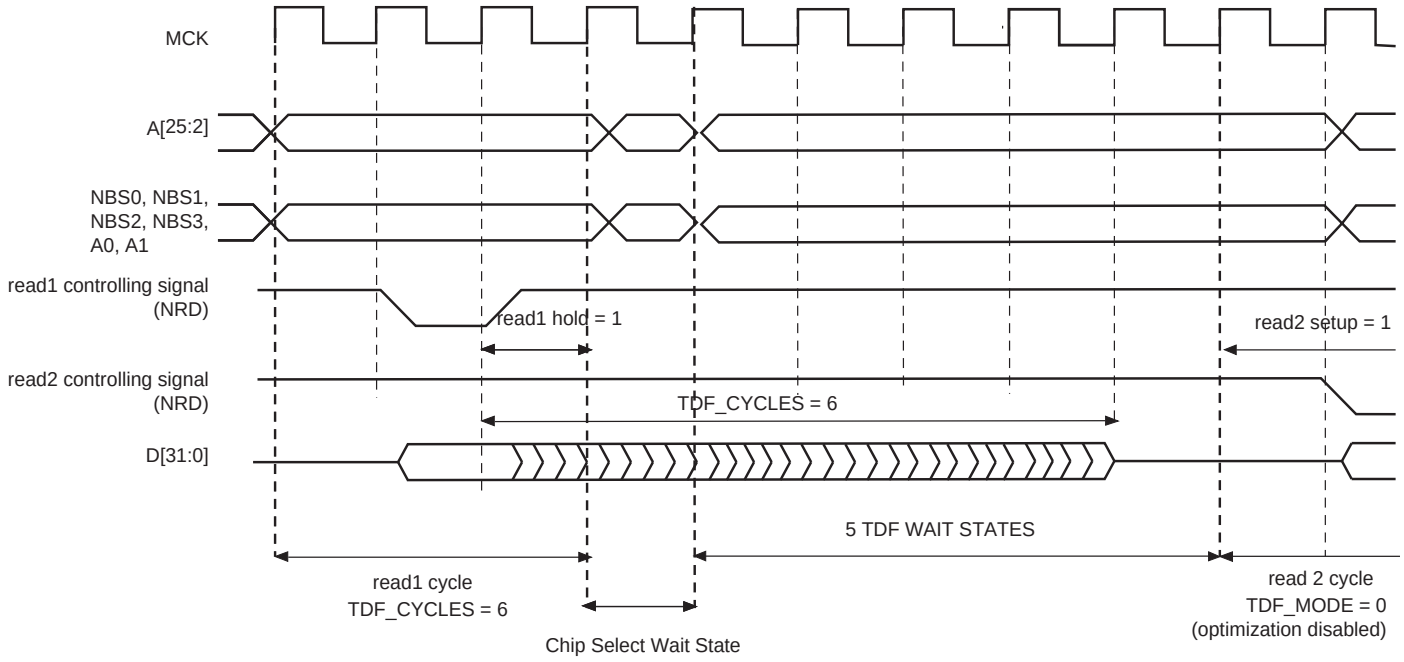


Figure 29-24: TDF Mode = 0: TDF wait states between a read and a write access on different chip selects

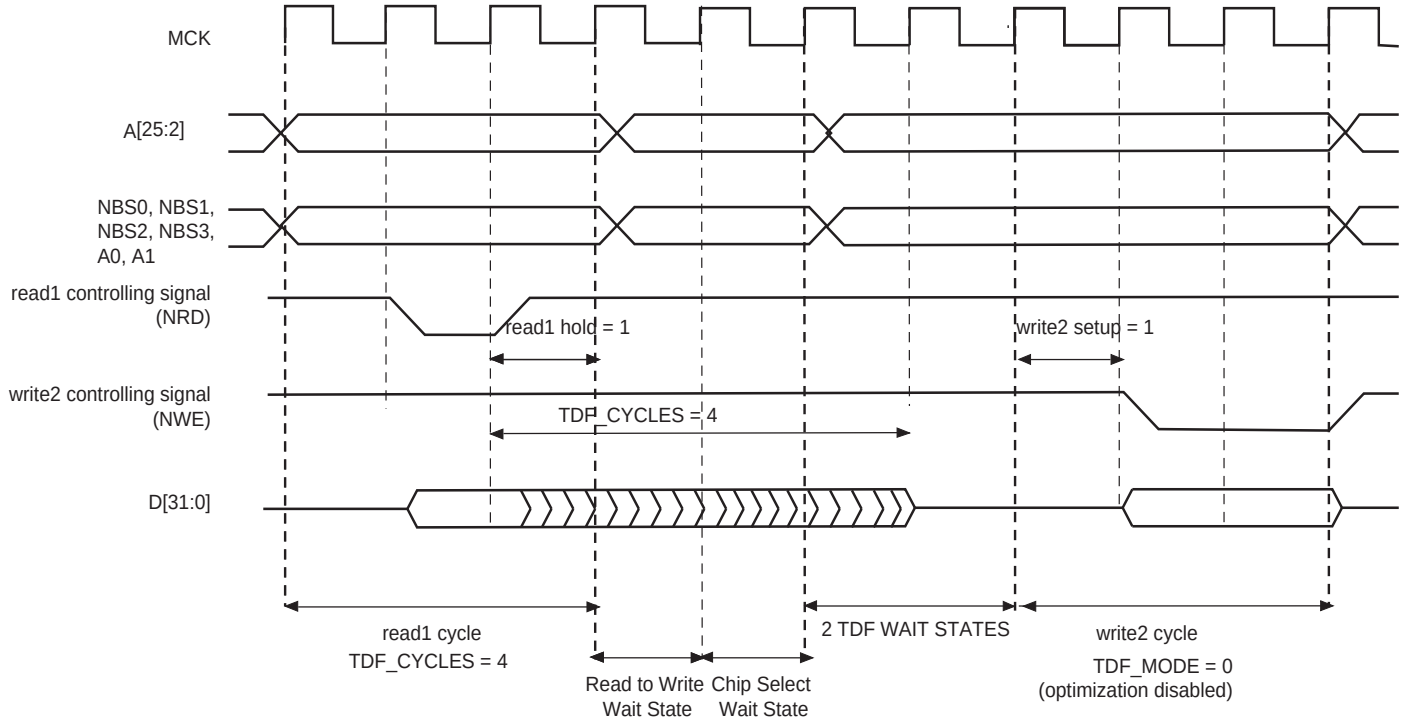
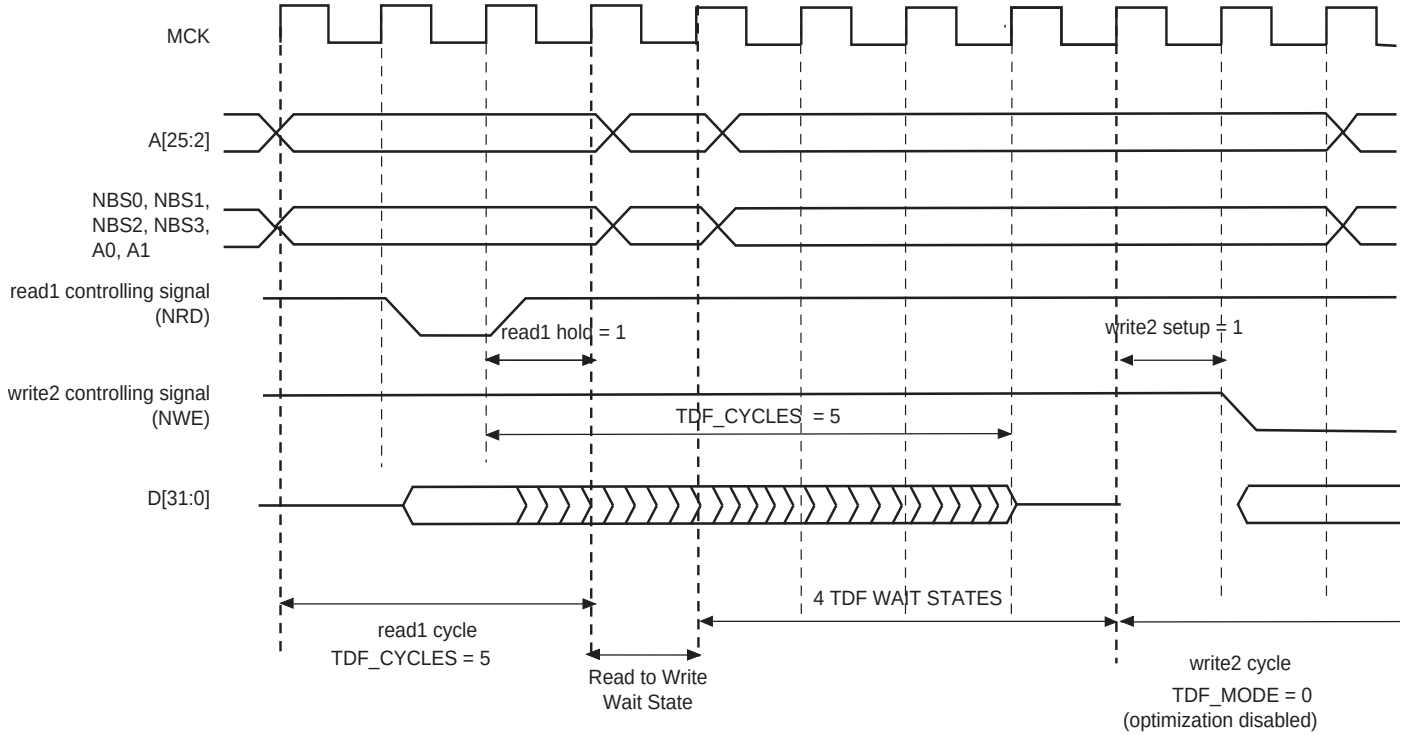


Figure 29-25: TDF Mode = 0: TDF wait states between read and write accesses on the same chip select



29.12 External Wait

Any access can be extended by an external device using the NWAIT input signal of the SMC. The EXNW_MODE field of the SMC_MODE register on the corresponding chip select must be set to either to “10” (frozen mode) or “11” (ready mode). When the EXNW_MODE is set to “00” (disabled), the NWAIT signal is simply ignored on the corresponding chip select. The NWAIT signal delays the read or write operation in regards to the read or write controlling signal, depending on the read and write modes of the corresponding chip select.

29.12.1 Restriction

When one of the EXNW_MODE is enabled, **it is mandatory to program at least one hold cycle for the read/write controlling signal. For that reason, the NWAIT signal cannot be used in Page Mode (“Asynchronous Page Mode”), or in Slow Clock Mode (“Slow Clock Mode”).**

The NWAIT signal is assumed to be a response of the external device to the read/write request of the SMC. Then NWAIT is examined by the SMC only in the pulse state of the read or write controlling signal. The assertion of the NWAIT signal outside the expected period has no impact on SMC behavior.

© 2017 Microchip Technology Inc.

29.12.2 Frozen Mode

When the external device asserts the NWAIT signal (active low), and after internal synchronization of this signal, the SMC state is frozen, i.e., SMC internal counters are frozen, and all control signals remain unchanged. When the resynchronized NWAIT signal is deasserted, the SMC completes the access, resuming the access from the point where it was stopped. See Figure 29-26. This mode must be selected when the external device uses the NWAIT signal to delay the access and to freeze the SMC.

The assertion of the NWAIT signal outside the expected period is ignored as illustrated in Figure 29-27.

Figure 29-26: Write Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)

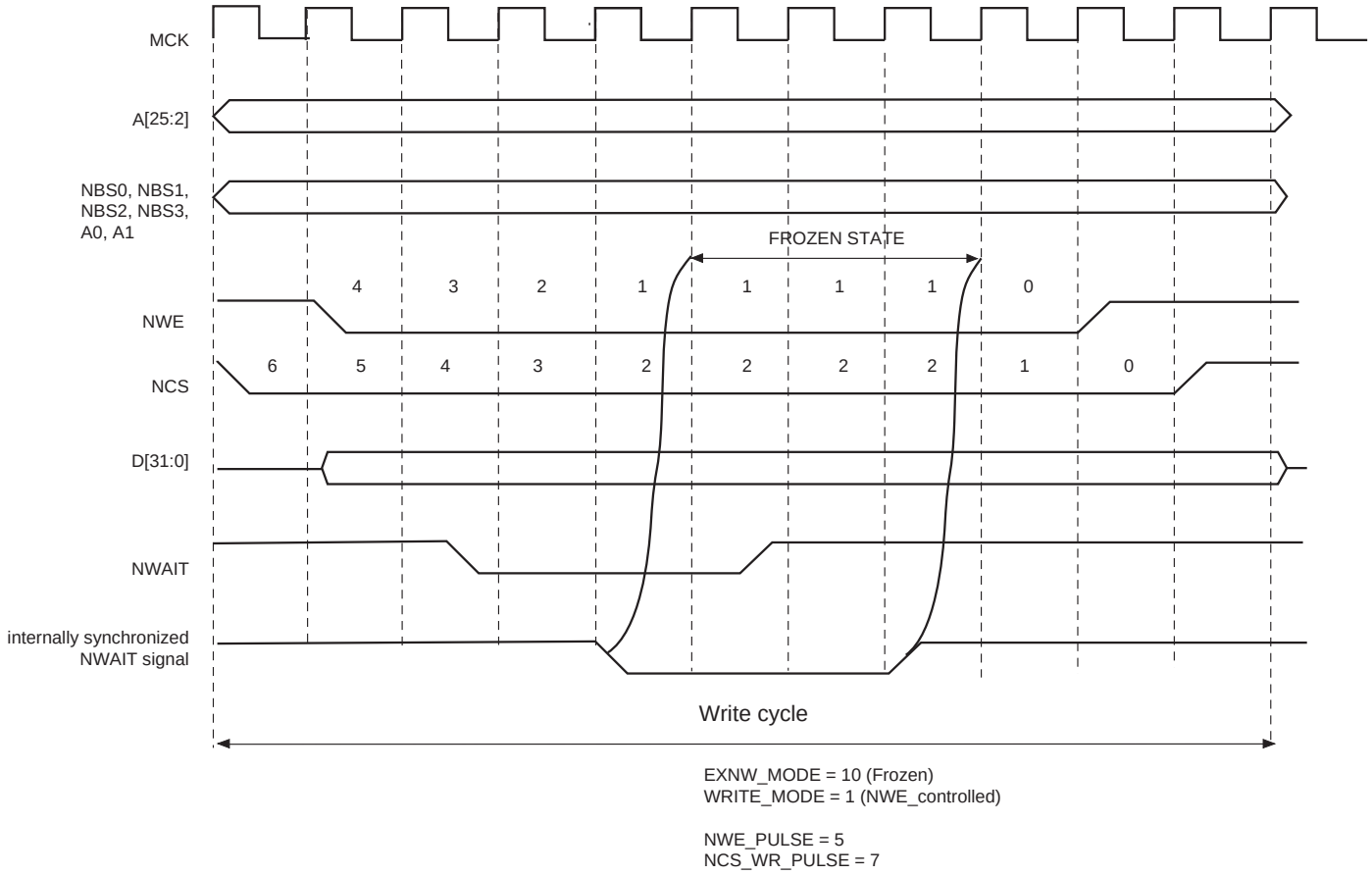
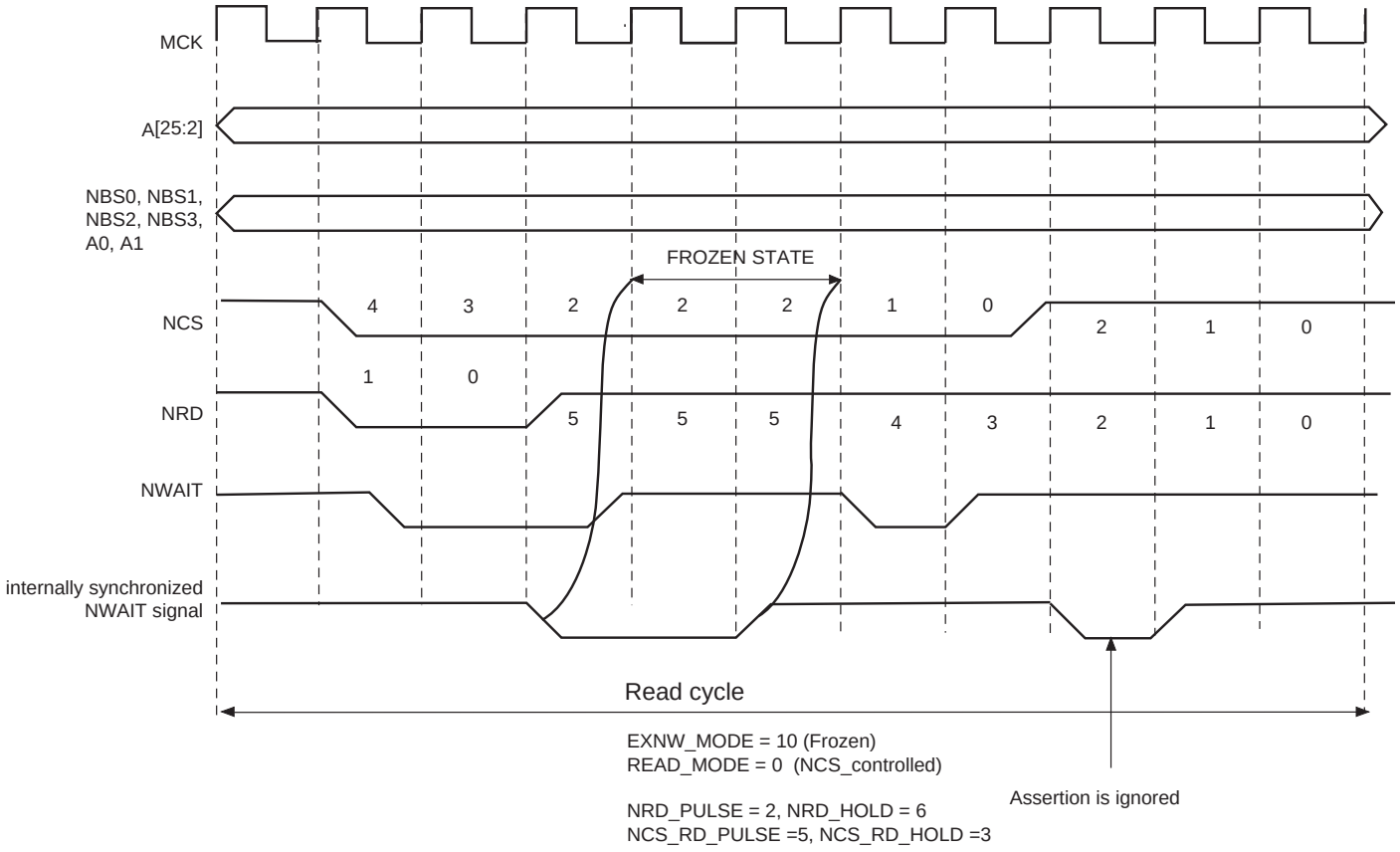


Figure 29-27: Read Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)



© 2017 Microchip Technology Inc.

29.12.3 Ready Mode

In Ready mode (EXNW_MODE = 11), the SMC behaves differently. Normally, the SMC begins the access by down counting the setup and pulse counters of the read/write controlling signal. In the last cycle of the pulse phase, the resynchronized NWAIT signal is examined.

If asserted, the SMC suspends the access as shown in Figure 29-28 and Figure 29-29. After deassertion, the access is completed: the hold step of the access is performed.

This mode must be selected when the external device uses deassertion of the NWAIT signal to indicate its ability to complete the read or write operation.

If the NWAIT signal is deasserted before the end of the pulse, or asserted after the end of the pulse of the controlling read/write signal, it has no impact on the access length as shown in Figure 29-29.

Figure 29-28: NWAIT Assertion in Write Access: Ready Mode (EXNW_MODE = 11)

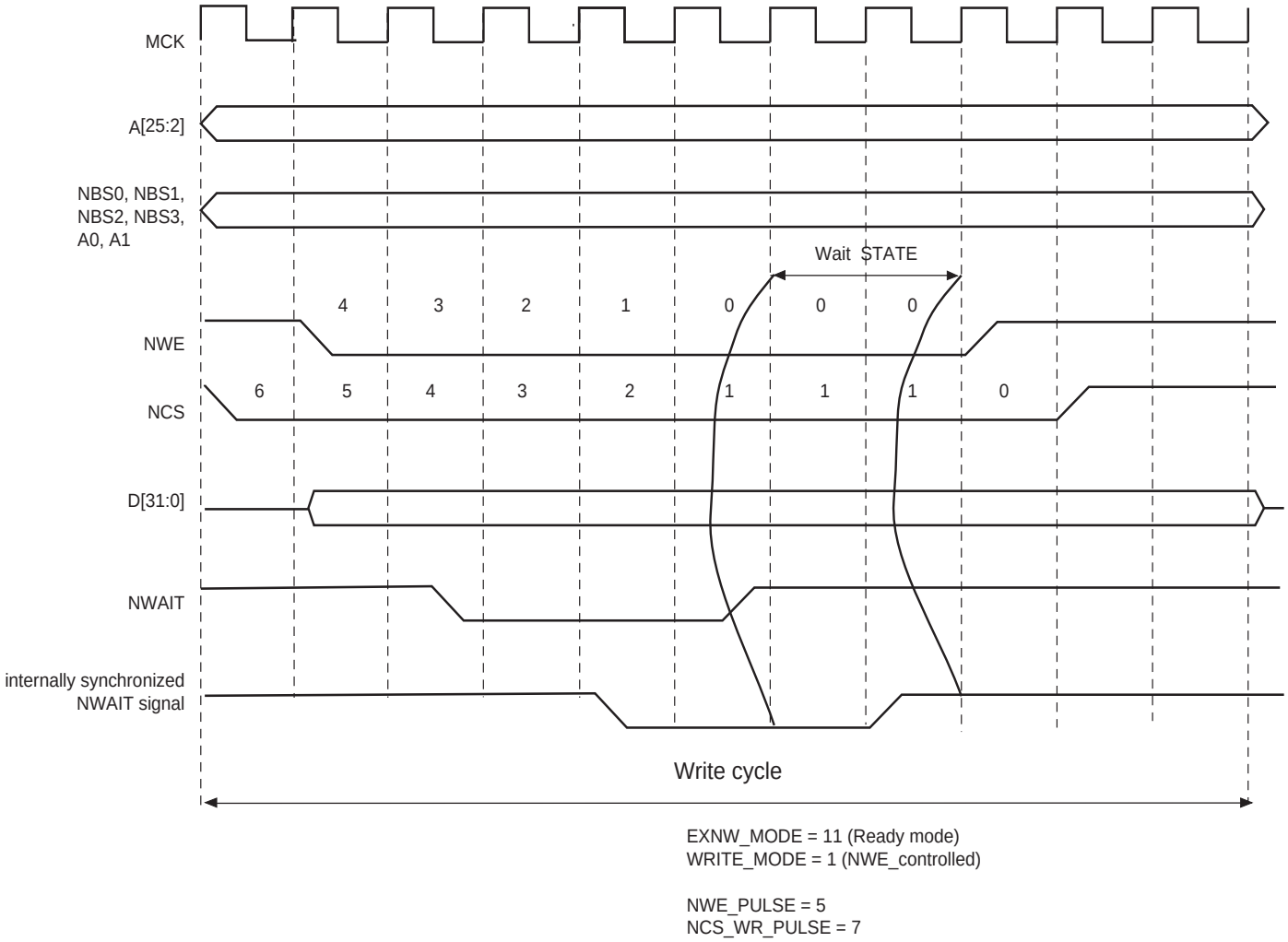
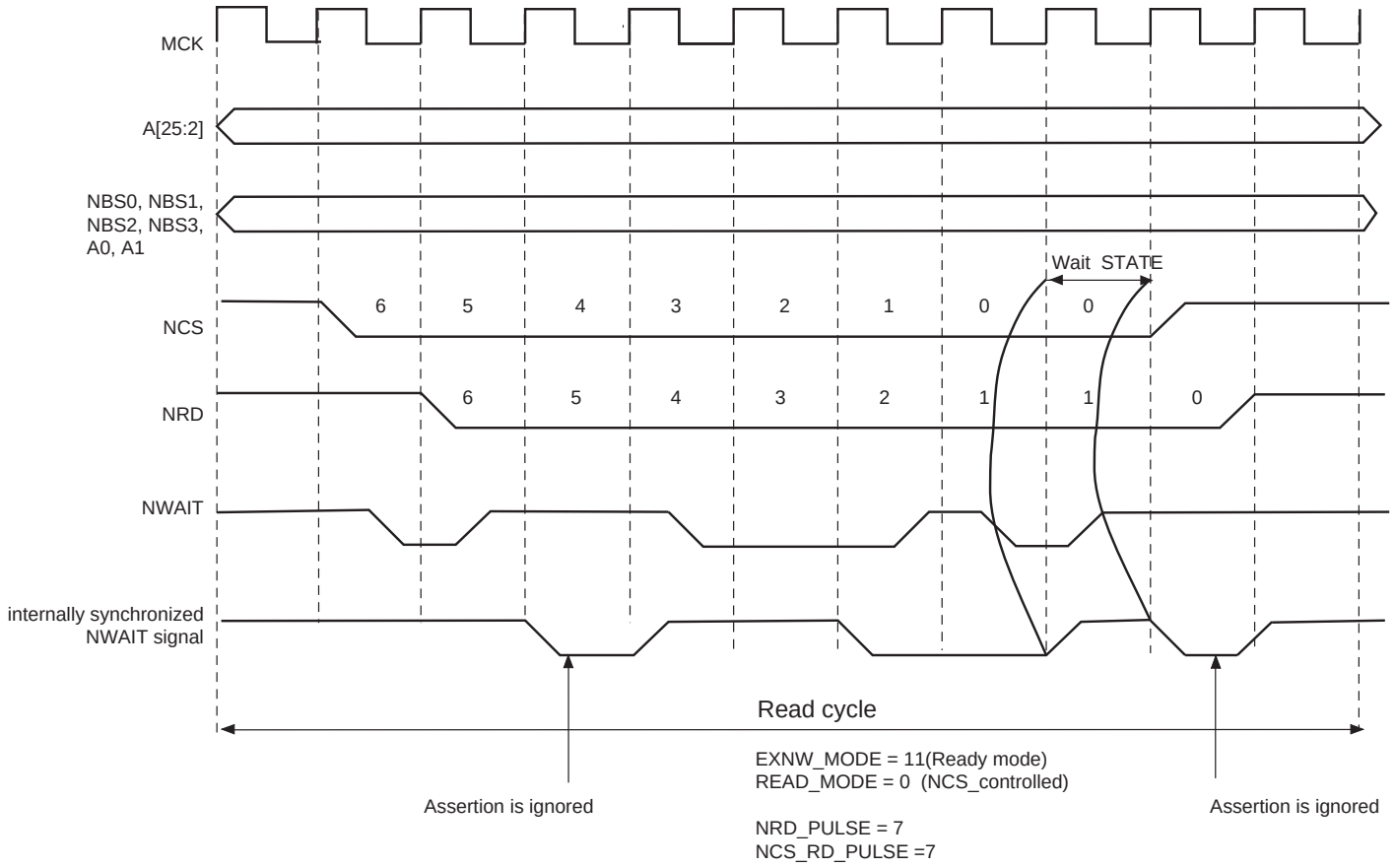


Figure 29-29: NWAIT Assertion in Read Access: Ready Mode (EXNW_MODE = 11)



© 2017 Microchip Technology Inc.

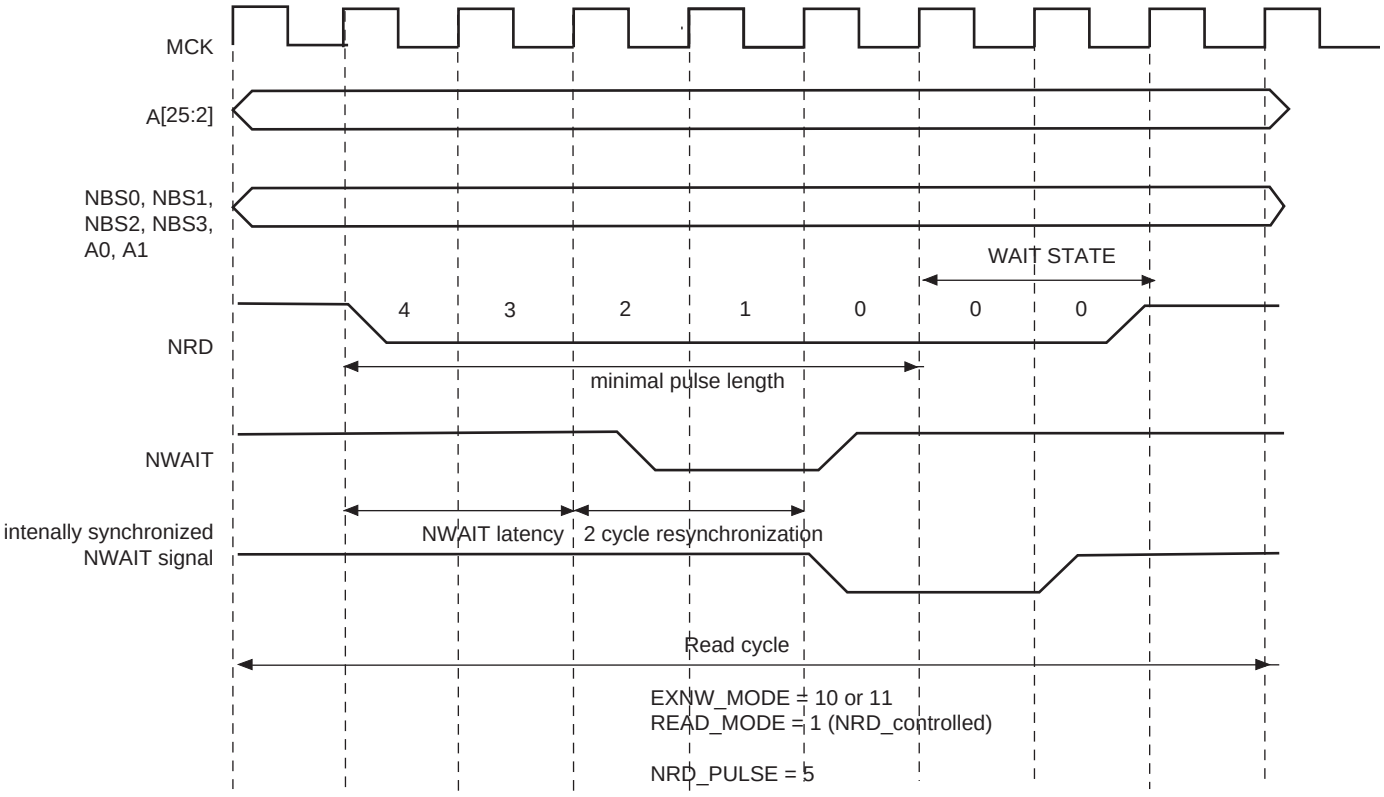
29.12.4 NWAIT Latency and Read/Write Timings

There may be a latency between the assertion of the read/write controlling signal and the assertion of the NWAIT signal by the device. The programmed pulse length of the read/write controlling signal must be at least equal to this latency plus the 2 cycles of resynchronization + 1 cycle. Otherwise, the SMC may enter the hold state of the access without detecting the NWAIT signal assertion. This is true in frozen mode as well as in ready mode. This is illustrated on Figure 29-30.

When EXNW_MODE is enabled (ready or frozen), the user must program a pulse length of the read and write controlling signal of at least:

minimal pulse length = NWAIT latency + 2 resynchronization cycles + 1 cycle

Figure 29-30: NWAIT Latency



29.13 Slow Clock Mode

The SMC is able to automatically apply a set of “slow clock mode” read/write waveforms when an internal signal driven by the Power Management Controller is asserted because MCK has been turned to a very slow clock rate (typically 32kHz clock rate). In this mode, the user-programmed waveforms are ignored and the slow clock mode waveforms are applied. This mode is provided so as to avoid reprogramming the User Interface with appropriate waveforms at very slow clock rate. When activated, the slow mode is active on all chip selects.

29.13.1 Slow Clock Mode Waveforms

Figure 29-31 illustrates the read and write operations in slow clock mode. They are valid on all chip selects. Table 29-6 indicates the value of read and write parameters in slow clock mode.

Figure 29-31: Read/write Cycles in Slow Clock Mode

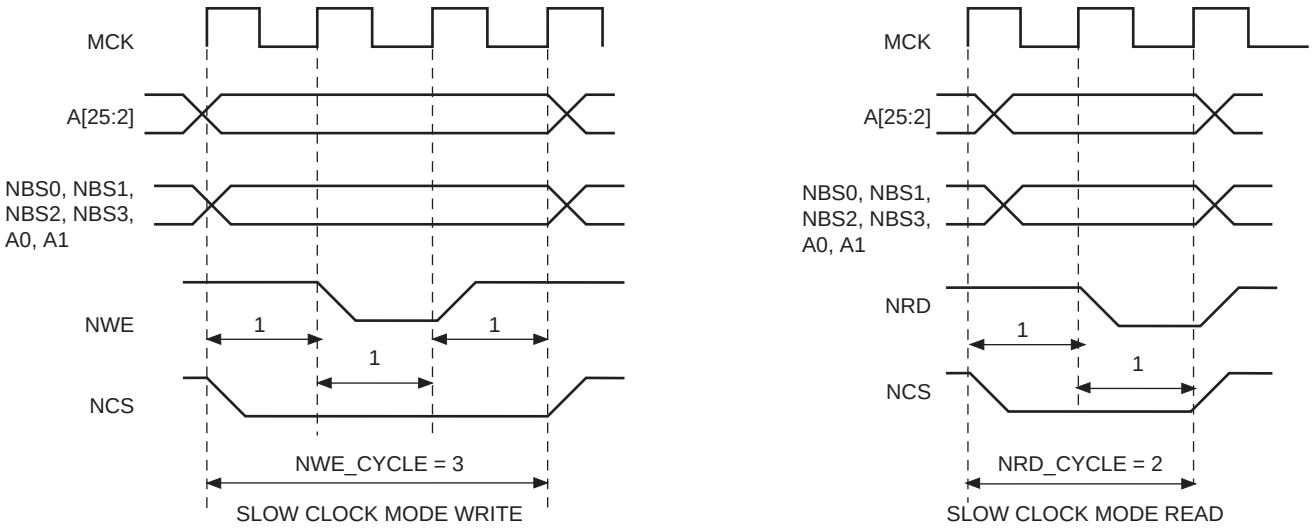


Table 29-6: Read and Write Timing Parameters in Slow Clock Mode

Read Parameters	Duration (cycles)	Write Parameters	Duration (cycles)
NRD_SETUP	1	NWE_SETUP	1
NRD_PULSE	1	NWE_PULSE	1
NCS_RD_SETUP	0	NCS_WR_SETUP	0
NCS_RD_PULSE	2	NCS_WR_PULSE	3
NRD_CYCLE	2	NWE_CYCLE	3

29.13.2 Switching from (to) Slow Clock Mode to (from) Normal Mode

When switching from slow clock mode to the normal mode, the current slow clock mode transfer is completed at high clock rate, with the set of slow clock mode parameters. See Figure 29-32. The external device may not be fast enough to support such timings.

Figure 29-33 illustrates the recommended procedure to properly switch from one mode to the other.

© 2017 Microchip Technology Inc.

Figure 29-32: Clock Rate Transition Occurs while the SMC is Performing a Write Operation

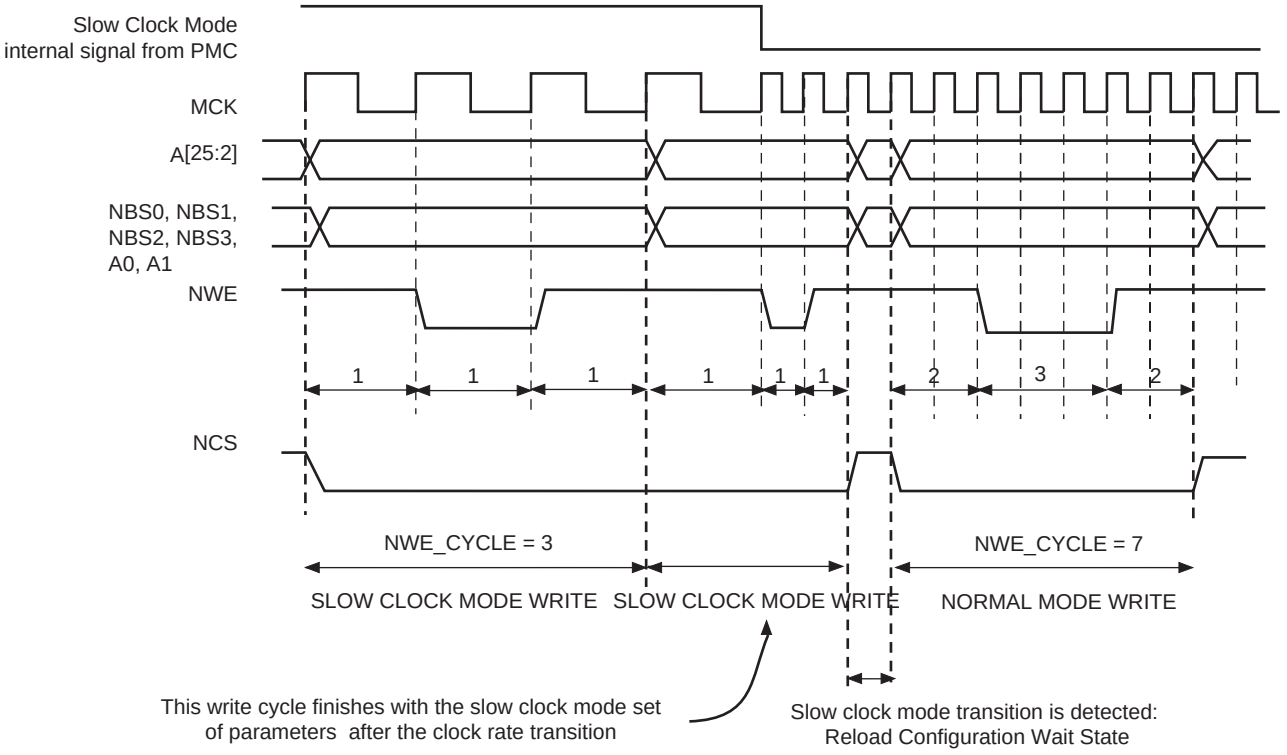
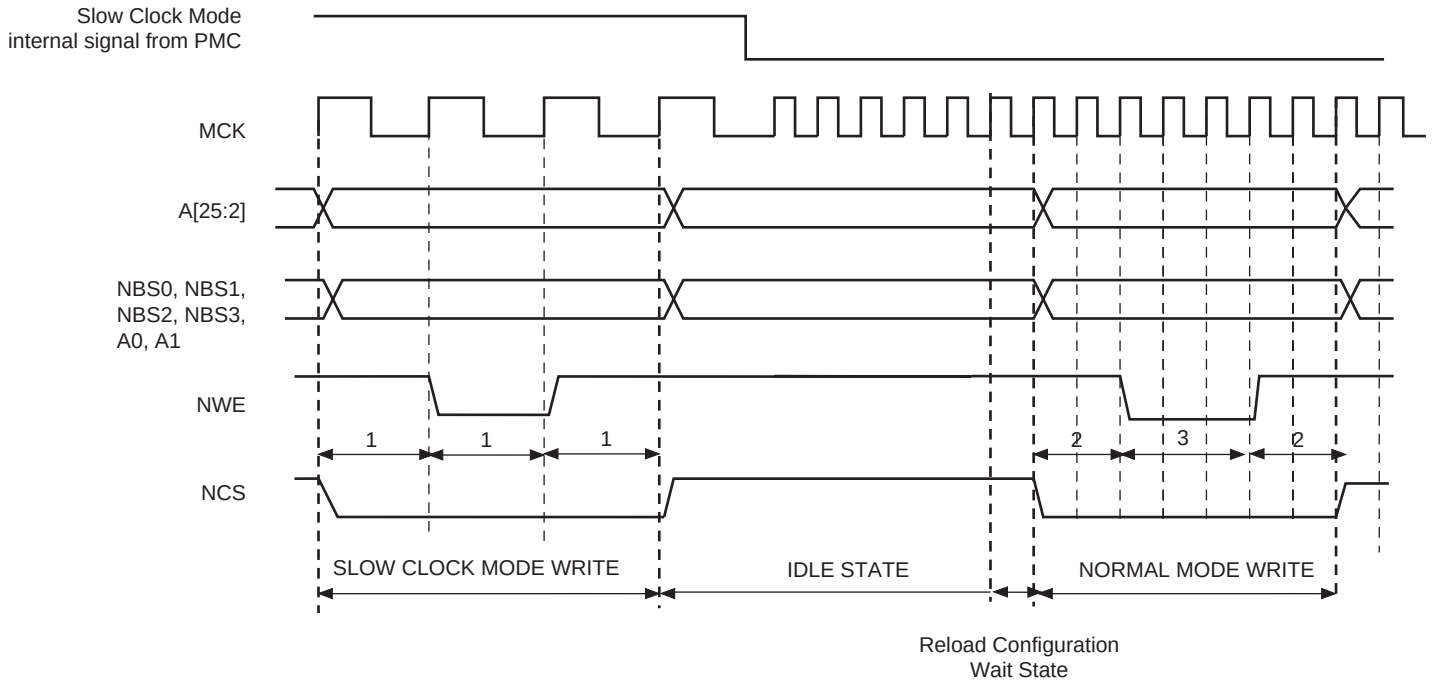


Figure 29-33: Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode



29.14 Asynchronous Page Mode

The SMC supports asynchronous burst reads in page mode, providing that the page mode is enabled in the SMC_MODE register (PMEN field). The page size must be configured in the SMC_MODE register (PS field) to 4, 8, 16 or 32 bytes.

The page defines a set of consecutive bytes into memory. A 4-byte page (resp. 8-, 16-, 32-byte page) is always aligned to 4-byte boundaries (resp. 8-, 16-, 32-byte boundaries) of memory. The MSB of data address defines the address of the page in memory, the LSB of address define the address of the data in the page as detailed in Table 29-7.

With page mode memory devices, the first access to one page (t_{pa}) takes longer than the subsequent accesses to the page (t_{sa}) as shown in Figure 29-34. When in page mode, the SMC enables the user to define different read timings for the first access within one page, and next accesses within the page.

Table 29-7: Page Address and Data Address within a Page

Page Size	Page Address (1)	Data Address in the Page (2)
4 bytes	A[25:2]	A[1:0]
8 bytes	A[25:3]	A[2:0]
16 bytes	A[25:4]	A[3:0]
32 bytes	A[25:5]	A[4:0]

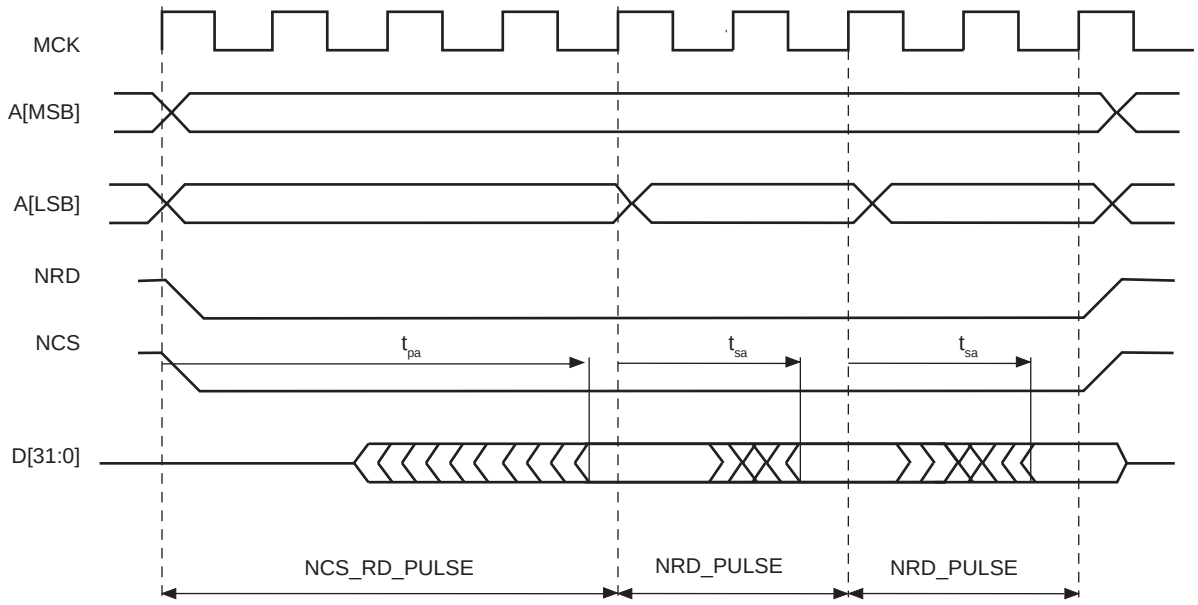
Note 1: 'A' denotes the address bus of the memory device

2: For 16-bit devices, the bit 0 of address is ignored. For 32-bit devices, bits [1:0] are ignored.

29.14.1 Protocol and Timings in Page Mode

Figure 29-34 shows the NRD and NCS timings in page mode access.

Figure 29-34: Page Mode Read Protocol (Address MSB and LSB are defined in Table 29-7)



The NRD and NCS signals are held low during all read transfers, whatever the programmed values of the setup and hold timings in the User Interface may be. Moreover, the NRD and NCS timings are identical. The pulse length of the first access to the page is defined with the NCS_RD_PULSE field of the SMC_PULSE register. The pulse length of subsequent accesses within the page are defined using the NRD_PULSE parameter.

The programming of the read timings in page mode is described in Table 29-8.

© 2017 Microchip Technology Inc.

Table 29-8: Programming of Read Timings in Page Mode

Parameter	Value	Definition
READ_MODE	'x'	No impact
NCS_RD_SETUP	'x'	No impact
NCS_RD_PULSE	t_{pa}	Access time of first access to the page
NRD_SETUP	'x'	No impact
NRD_PULSE	t_{sa}	Access time of subsequent accesses in the page
NRD_CYCLE	'x'	No impact

The SMC does not check the coherency of timings. It will always apply the NCS_RD_PULSE timings as page access timing (t_{pa}) and the NRD_PULSE for accesses to the page (t_{sa}), even if the programmed value for t_{pa} is shorter than the programmed value for t_{sa}.

29.14.2 Byte Access Type in Page Mode

The byte access type (BAT) configuration remains active in page mode. For 16-bit or 32-bit page mode devices that require byte selection signals, write a 0 to the BAT bit in the SMC Mode Register (SMC_MODE) to select the byte select access type.

29.14.3 Page Mode Restriction

The page mode is not compatible with the use of the NWAIT signal. Using the page mode and the NWAIT signal may lead to unpredictable behavior.

29.14.4 Sequential and Non-sequential Accesses

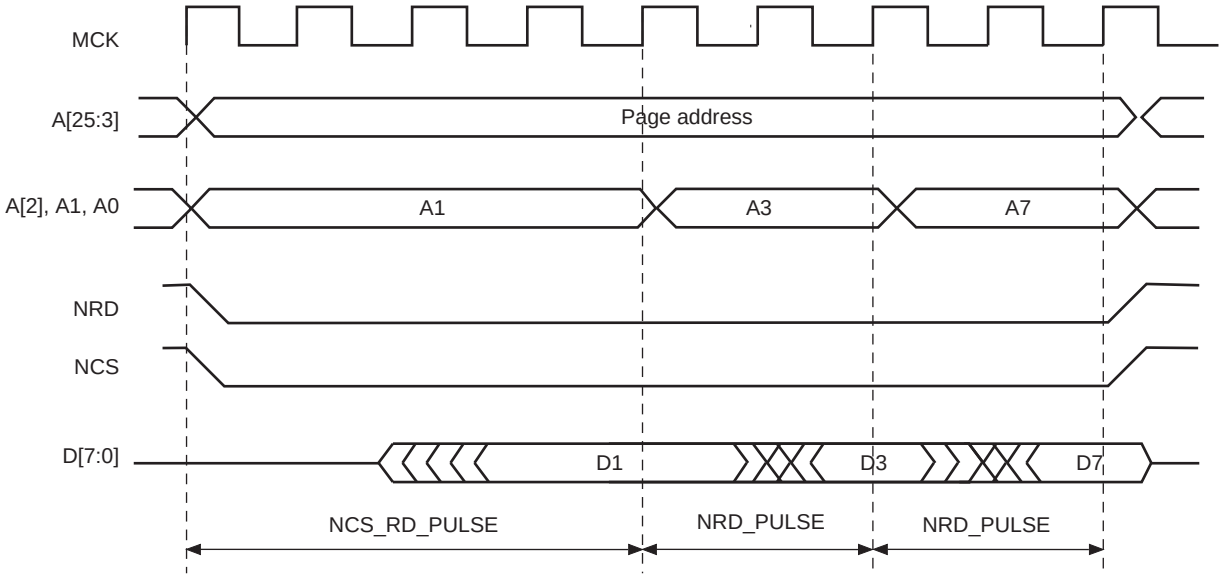
If the chip select and the MSB of addresses as defined in Table 29-7 are identical, then the current access lies in the same page as the previous one, and no page break occurs.

Using this information, all data within the same page, sequential or not sequential, are accessed with a minimum access time (t_{sa}). Figure 29-35 illustrates access to an 8-bit memory device in page mode, with 8-byte pages. Access to D1 causes a page access with a long access time (t_{pa}). Accesses to D3 and D7, though they are not sequential accesses, only require a short access time (t_{sa}).

If the MSB of addresses are different, the SMC performs the access of a new page. In the same way, if the chip select is different from the previous access, a page break occurs. If two sequential accesses are made to the page mode memory, but separated by an other internal or external peripheral access, a page break occurs on the second access because the chip select of the device was deasserted between both accesses.

© 2017 Microchip Technology Inc.

Figure 29-35: Access to Non-sequential Data within the Same Page



29.15 Register Write Protection

To prevent any single software error from corrupting SMC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the SMC Write Protection Mode Register (SMC_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the SMC Write Protection Status Register (SMC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the SMC_WPSR.

The following registers can be write-protected:

- SMC Setup Register
- SMC Pulse Register
- SMC Cycle Register
- SMC Mode Register

© 2017 Microchip Technology Inc.

29.16 Static Memory Controller (SMC) User Interface

The SMC is programmed using the registers listed in Table 29-9. For each chip select, a set of four registers is used to program the parameters of the external device connected on it. In Table 29-9, “CS_number” denotes the chip select number. Sixteen bytes (0x10) are required per chip select.

Note: The user must confirm the SMC configuration by writing any one of the SMC_MODE registers.

Table 29-9: Register Mapping

Offset	Register	Name	Access	Reset
0x10 x CS_number + 0x00	SMC Setup Register	SMC_SETUP	Read/Write	0x00000000
0x10 x CS_number + 0x04	SMC Pulse Register	SMC_PULSE	Read/Write	0x01010101
0x10 x CS_number + 0x08	SMC Cycle Register	SMC_CYCLE	Read/Write	0x00030003
0x10 x CS_number + 0x0C	SMC Mode Register	SMC_MODE	Read/Write	0x10001000
0xC0–0xDC	Reserved	–	–	–
0xE4	SMC Write Protection Mode Register	SMC_WPMR	Read/Write	0x00000000
0xE8	SMC Write Protection Status Register	SMC_WPSR	Read-only	0x00000000
0xEC–0xFC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

29.16.1 SMC Setup Register

Name:SMC_SETUP[0..5]

Address:0xFFFFEA00 [0], 0xFFFFEA10 [1], 0xFFFFEA20 [2], 0xFFFFEA30 [3], 0xFFFFEA40 [4], 0xFFFFEA50 [5]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	NCS_RD_SETUP					

23	22	21	20	19	18	17	16
–	–	NRD_SETUP					

15	14	13	12	11	10	9	8
–	–	NCS_WR_SETUP					

7	6	5	4	3	2	1	0
–	–	NWE_SETUP					

This register can only be written if the WPEN bit is cleared in the SMC Write Protection Mode Register.

NWE_SETUP: NWE Setup Length

The NWE signal setup length is defined as:

NWE setup length = (128* NWE_SETUP[5] + NWE_SETUP[4:0]) clock cycles

NCS_WR_SETUP: NCS Setup Length in WRITE Access

In write access, the NCS signal setup length is defined as:

NCS setup length = (128* NCS_WR_SETUP[5] + NCS_WR_SETUP[4:0]) clock cycles

NRD_SETUP: NRD Setup Length

The NRD signal setup length is defined in clock cycles as:

NRD setup length = (128* NRD_SETUP[5] + NRD_SETUP[4:0]) clock cycles

NCS_RD_SETUP: NCS Setup Length in READ Access

In read access, the NCS signal setup length is defined as:

NCS setup length = (128* NCS_RD_SETUP[5] + NCS_RD_SETUP[4:0]) clock cycles

© 2017 Microchip Technology Inc.

29.16.2 SMC Pulse Register

Name:SMC_PULSE[0..5]

Address:0xFFFFEA04 [0], 0xFFFFEA14 [1], 0xFFFFEA24 [2], 0xFFFFEA34 [3], 0xFFFFEA44 [4], 0xFFFFEA54 [5]

Access: Read/Write

31	30	29	28	27	26	25	24
–	NCS_RD_PULSE						

23	22	21	20	19	18	17	16
–	NRD_PULSE						

15	14	13	12	11	10	9	8
–	NCS_WR_PULSE						

7	6	5	4	3	2	1	0
–	NWE_PULSE						

This register can only be written if the WPEN bit is cleared in the SMC Write Protection Mode Register.

NWE_PULSE: NWE Pulse Length

The NWE signal pulse length is defined as:

NWE pulse length = (256* NWE_PULSE[6] + NWE_PULSE[5:0]) clock cycles

The NWE pulse length must be at least 1 clock cycle.

NCS_WR_PULSE: NCS Pulse Length in WRITE Access

In write access, the NCS signal pulse length is defined as:

NCS pulse length = (256* NCS_WR_PULSE[6] + NCS_WR_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

NRD_PULSE: NRD Pulse Length

In standard read access, the NRD signal pulse length is defined in clock cycles as:

NRD pulse length = (256* NRD_PULSE[6] + NRD_PULSE[5:0]) clock cycles

The NRD pulse length must be at least 1 clock cycle.

In page mode read access, the NRD_PULSE parameter defines the duration of the subsequent accesses in the page.

NCS_RD_PULSE: NCS Pulse Length in READ Access

In standard read access, the NCS signal pulse length is defined as:

NCS pulse length = (256* NCS_RD_PULSE[6] + NCS_RD_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

In page mode read access, the NCS_RD_PULSE parameter defines the duration of the first access to one page.

© 2017 Microchip Technology Inc.

29.16.3 SMC Cycle Register

Name:SMC_CYCLE[0..5]

Address:0xFFFFEA08 [0], 0xFFFFEA18 [1], 0xFFFFEA28 [2], 0xFFFFEA38 [3], 0xFFFFEA48 [4], 0xFFFFEA58 [5]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	NRD_CYCLE

23	22	21	20	19	18	17	16
NRD_CYCLE							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NWE_CYCLE

7	6	5	4	3	2	1	0
NWE_CYCLE							

This register can only be written if the WPEN bit is cleared in the SMC Write Protection Mode Register.

NWE_CYCLE: Total Write Cycle Length

The total write cycle length is the total duration in clock cycles of the write cycle. It is equal to the sum of the setup, pulse and hold steps of the NWE and NCS signals. It is defined as:

Write cycle length = (NWE_CYCLE[8:7]*256 + NWE_CYCLE[6:0]) clock cycles

NRD_CYCLE: Total Read Cycle Length

The total read cycle length is the total duration in clock cycles of the read cycle. It is equal to the sum of the setup, pulse and hold steps of the NRD and NCS signals. It is defined as:

Read cycle length = (NRD_CYCLE[8:7]*256 + NRD_CYCLE[6:0]) clock cycles

© 2017 Microchip Technology Inc.

29.16.4 SMC Mode Register

Name:SMC_MODE[0..5]

Address:0xFFFFEA0C [0], 0xFFFFEA1C [1], 0xFFFFEA2C [2], 0xFFFFEA3C [3], 0xFFFFEA4C [4], 0xFFFFEA5C [5]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	PS		–	–	–	PMEN
23	22	21	20	19	18	17	16
–	–	–	TDF_MODE	TDF_CYCLES			
15	14	13	12	11	10	9	8
–	–	DBW		–	–	–	BAT
7	6	5	4	3	2	1	0
–	–	EXNW_MODE		–	–	WRITE_MODE	READ_MODE

This register can only be written if the WPEN bit is cleared in the SMC Write Protection Mode Register.

The user must confirm the SMC configuration by writing any one of the SMC_MODE registers.

READ_MODE: Selection of the Control Signal for Read Operation

Value	Name	Description
0	NCS_CTRL	Read operation controlled by NCS signal - If TDF cycles are programmed, the external bus is marked busy after the rising edge of NCS. - If TDF optimization is enabled (TDF_MODE = 1), TDF wait states are inserted after the setup of NCS.
1	NRD_CTRL	Read operation controlled by NRD signal - If TDF cycles are programmed, the external bus is marked busy after the rising edge of NRD. - If TDF optimization is enabled (TDF_MODE = 1), TDF wait states are inserted after the setup of NRD.

WRITE_MODE: Selection of the Control Signal for Write Operation

Value	Name	Description
0	NCS_CTRL	Write operation controlled by NCS signal—If TDF optimization is enabled (TDF_MODE = 1), TDF wait states will be inserted after the setup of NCS.
1	NWE_CTRL	Write operation controlled by NWE signal—If TDF optimization is enabled (TDF_MODE = 1), TDF wait states will be inserted after the setup of NWE.

EXNW_MODE: NWAIT Mode

The NWAIT signal is used to extend the current read or write signal. It is only taken into account during the pulse phase of the read and write controlling signal. When the use of NWAIT is enabled, at least one cycle hold duration must be programmed for the read and write controlling signal.

Value	Name	Description
00	DISABLED	Disabled Mode—The NWAIT input signal is ignored on the corresponding Chip Select.
01	—	Reserved
10	FROZEN	Frozen Mode—If asserted, the NWAIT signal freezes the current read or write cycle. After deassertion, the read/write cycle is resumed from the point where it was stopped.
11	READY	Ready Mode—The NWAIT signal indicates the availability of the external device at the end of the pulse of the controlling read or write signal, to complete the access. If high, the access normally completes. If low, the access is extended until NWAIT returns high.

BAT: Byte Access Type

This field is used only if DBW defines a 16- or 32-bit data bus.

Value	Name	Description
0	BYTE_SELECT	Byte select access type: - Write operation is controlled using NCS, NWE, NBS0, NBS1, NBS2 and NBS3 - Read operation is controlled using NCS, NRD, NBS0, NBS1, NBS2 and NBS3
1	BYTE_WRITE	Byte write access type: - Write operation is controlled using NCS, NWR0, NWR1, NWR2, NWR3 - Read operation is controlled using NCS and NRD

DBW: Data Bus Width

Value	Name	Description
00	BIT_8	8-bit bus
01	BIT_16	16-bit bus
10	BIT_32	32-bit bus
11	—	Reserved

TDF_CYCLES: Data Float Time

This field gives the integer number of clock cycles required by the external device to release the data after the rising edge of the read controlling signal. The SMC always provides one full cycle of bus turnaround after the TDF_CYCLES period. The external bus cannot be used by another chip select during TDF_CYCLES + 1 cycles. From 0 up to 15 TDF_CYCLES can be set.

TDF_MODE: TDF Optimization

1: TDF optimization enabled—The number of TDF wait states is optimized using the setup period of the next read/write access.

0: TDF optimization disabled—The number of TDF wait states is inserted before the next access begins.

PMEN: Page Mode Enabled

1: Asynchronous burst read in page mode is applied on the corresponding chip select.

0: Standard read is applied.

PS: Page Size

If page mode is enabled, this field indicates the size of the page in bytes.

Value	Name	Description
00	BYTE_4	4-byte page
01	BYTE_8	8-byte page
10	BYTE_16	16-byte page
11	BYTE_32	32-byte page

© 2017 Microchip Technology Inc.

29.16.5 SMC Write Protection Mode Register

Name:SMC_WPMR

Address:0xFFFFEAE4

Access: Read//Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPEN

WPEN: Write Protection Enable

0: Disables write protection if WPKEY value corresponds to 0x534D43 (“SMC” in ASCII).

1: Enables write protection if WPKEY value corresponds to 0x534D43 (“SMC” in ASCII).

See Section 29.9.5 ”Coding Timing Parameters” for list of write-protected registers.

WPKEY: Write Protection Key

Value	Name	Description
0x534D43	PASSWD	Writing any other value in this field aborts the write operation of bit WPEN. Always reads as 0.

29.16.6 SMC Write Protection Status Register

Name:SMC_WPSR

Address:0xFFFFEAE8

Access: Read-only

31	30	29	28	27	26	25	24
—	—	—	—	—	—	—	—

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPVS

WPVS: Write Protection Violation Status

0: No write protection violation has occurred since the last read of the SMC_WPSR.

1: A write protection violation occurred since the last read of the SMC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

WPVSRC: Write Protection Violation Source

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

© 2017 Microchip Technology Inc.

30. DDR SDR SDRAM Controller (DDRSDRC)

30.1 Description

The DDR SDR SDRAM Controller (DDRSDRC) is a multiport memory controller. It comprises four slave AHB interfaces. All simultaneous accesses (four independent AHB ports) are interleaved to maximize memory bandwidth and minimize transaction latency due to SDRAM protocol.

The DDRSDRC extends the memory capabilities of a chip by providing the interface to an external 16-bit or 32-bit SDR-SDRAM device and external 16-bit DDR-SDRAM device. The page size supports ranges from 2048 to 16384 rows and from 256 to 4096 columns. It supports byte (8-bit), half-word (16-bit) and word (32-bit) accesses.

The DDRSDRC supports a read or write burst length (BL) of eight locations. This enables the command and address bus to anticipate the next command, thus reducing latency imposed by the SDRAM protocol and improving the SDRAM bandwidth. Moreover, the DDRSDRC keeps track of the active row in each bank, thus maximizing SDRAM performance, e.g., the application may be placed in one bank and data in the other banks. To optimize performance, it is advisable to avoid accessing different rows in the same bank. The DDRSDRC supports a CAS latency of 2 or 3 and optimizes the read access depending on the frequency.

The features of Self refresh, Power-down, and Deep Power-down modes minimize the consumption of the SDRAM device.

The DDRSDRC user interface is compliant with Arm Advanced Peripheral Bus (APB rev2).

Note: The term “SDRAM device” regroups SDR-SDRAM, Low-power SDR-SDRAM, Low-power DDR1-SDRAM and DDR2-SDRAM devices.

30.2 Embedded Characteristics

- AMBA compliant interface, interfaces directly to the Arm advanced high performance bus (AHB)
  - Four AHB interfaces, management of all accesses maximizes memory bandwidth and minimizes transaction latency
  - AHB transfer: Word, Half-Word, Byte access
- Supports DDR2-SDRAM, Low-power DDR1-SDRAM, SDR-SDRAM and Low-power SDR-SDRAM
- Numerous configurations supported
  - 2K, 4K, 8K, 16K row address memory parts
  - SDRAM with four and eight internal banks
  - SDR-SDRAM with 16- or 32-bit data path
  - DDR-SDRAM with 16-bit data path
  - One Chip Select for SDRAM device (256 Mbyte address space)
- Programming facilities
  - Multibank ping-pong access (up to or 4 banks or 8 banks opened at same time = reduces average latency of transactions)
  - Timing parameters specified by software
  - Automatic refresh operation, refresh rate is programmable
  - Automatic update of DS and PASR parameters (Low-power SDRAM Devices)
- Energy-saving capabilities
  - Self refresh, Power-down, Active Power-down and Deep Power-down modes supported
- SDRAM power-up initialization by software
- CAS Latency of 2, 3 supported
- Reset function supported (DDR2-SDRAM)
- ODT (On-die Termination) not supported
- Auto Precharge command not used
- SDR-SDRAM with 16-bit datapath and eight columns not supported
- DDR2-SDRAM with eight internal banks supported
- Linear and interleaved decoding supported
- SDR-SDRAM or Low-power DDR1-SDRAM with 2 internal banks not supported
- Clock frequency change in Precharge Power-down mode not supported
- OCD (Off-chip Driver) mode not supported

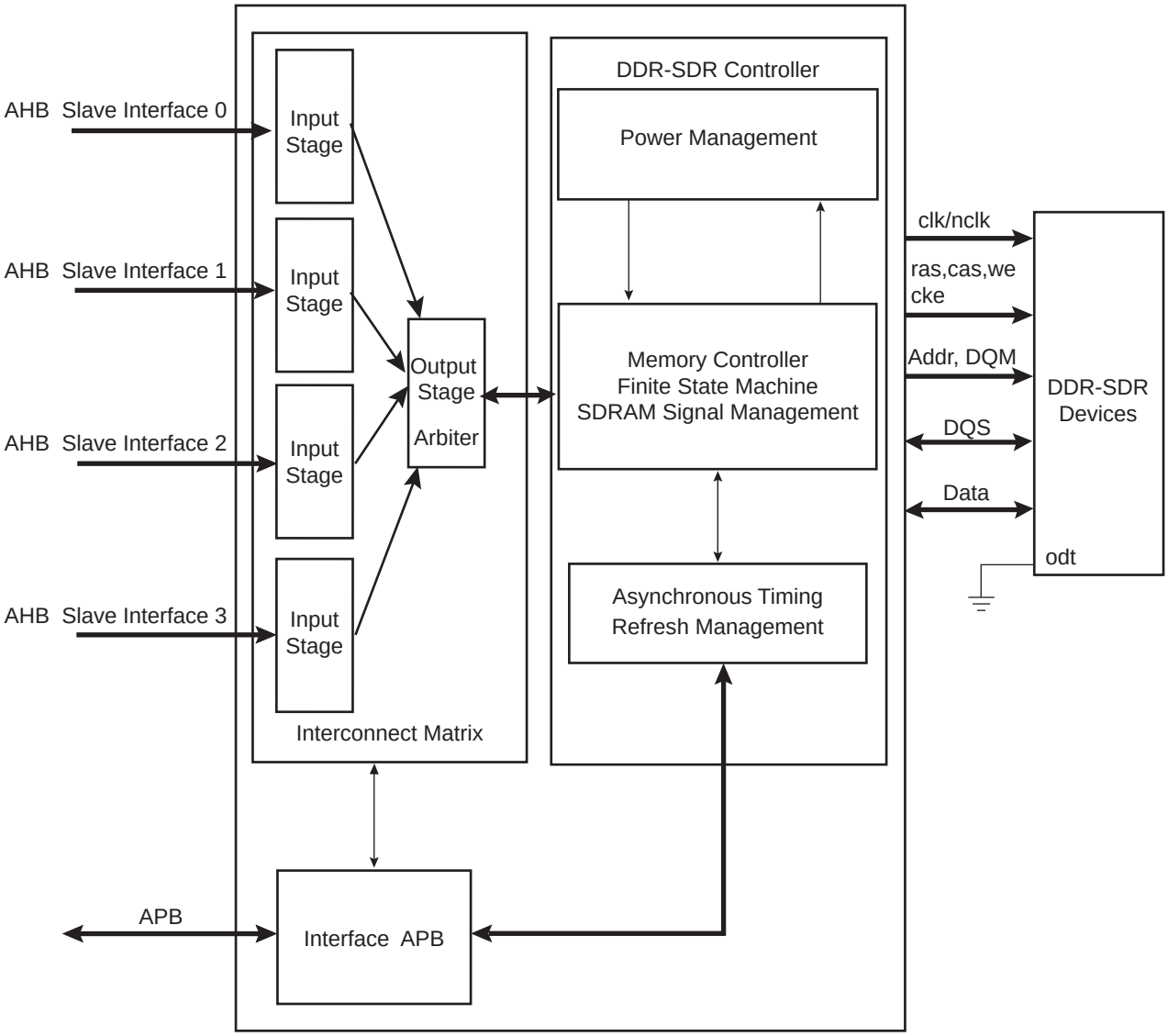
© 2017 Microchip Technology Inc.

30.3 DDRSDRC Module Diagram

The DDRSDRC is partitioned in two blocks (see Figure 30-1):

- An Interconnect-Matrix that manages concurrent accesses on the AHB bus between four AHB masters and integrates an arbiter.
- A controller that translates AHB requests (Read/Write) in the SDRAM protocol.

Figure 30-1: DDRSDRC Module Diagram



30.4 Initialization Sequence

The addresses given are for example purposes only. The real address depends on implementation in the product.

30.4.1 SDR-SDRAM Initialization

The initialization sequence is generated by software. The following sequence initializes SDR-SDRAM devices:

1. Program the memory device type in the Memory Device Register (see "DDRSDRC Memory Device Register").
2. Program the features of the SDR-SDRAM device in the Timing Register (asynchronous timing (trc, tras, etc.)), and in the Configuration Register (number of columns, rows, banks, CAS latency) (see "DDRSDRC Configuration Register" , "DDRSDRC Timing Parameter 0 Register" and "DDRSDRC Timing Parameter 1 Register").
3. For low-power SDRAM, drive strength (DS) and partial array self refresh (PASR) must be set in the Low-power Register (see

© 2017 Microchip Technology Inc.

"DDRSDRC Low-power Register").

A minimum pause of 200 µs is provided to precede any signal toggle.

4. A NOP command is issued to the SDR-SDRAM. To program the NOP command, the application must configure the MODE field value to 1 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to any SDR-SDRAM address to acknowledge this command. Now the clock which drives SDR-SDRAM device is enabled.
5. An All Banks Precharge command is issued to the SDR-SDRAM. To program All Banks Precharge command, the application must configure the MODE field value to 2 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to any SDR-SDRAM address to acknowledge this command.
6. Eight CAS before RAS (CBR) auto-refresh cycles are provided. To program the auto refresh command (CBR), the application must configure the MODE field value to 4 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to any SDR-SDRAM location eight times to acknowledge these commands.
7. A Mode Register set (MRS) cycle is issued to program the parameters of the SDR-SDRAM devices, in particular CAS latency and burst length. The application must configure the MODE field value to 3 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to the SDR-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] are set to 0. For example, with a 16-bit 128 MB SDR-SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be performed at the address 0x20000000.

   Note: This address is for example purposes only. The real address is dependent on implementation in the product.
8. For low-power SDR-SDRAM initialization, an Extended Mode Register set (EMRS) cycle is issued to program the SDR-SDRAM parameters (PASR, DS acronyms in JEDEC datasheet). The application must configure the MODE field value to 5 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to the SDR-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 1 and BA[0] is set to 0. For example, with a 16-bit 128 MB SDRAM, (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be performed at the address 0x20800000.
9. The application must go into Normal mode by configuring the MODE field value to 0 in the Mode Register (see "DDRSDRC Mode Register") and performing a write access at any location in the SDRAM to acknowledge this command.
10. Write the refresh rate into the COUNT field in the DDRSDRC Refresh Timer Register (DDRSDRC_RTR). (Refresh rate = delay between refresh cycles). The SDR-SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, DDRSDRC_RTR.COUNT must be configured to 15.625 × 100 MHz = 1562 (0x061A) or 7.81 × 100 MHz = 781 (0x030D).

After initialization, the SDR-SDRAM device is fully functional.

30.4.2 Low-power DDR1-SDRAM Initialization

The initialization sequence is generated by software. The following sequence initializes low-power DDR1-SDRAM devices:

1. Program the memory device type in the Memory Device Register (see "DDRSDRC Memory Device Register").
2. Program the features of the low-power DDR1-SDRAM device in the Configuration Register: asynchronous timing (TRC, TRAS, etc.), number of columns, rows, banks, CAS latency. See "DDRSDRC Configuration Register" , "DDRSDRC Timing Parameter 0 Register" and "DDRSDRC Timing Parameter 1 Register" .
3. Program Partial array self refresh (PASR) and Drive strength (DS) in the Low-power Register. See "DDRSDRC Low-power Register" .
4. An NOP command will be issued to the low-power DDR1-SDRAM. To program the NOP command, the application must configure the MODE field value to 1 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to any DDR1-SDRAM address to acknowledge this command. Now clocks which drive DDR1-SDRAM device are enabled.

A minimum pause of 200 µs will be provided to precede any signal toggle.

5. An All Banks Precharge command is issued to the low-power DDR1-SDRAM. To program the All Banks Precharge command, the application must configure the MODE field value to 2 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to any low-power DDR1-SDRAM address to acknowledge this command.
6. Two CAS before RAS (CBR) auto-refresh cycles are provided. To program the auto refresh command (CBR), the application must configure the MODE field value to 4 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to any low-power DDR1-SDRAM location twice to acknowledge these commands.
7. An Extended Mode Register set (EMRS) cycle is issued to program the low-power DDR1-SDRAM parameters (carried on PASR, DS fields in DDRSDRC Low-power Register (DDRSDRC_LPR). The application must configure the MODE field value to 5 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to the SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 1 BA[0] is set to 0. For example, with a 16-bit 128 MB SDRAM (12 rows, 9 columns, 4 banks) bank address, the low-power DDR1-SDRAM write access should be performed at address 0x20800000.

   Note: This address is for example purposes only. The real address is dependent on implementation in the product.
8. A Mode Register set (MRS) cycle is issued to program the parameters of the low-power DDR1-SDRAM devices, in particular CAS

latency, burst length. The application must configure the MODE field value to 3 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to the low-power DDR1-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] bits are set to 0. For example, with a 16-bit 128 MB low-power DDR1-SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be performed at the address 0x20000000. The application must go into Normal mode by configuring the MODE field value to 0 in the Mode Register (see "DDRSDRC Mode Register") and performing a write access at any location in the low-power DDR1-SDRAM to acknowledge this command.

9. Perform a write access to any low-power DDR1-SDRAM address.
10. Write the refresh rate into the COUNT field in the DDRSDRC Refresh Timer Register (DDRSDRC_RTR). (Refresh rate = delay between refresh cycles). The low-power DDR1-SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, DDRSDRC_RTR.COUNT must be configured to 15.625 × 100 MHz = 1562 (0x061A) or 7.81 × 100 MHz = 781 (0x030D).
11. After initialization, the low-power DDR1-SDRAM device is fully functional.

30.4.3 DDR2-SDRAM Initialization

The initialization sequence is generated by software. The following sequence initializes DDR2-SDRAM devices:

1. Program the memory device type in the Memory Device Register (see "DDRSDRC Memory Device Register").
2. Program the features of DDR2-SDRAM device in the Timing Register (asynchronous timing (trc, tras, etc.)), and in the Configuration Register (number of columns, rows, banks, CAS latency and output drive strength) (see "DDRSDRC Configuration Register" , "DDRSDRC Timing Parameter 0 Register" and "DDRSDRC Timing Parameter 1 Register").
3. An NOP command is issued to the DDR2-SDRAM. To program the NOP command, the application must configure the MODE field value to 1 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to any DDR2-SDRAM address to acknowledge this command. Now clocks which drive DDR2-SDRAM device are enabled.

A minimum pause of 200 µs is provided to precede any signal toggle.

4. An NOP command is issued to the DDR2-SDRAM. To program the NOP command, the application must configure the MODE field value to 1 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to any DDR2-SDRAM address to acknowledge this command. Now CKE is driven high.
5. An All Banks Precharge command is issued to the DDR2-SDRAM. To program the All Banks Precharge command, the application must configure the MODE field value to 2 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to any DDR2-SDRAM address to acknowledge this command.
6. An Extended Mode Register set (EMRS2) cycle is issued to choose between commercial or high temperature operations. The application must configure the MODE field value to 5 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 1 and BA[0] is set to 0. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR2-SDRAM write access should be performed at the address 0x20800000.

   Note: This address is for example purposes only. The real address is dependent on implementation in the product.

7. An Extended Mode Register set (EMRS3) cycle is issued to set the Extended Mode Register to 0. The application must configure the MODE field value to 5 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 1 and BA[0] is set to 1. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR2-SDRAM write access should be performed at the address 0x20C00000.
8. An Extended Mode Register set (EMRS1) cycle is issued to enable DLL. The application must configure the MODE field value to 5 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 0 and BA[0] is set to 1. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR2-SDRAM write access should be performed at the address 0x20400000.

An additional 200 cycles of clock are required for locking DLL.

9. Program "Enable DLL reset" by setting the DLL bit in the Configuration Register (see "DDRSDRC Configuration Register").
10. A Mode Register set (MRS) cycle is issued to reset DLL. The application must configure the MODE field value to 3 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] bits are set to 0. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be performed at the address 0x20000000.
11. An All Banks Precharge command is issued to the DDR2-SDRAM. To program the All Banks Precharge command, the application must configure the MODE field value to 2 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to any DDR2-SDRAM address to acknowledge this command.
12. Two CAS before RAS (CBR) auto-refresh cycles are provided. To program the auto refresh command (CBR), the application must configure the MODE field value to 4 in the Mode Register (see "DDRSDRC Mode Register") and perform a write access to any

© 2017 Microchip Technology Inc.

DDR2-SDRAM location twice to acknowledge these commands.

13. Program “Disable DLL reset” by clearing DLL bit in the Configuration Register (see “DDRSDRC Configuration Register”).
14. A Mode Register set (MRS) cycle is issued to program the parameters of the DDR2-SDRAM devices, in particular CAS latency, burst length and to disable DLL reset. The application must configure the MODE field value to 3 in the Mode Register (see “DDRSDRC Mode Register”) and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] are set to 0. For example, with a 16-bit 128 MB SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be performed at the address 0x20000000.
15. Program “OCD calibration default” by configuring the OCD field value to 7 in the Configuration Register (see “DDRSDRC Configuration Register”).
16. An Extended Mode Register set (EMRS1) cycle is issued to OCD default value. The application must configure the MODE field value to 5 in the Mode Register (see “DDRSDRC Mode Register”) and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 0 and BA[0] is set to 1. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR2-SDRAM write access should be performed at the address 0x20400000.
17. Program “Exit from OCD calibration mode” by configuring the OCD field value to 0 in the Configuration Register (see “DDRSDRC Configuration Register”).
18. An Extended Mode Register set (EMRS1) cycle is issued to enable OCD exit. The application must configure the MODE field value to 5 in the Mode Register (see “DDRSDRC Mode Register”) and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 0 and BA[0] is set to 1. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR2-SDRAM write access should be performed at the address 0x20400000.
19. Program the Normal mode in the Mode Register (see “DDRSDRC Mode Register”) and perform a write access to any DDR2-SDRAM address to acknowledge this command.
20. Perform a write access to any DDR2-SDRAM address.
21. Write the refresh rate into the COUNT field in the DDRSDRC Refresh Timer Register (DDRSDRC_RTR). (Refresh rate = delay between refresh cycles). The DDR2-SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 133 MHz frequency, DDRSDRC_RTR.COUNT must be configured to 15.625 × 133 MHz = 2079 (0x081F) or 7.81 × 133 MHz = 1039 (0x040F).

After initialization, the DDR2-SDRAM devices are fully functional.

30.5 Functional Description

30.5.1 SDRAM Controller Write Cycle

The DDRSDRC allows burst access or single access in Normal mode (DDRSDRC_MR.MODE = 0). Whatever the access type, the DDRSDRC keeps track of the active row in each bank, thus maximizing performance.

The SDRAM device is programmed with a burst length equal to 8. This determines the length of a sequential data input by the write command that is set to 8. The latency from write command to data input is fixed to 1 in the case of DDR-SDRAM devices. In the case of SDR-SDRAM devices, there is no latency from write command to data input.

To initiate a single access, the DDRSDRC checks if the page access is already open. If row/bank addresses match with the previous row/bank addresses, the controller generates a write command. If the bank addresses are not identical or if bank addresses are identical but the row addresses are not identical, the controller generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (T_{RP}) commands and active/write (TRCD) command. As the burst length is fixed to 8, in the case of single access, it has to stop the burst, otherwise seven invalid values may be written. In the case of SDR-SDRAM devices, a Burst Stop command is generated to interrupt the write operation. In the case of DDR-SDRAM devices, Burst Stop command is not supported for the burst write operation. In order to then interrupt the write operation, DM must be set to 1 to mask invalid data (see Figure 30-2 and Figure 30-5) and DQS must continue to toggle.

To initiate a burst access, the DDRSDRC uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the SDRAM device is carried out. If the next access is a write non-sequential access, then an automatic access break is inserted, the DDRSDRC generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (TRP) commands and active/write (TRCD) commands.

For a definition of timing parameters, refer to Section 30.7.4 ”DDRSDRC Timing Parameter 0 Register”.

Write accesses to the SDRAM devices are burst oriented and the burst length is programmed to 8. It determines the maximum number of column locations that can be accessed for a given write command. When the write command is issued, eight columns are selected. All accesses for that burst take place within these eight columns, thus the burst wraps within these eight columns if a boundary is reached. These eight columns are selected by addr[13:3]. addr[2:0] is used to select the starting location within the block.

© 2017 Microchip Technology Inc.

In the case of incrementing burst (INCR/INCR4/INCR8/INCR16), the addresses can cross the 16-byte boundary of the SDRAM device. For example, in the case of DDR-SDRAM devices, when a transfer (INCR4) starts at address 0x0C, the next access is 0x10, but since the burst length is programmed to 8, the next access is at 0x00. Since the boundary is reached, the burst is wrapping. The DDRSDRC takes this feature of the SDRAM device into account. In the case of transfer starting at address 0x04/0x08/0x0C (DDR-SDRAM devices) or starting at address 0x10/0x14/0x18/0x1C, two write commands are issued to avoid to wrap when the boundary is reached. The last write command is subject to DM input logic level. If DM is registered high, the corresponding data input is ignored and write access is not performed. This avoids additional writing being done.

Figure 30-2: Single Write Access, Row Closed, Low-power DDR1-SDRAM Device

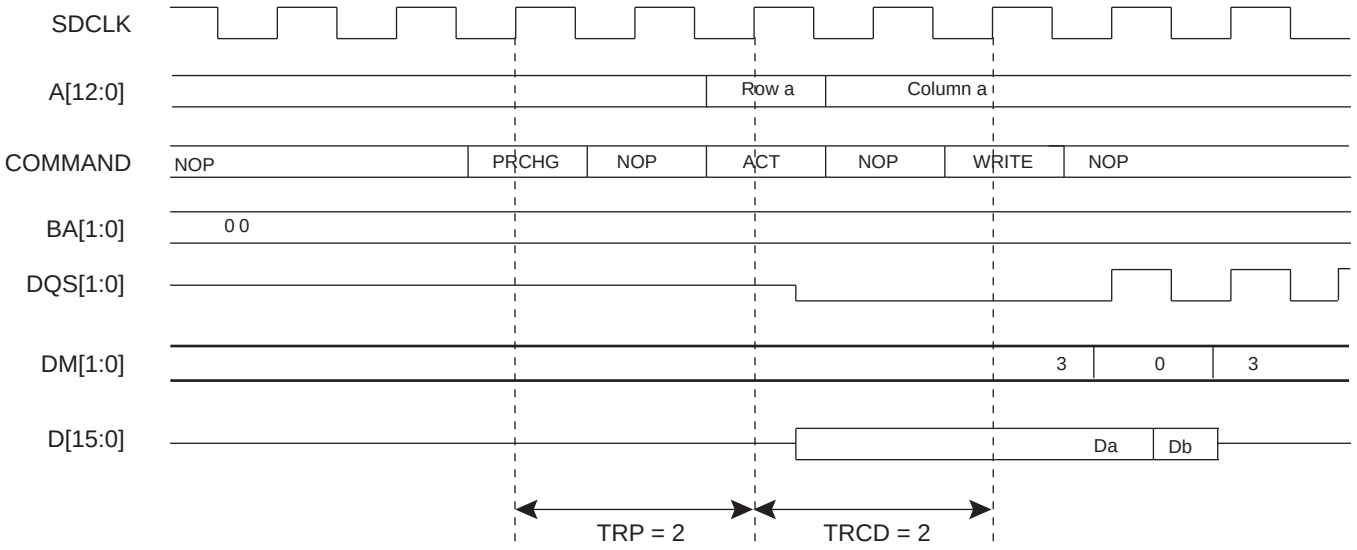
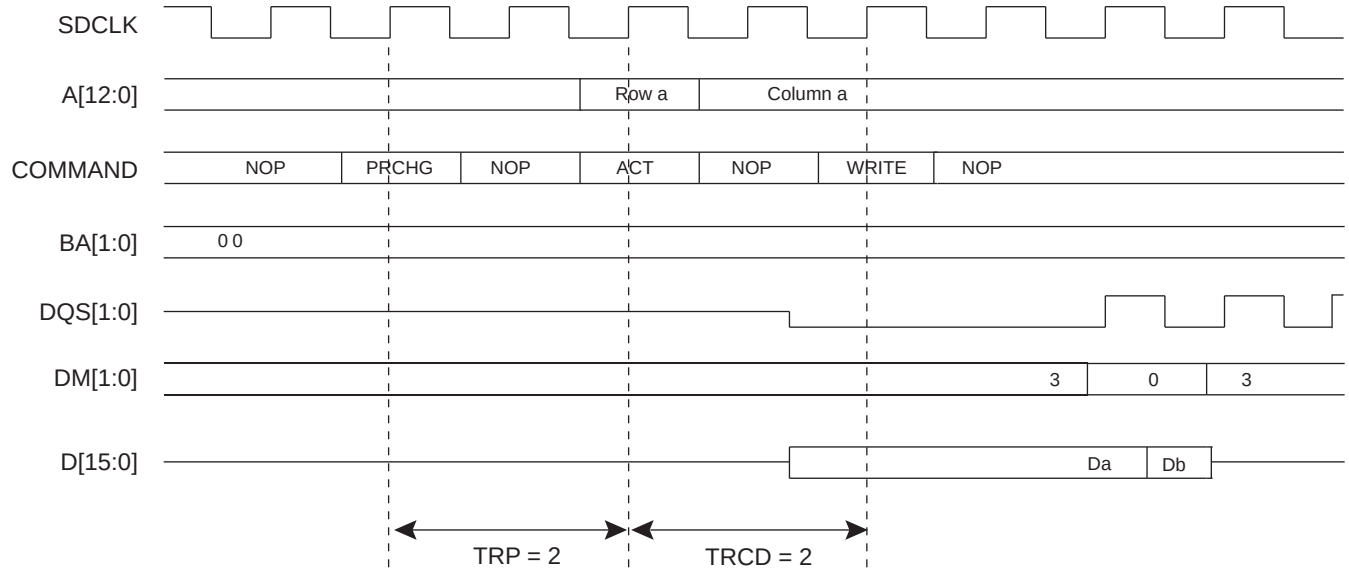


Figure 30-3: Single Write Access, Row Closed, DDR2-SDRAM Device



© 2017 Microchip Technology Inc.

Figure 30-4: Single Write Access, Row Closed, SDR-SDRAM Device

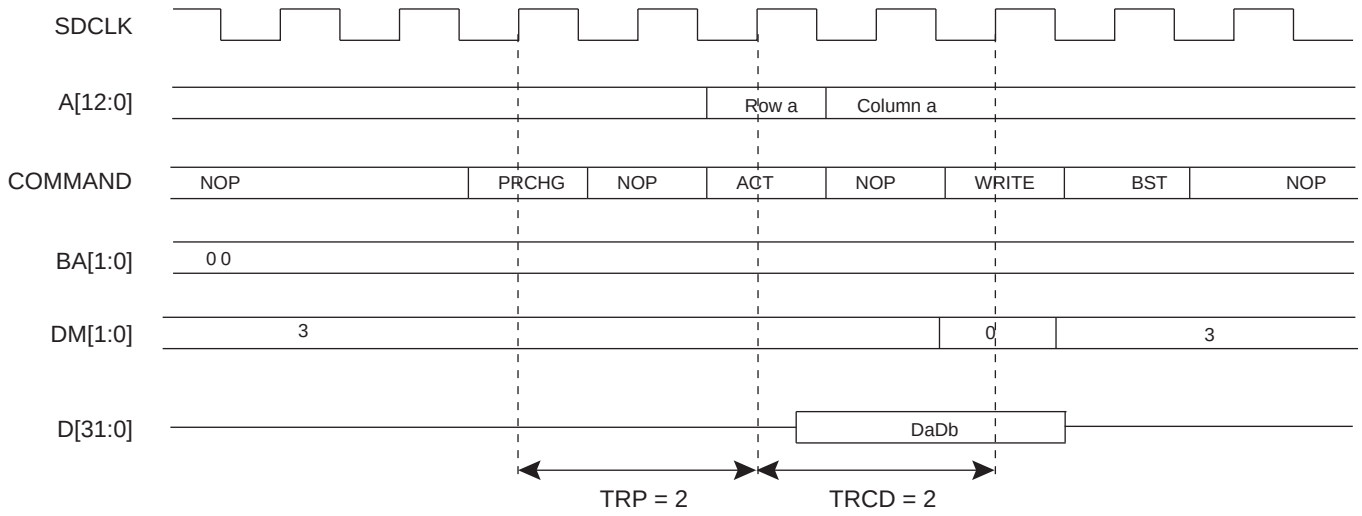


Figure 30-5: Burst Write Access, Row Closed, Low-power DDR1-SDRAM Device

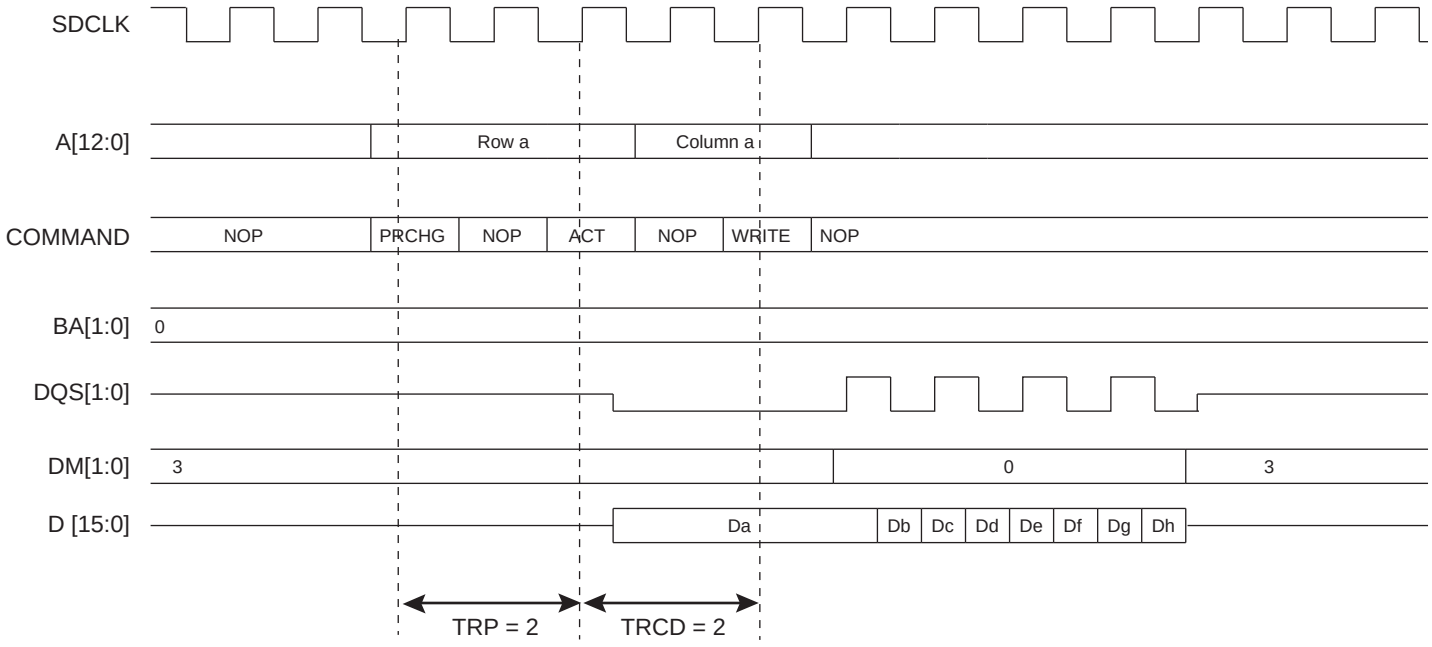


Figure 30-6: Burst Write Access, Row Closed, DDR2-SDRAM Device

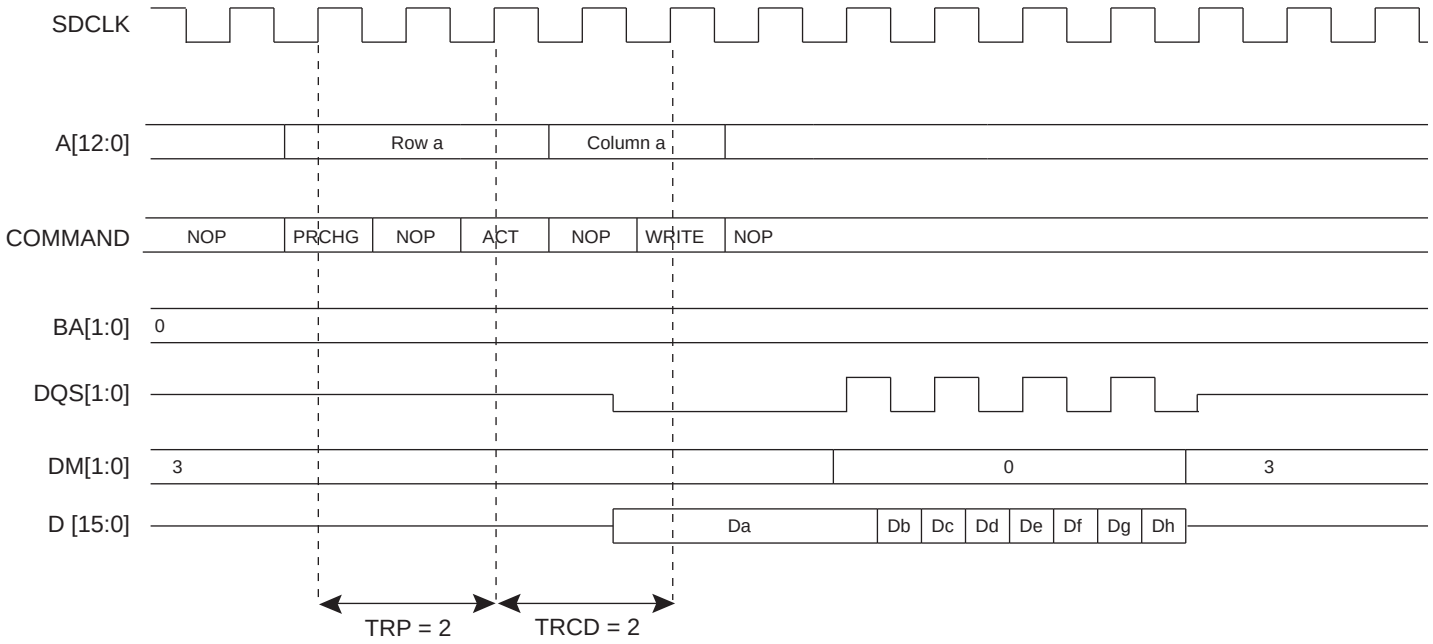
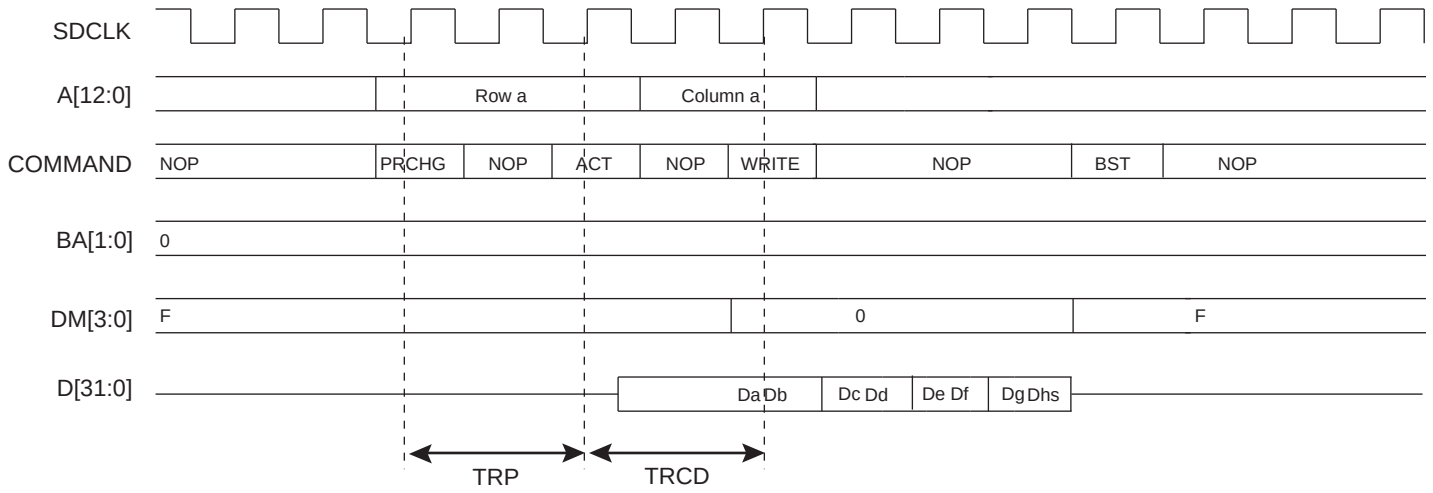


Figure 30-7: Burst Write Access, Row Closed, SDR-SDRAM Device

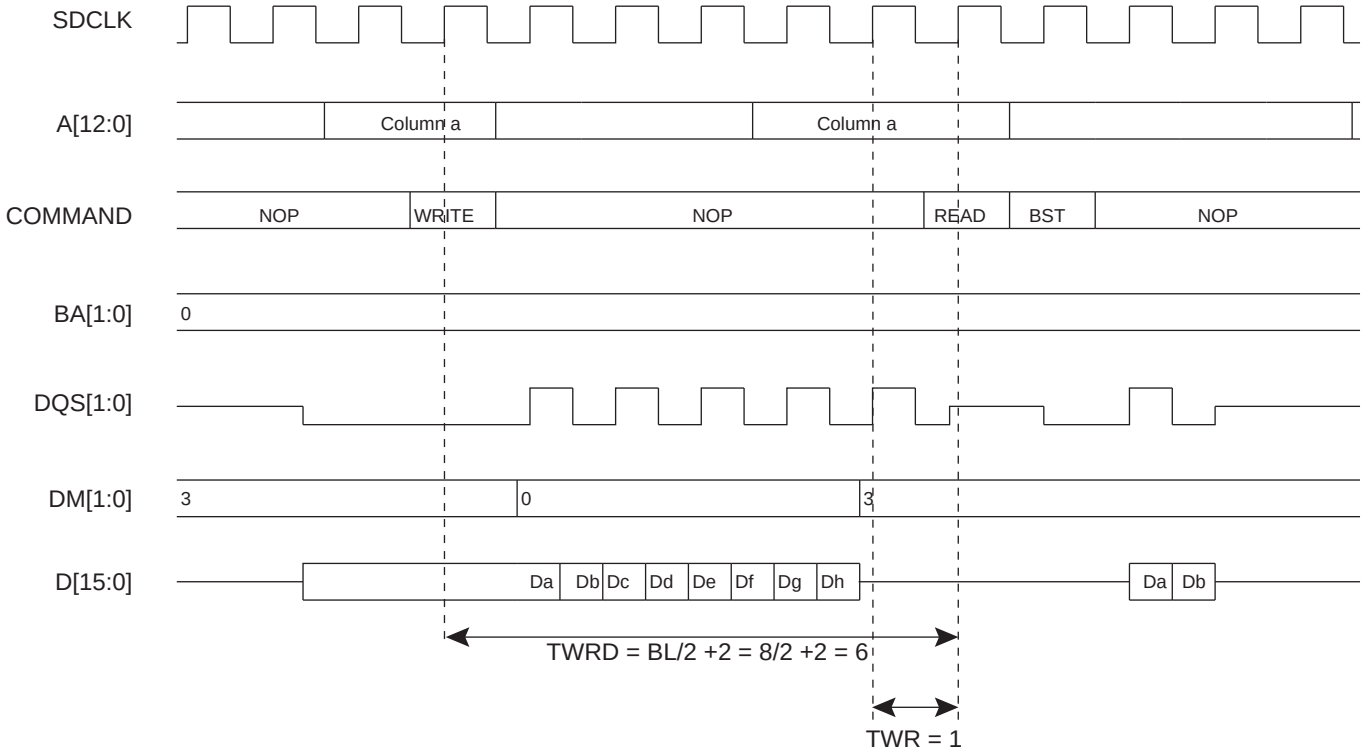


A write command can be followed by a read command. To avoid breaking the current write burst, TWTR/TWRD (BL/2 + 2 = 6 cycles) should be met. See Figure 30-8.

Note: TWRD = Write to read command delay (to input all data)

© 2017 Microchip Technology Inc.

Figure 30-8: Write Command Followed By a Read Command Without Burst Write Interrupt, Low-power DDR1-SDRAM Device



In the case of a single write access, write operation should be interrupted by a read access but DM must be input 1 cycle prior to the read command to avoid writing invalid data. (See Figure 30-9.)

Figure 30-9: Single Write Access Followed By A Read Access Low-power DDR1-SDRAM Devices

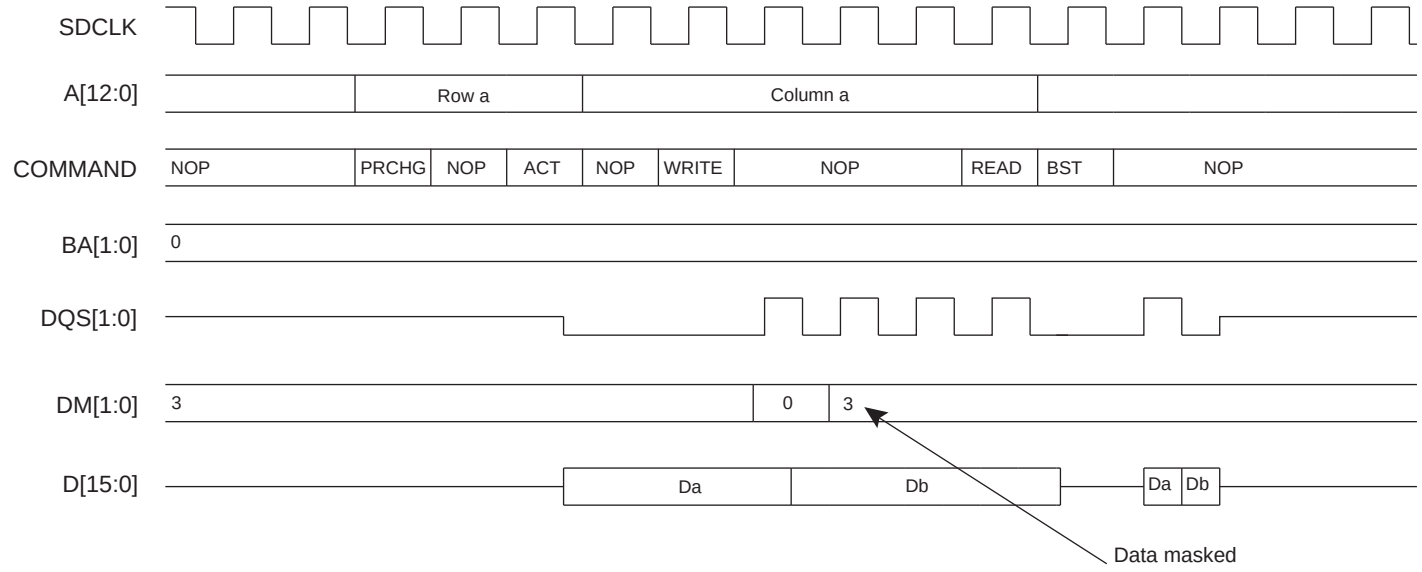
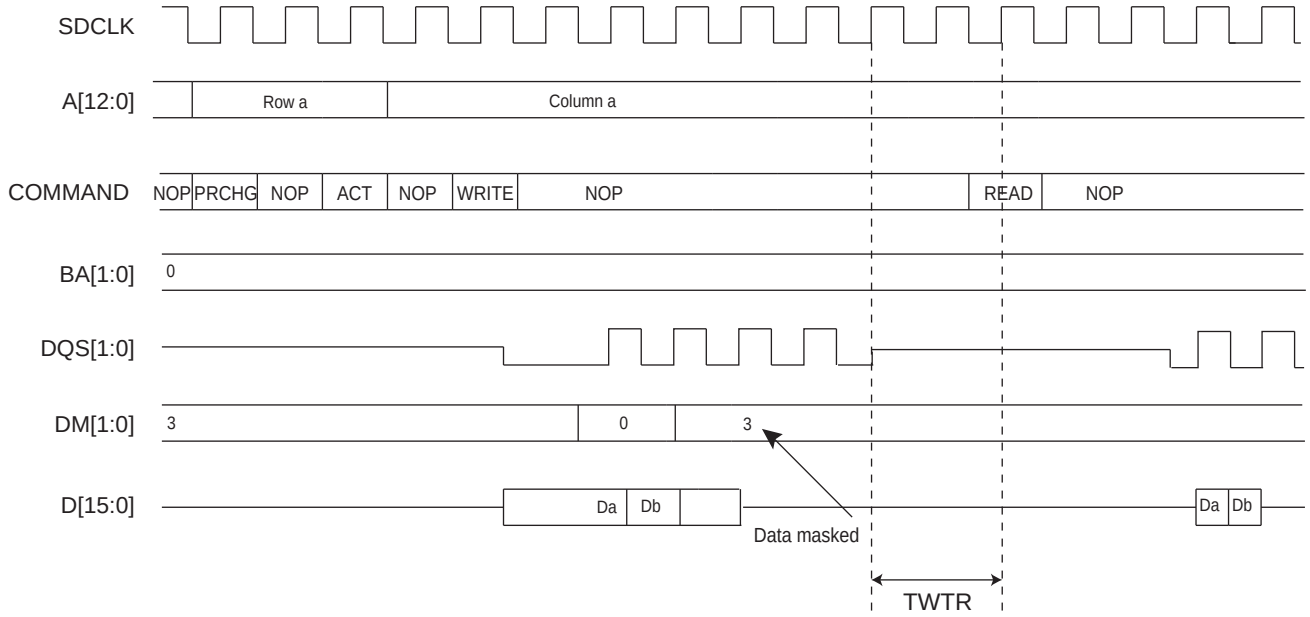


Figure 30-10: Single Write Access Followed By A Read Access, DDR2-SDRAM Device



30.5.2 SDRAM Controller Read Cycle

The DDRSDRC allows burst access or single access in Normal mode (DDRSDRC_MR.MODE = 0). Whatever access type, the DDRSDRC keeps track of the active row in each bank, thus maximizing performance of the DDRSDRC.

The SDRAM devices are programmed with a burst length equal to 8 which determines the length of a sequential data output by the read command that is set to 8. The latency from read command to data output is equal to 2 or 3. This value is programmed during the initialization phase (see Section 30.4.1 "SDR-SDRAM Initialization").

To initiate a single access, the DDRSDRC checks if the page access is already open. If row/bank addresses match with the previous row/bank addresses, the controller generates a read command. If the bank addresses are not identical or if bank addresses are identical but the row addresses are not identical, the controller generates a precharge command, activates the new row and initiates a read command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (TRP) commands and active/read (TRCD) command. After a read command, additional wait states are generated to comply with CAS latency. The DDRSDRC supports a CAS latency of two, two and half, and three (2 or 3 clocks delay). As the burst length is fixed to 8, in the case of single access or burst access inferior to eight data requests, it has to stop the burst otherwise seven or X values could be read. Burst Stop (BST) command is used to stop output during a burst read.

To initiate a burst access, the DDRSDRC checks the transfer type signal. If the next accesses are sequential read accesses, reading to the SDRAM device is carried out. If the next access is a read non-sequential access, then an automatic page break can be inserted. If the bank addresses are not identical or if bank addresses are identical but the row addresses are not identical, the controller generates a precharge command, activates the new row and initiates a read command. In the case where the page access is already open, a read command is generated.

To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (TRP) commands and active/read (TRCD) commands. The DDRSDRC supports a CAS latency of two, two and half, and three (2 or 3 clocks delay). During this delay, the controller uses internal signals to anticipate the next access and improve the performance of the controller. Depending on the latency (2/3), the DDRSDRC anticipates 2 or 3 read accesses. In the case of burst of specified length, accesses are not anticipated, but if the burst is broken (border, busy mode, etc.), the next access is treated as an incrementing burst of unspecified length, and depending on the latency (2/3), the DDRSDRC anticipates 2 or 3 read accesses.

For a definition of timing parameters, refer to Section 30.7.3 "DDRSDRC Configuration Register".

Read accesses to the SDRAM are burst oriented and the burst length is programmed to 8. It determines the maximum number of column locations that can be accessed for a given read command. When the read command is issued, eight columns are selected. All accesses for that burst take place within these eight columns, meaning that the burst wraps within these eight columns if the boundary is reached. These eight columns are selected by addr[13:3]; addr[2:0] is used to select the starting location within the block.

In the case of incrementing burst (INCR/INCR4/INCR8/INCR16), the addresses can cross the 16-byte boundary of the SDRAM device. For example, when a transfer (INCR4) starts at address 0x0C, the next access is 0x10, but since the burst length is programmed to 8, the next access is 0x00. Since the boundary is reached, the burst wraps. The DDRSDRC takes into account this feature of the SDRAM device.

© 2017 Microchip Technology Inc.

In the case of DDR-SDRAM devices, transfers start at address 0x04/0x08/0x0C. In the case of SDR-SDRAM devices, transfers start at address 0x14/0x18/0x1C. Two read commands are issued to avoid wrapping when the boundary is reached. The last read command may generate additional reading (1 read cmd = 4 DDR words or 1 read cmd = 8 SDR words).

To avoid additional reading, it is possible to use the burst stop command to truncate the read burst and to decrease power consumption.

Figure 30-11: Single Read Access, Row Closed, Latency = 2, Low-power DDR1-SDRAM Device

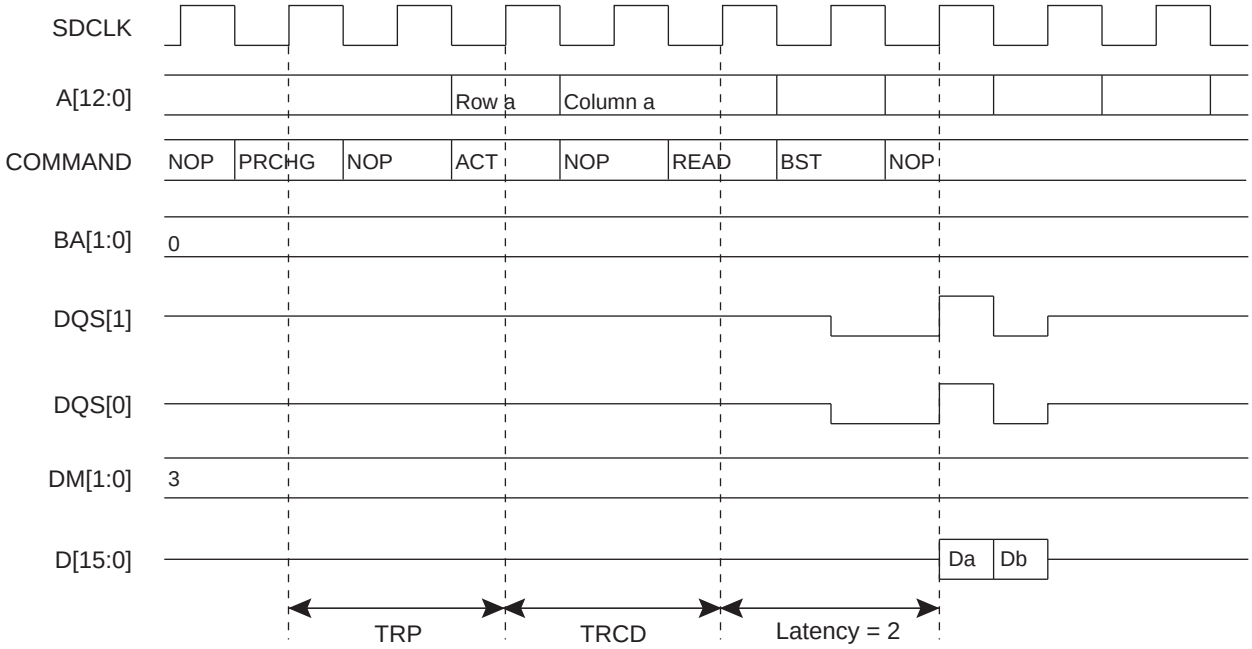


Figure 30-12: Single Read Access, Row Closed, Latency = 3, DDR2-SDRAM Device

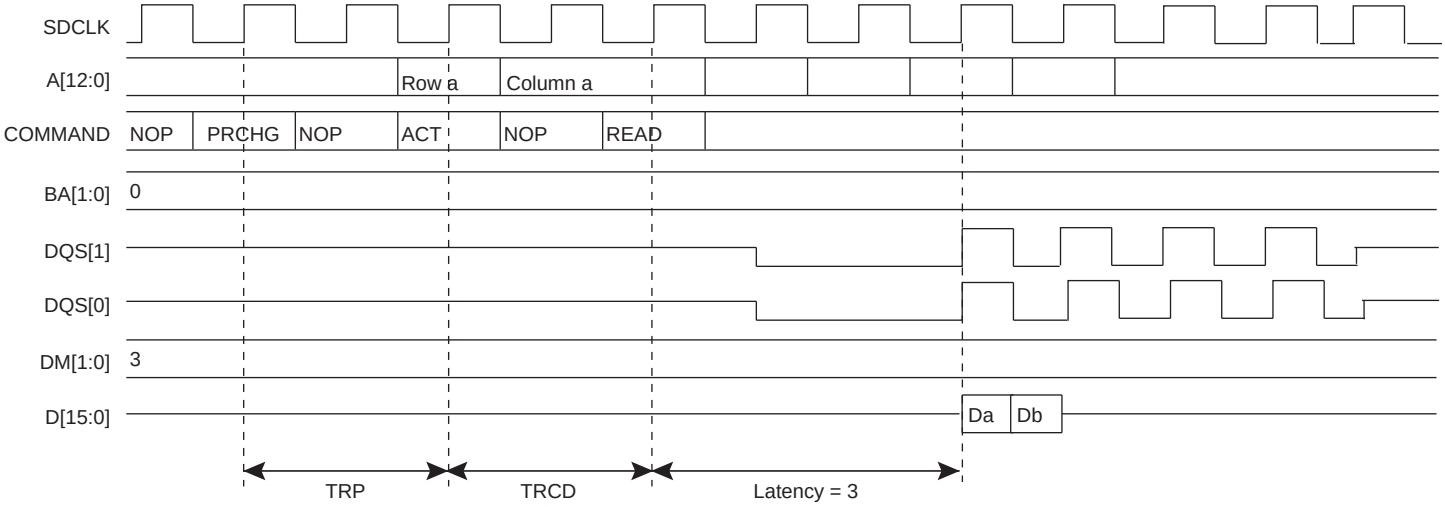


Figure 30-13: Single Read Access, Row Closed, Latency = 2, SDR-SDRAM Device

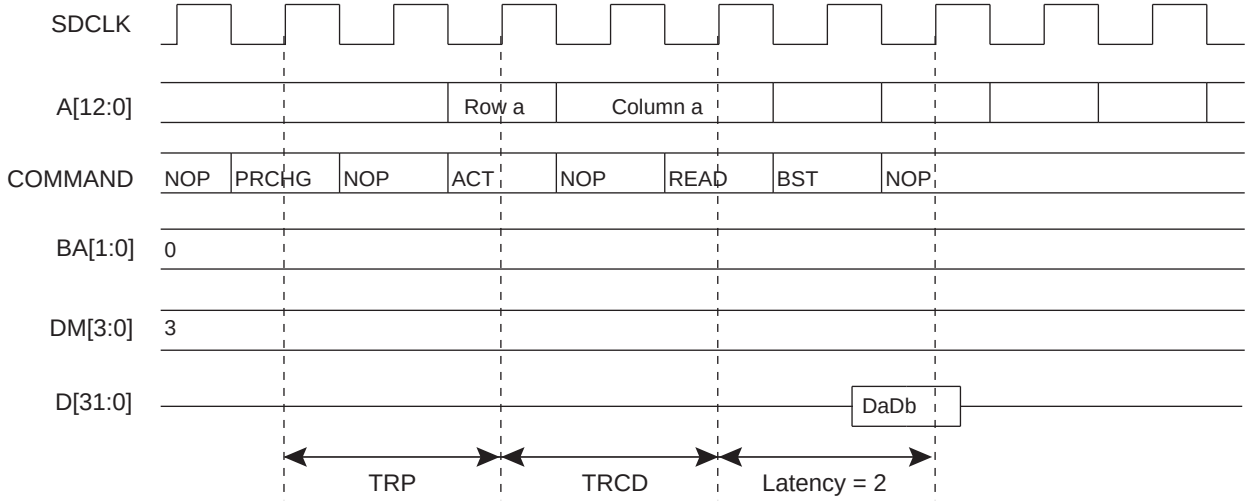
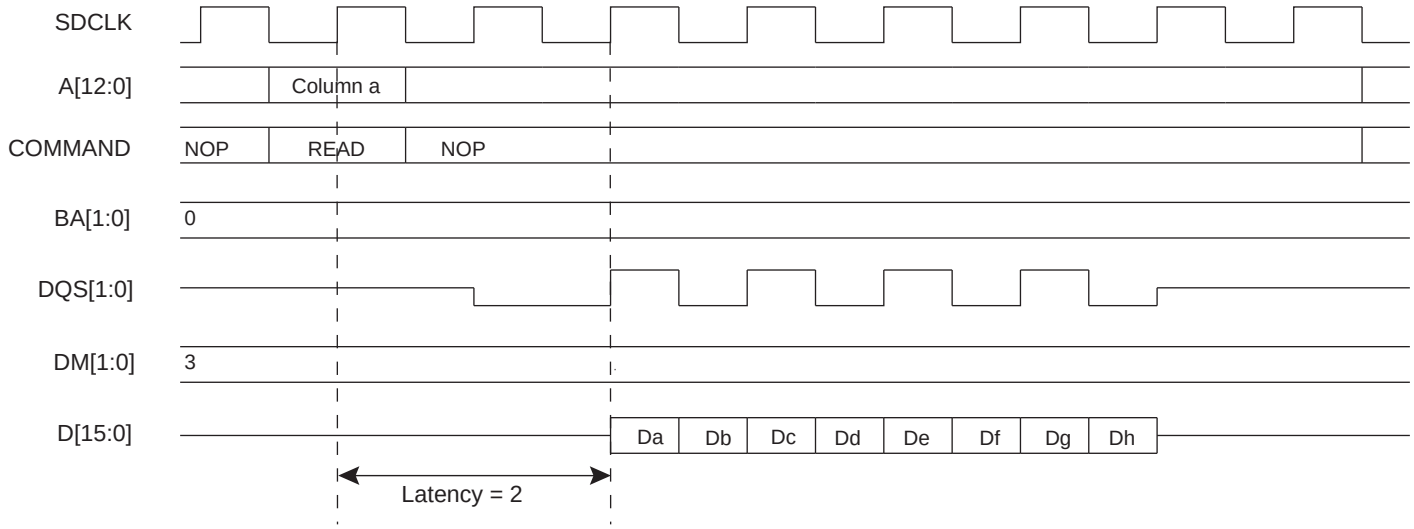


Figure 30-14: Burst Read Access, Latency = 2, Low-power DDR1-SDRAM Devices



© 2017 Microchip Technology Inc.

Figure 30-15: Burst Read Access, Latency = 3, DDR2-SDRAM Devices

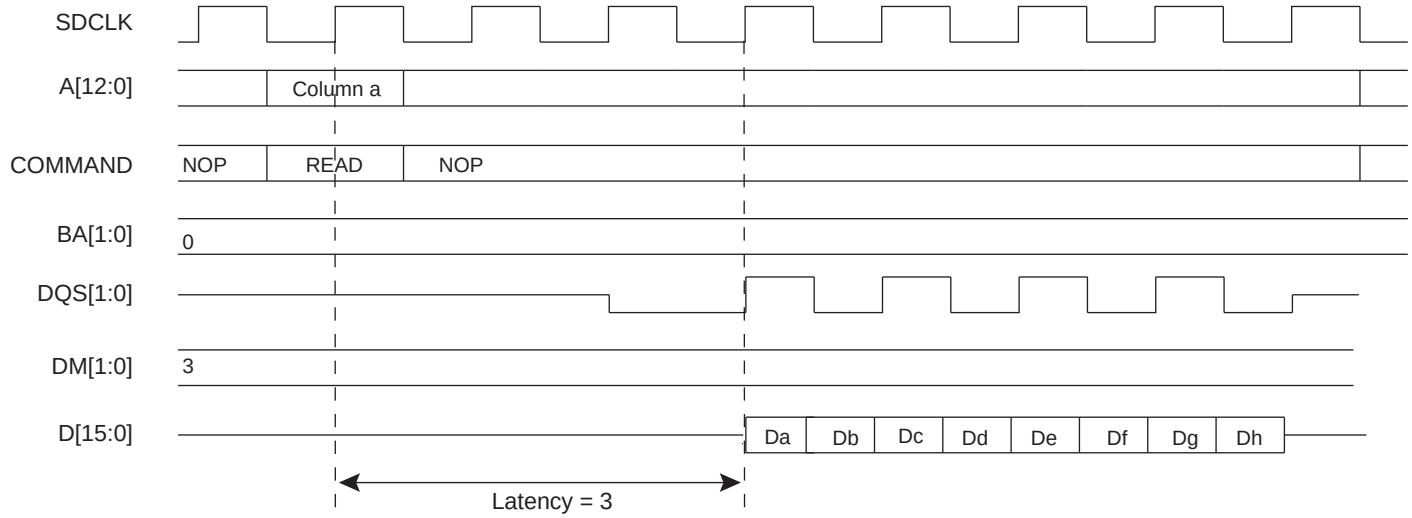
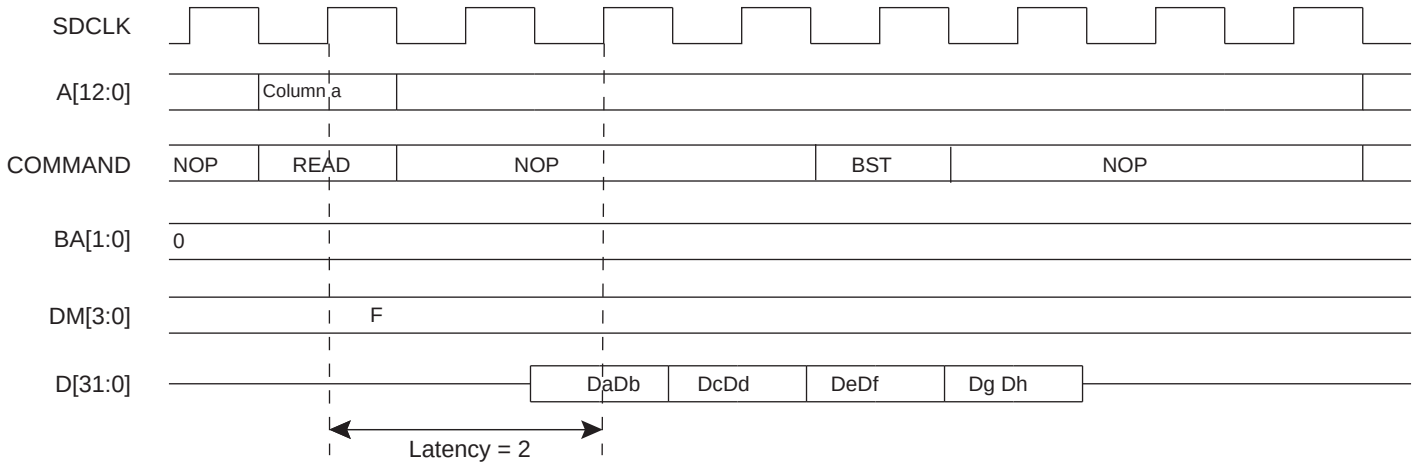


Figure 30-16: Burst Read Access, Latency = 2, SDR-SDRAM Devices



30.5.3 Refresh (Auto-refresh Command)

An auto-refresh command is used to refresh the DDRSDRC. Refresh addresses are generated internally by the SDRAM device and incremented after each auto-refresh automatically. The DDRSDRC generates these auto-refresh commands periodically. A timer is loaded with the value in the register DDRSDRC_TR that indicates the number of clock cycles between refresh cycles. When the DDRSDRC initiates a refresh of an SDRAM device, internal memory accesses are not delayed. However, if the CPU tries to access the SDRAM device, the slave indicates that the device is busy. A request of refresh does not interrupt a burst transfer in progress.

30.5.4 Power Management

30.5.4.1 Self Refresh Mode

This mode is activated by configuring the Low-power Command Bits (LPCB) field value to 1 in the DDRSDRC Low-power Register (DDRSDRC_LPR).

Self Refresh mode is used to reduce power consumption, i.e., when no access to the SDRAM device is possible. In this case, power consumption is very low. In Self Refresh mode, the SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto-refresh cycles. All the inputs to the SDRAM device become “don’t care” except CKE, which remains low. As soon as the SDRAM device is selected, the DDRSDRC provides a sequence of commands and exits Self Refresh mode.

The DDRSDRC re-enables Self Refresh mode as soon as the SDRAM device is not selected. It is possible to define when Self Refresh mode will be enabled by configuring the TIMEOUT command field in the DDRSDRC Low-power Register (DDRSDRC_LPR) (see Section 30.7.7 ”DDRSDRC Low-power Register”):

- 0 = Self Refresh mode is enabled as soon as the SDRAM device is not selected
- 1 = Self Refresh mode is enabled 64 clock cycles after completion of the last access
- 2 = Self Refresh mode is enabled 128 clock cycles after completion of the last access

As soon as the SDRAM device is no longer selected, All Banks Precharge command is generated followed by a Self Refresh command. If, between these two commands an SDRAM access is detected, Self Refresh command will be replaced by an Auto-refresh command. According to the application, more Auto-refresh commands will be performed when the Self Refresh mode is enabled during the application.

This controller also interfaces low-power SDRAM. Compared to standard SDRAM, these devices add a feature: A single quarter, one half quarter or all banks of the SDRAM array can be enabled in Self Refresh mode. Disabled banks will be not refreshed in Self Refresh mode. This feature permits to reduce the Self Refresh current. The Extended Mode Register controls this feature, it includes Partial Array Self Refresh (PASR) parameters and Drive Strength (DS). These parameters are set during the initialization phase. After initialization, as soon as PASR/DS fields are modified, the Extended Mode Register in the memory of the external device is accessed automatically and PASR/DS fields are updated before entry into Self Refresh mode if DDRSDRC does not share an external bus with another controller or during a refresh command, and a pending read or write access, if DDRSDRC does share an external bus with another controller. The type of update is determined by the value of the DDRSDRC_LPR.UPD_MR field.

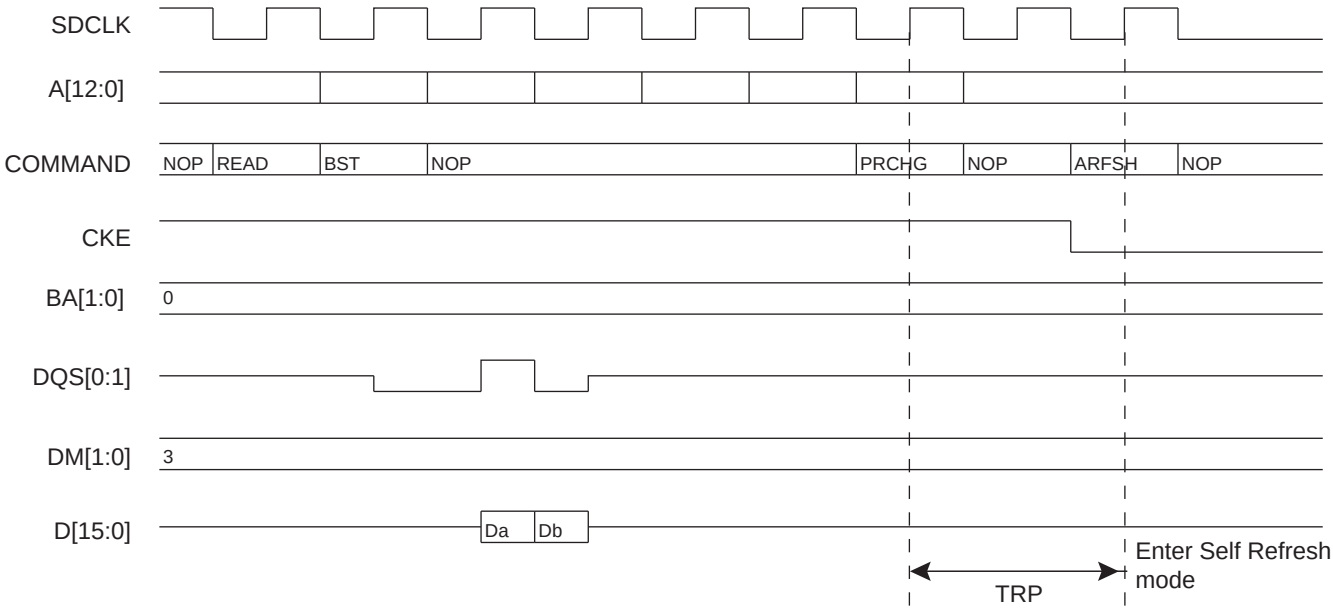
The low-power SDR-SDRAM must remain in Self Refresh mode for a minimum period of TRAS periods and may remain in Self Refresh mode for an indefinite period. (See Figure 30-17.)

The low-power DDR1-SDRAM must remain in Self Refresh mode for a minimum of TRFC periods and may remain in Self Refresh mode for an indefinite period.

The DDR2-SDRAM must remain in Self Refresh mode for a minimum of TCKE periods and may remain in Self Refresh mode for an indefinite period.

Note: Some SDRAM providers impose that 4K cycles of burst auto-refresh are required before Self Refresh entry and immediately after Self Refresh exit. This constraint is not supported.

Figure 30-17: Self Refresh Mode Entry, DDRSDRC_LPR.TIMEOUT = 0



© 2017 Microchip Technology Inc.

Figure 30-18: Self Refresh Mode Entry, DDRSDRC_LPR.TIMEOUT = 1 or 2

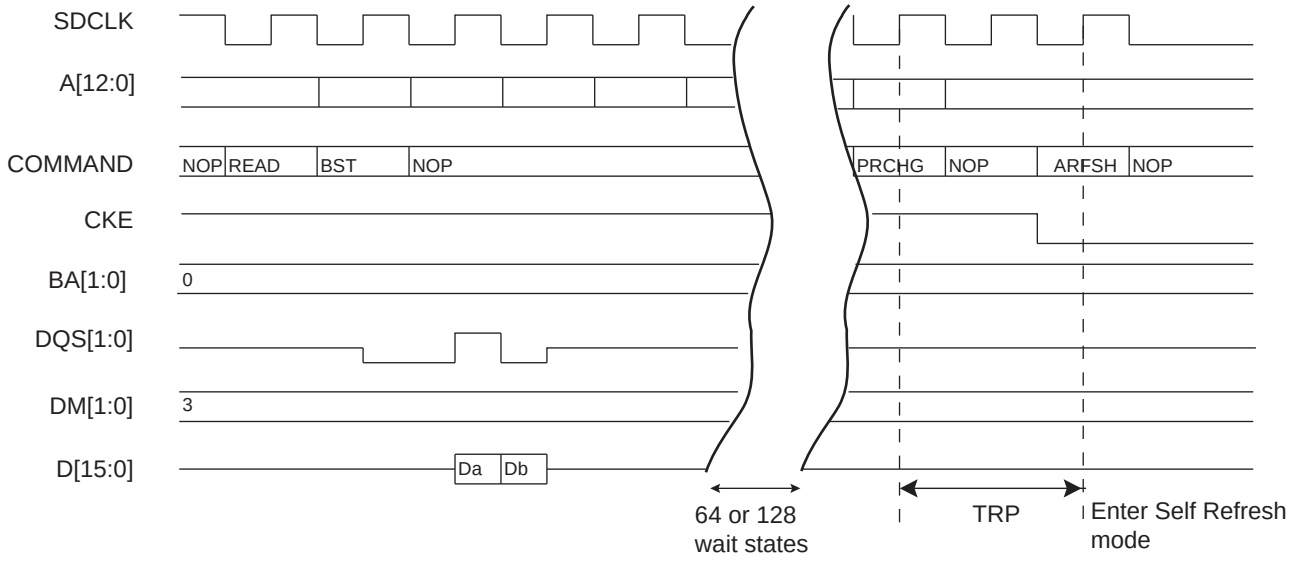


Figure 30-19: Self Refresh Mode Exit

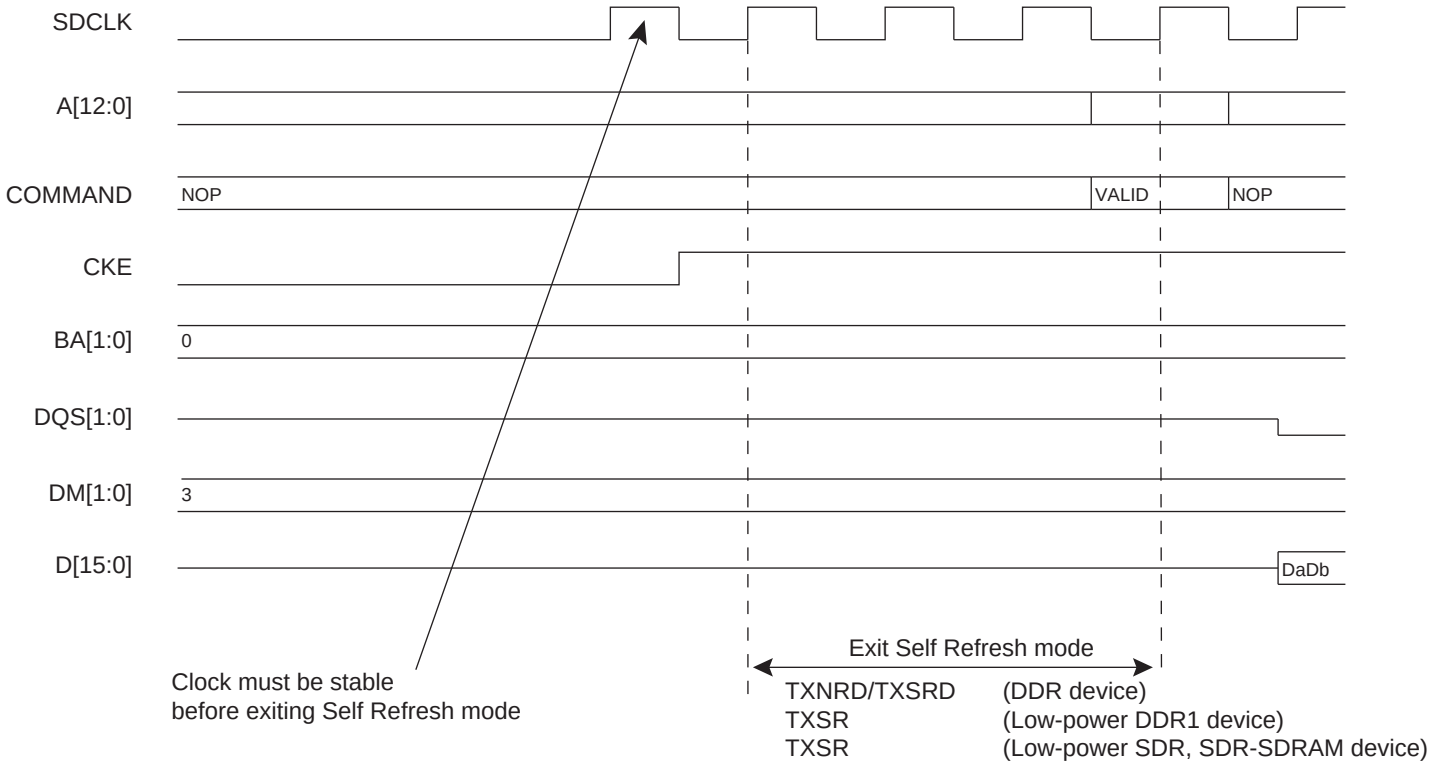


Figure 30-20: Self Refresh and Automatic Update

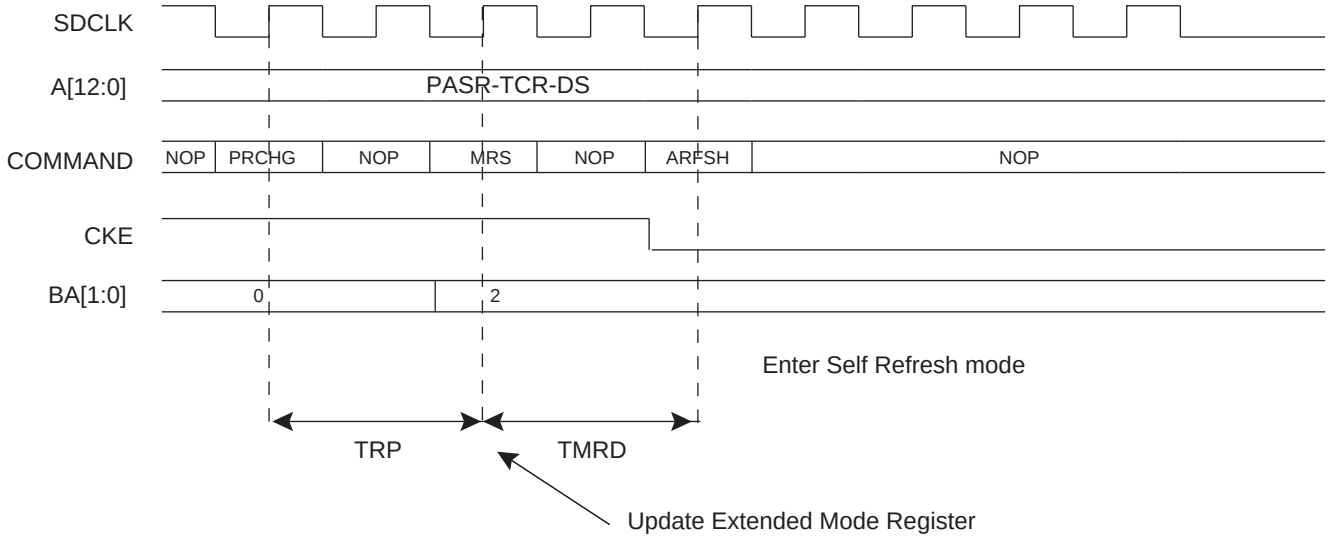
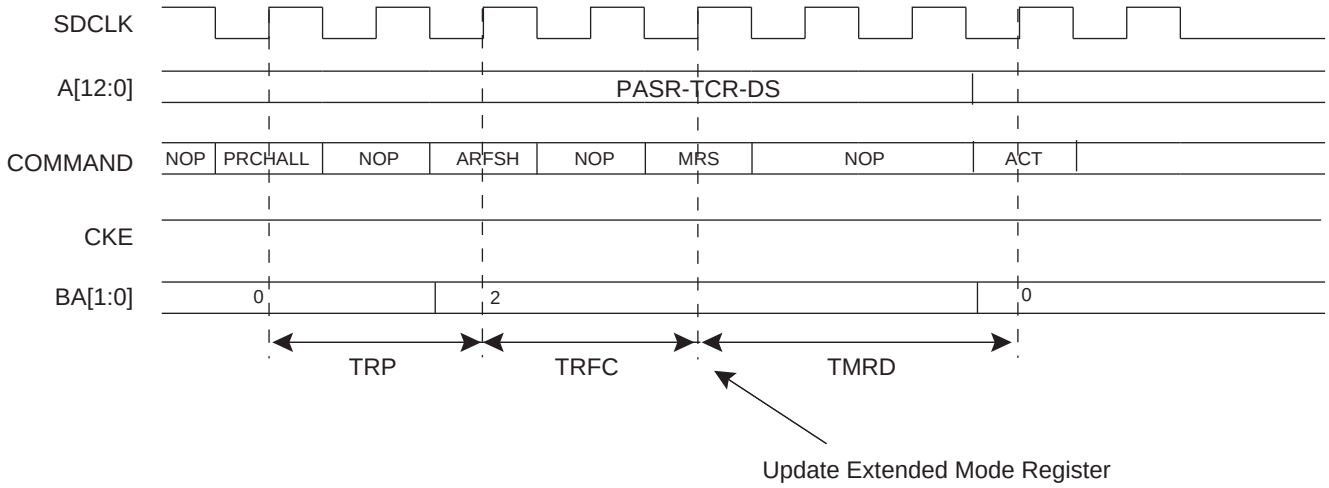


Figure 30-21: Automatic Update During Auto-refresh Command and SDRAM Access



30.5.4.2 Power-down Mode

This mode is activated by configuring the Low-power Command Bits (LPCB) field value to 2 in the DDRSDRC_LPR.

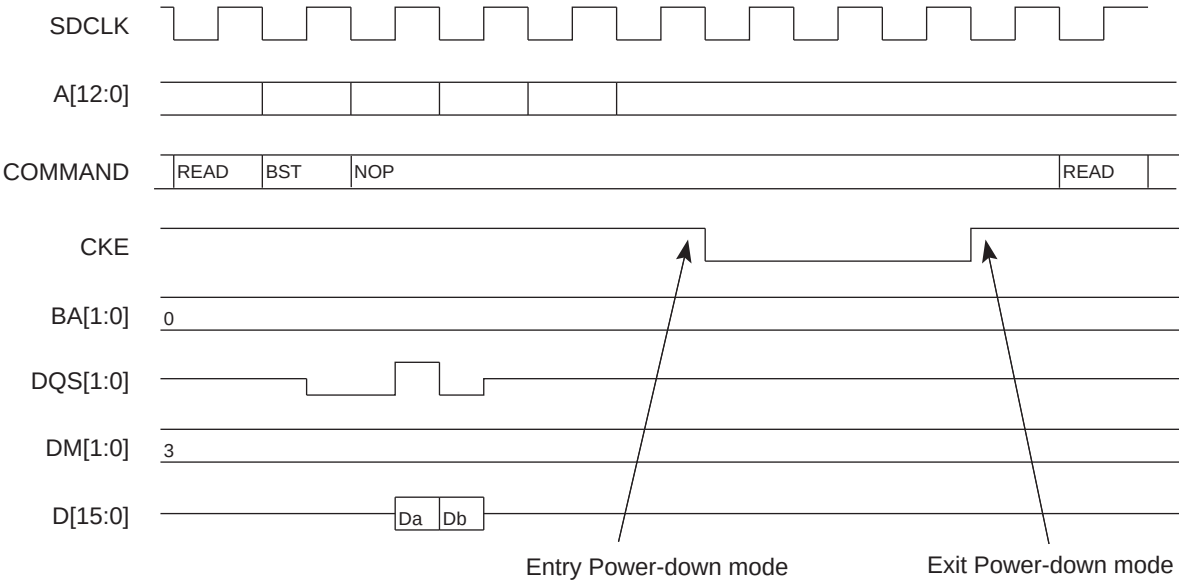
Power-down mode is used when no access to the SDRAM device is possible. In this mode, power consumption is greater than in Self Refresh mode. This state is similar to Normal mode (No low-power mode/No Self Refresh mode), but the CKE pin is low and the input and output buffers are deactivated as soon the SDRAM device is no longer accessible. In contrast to Self Refresh mode, the SDRAM device cannot remain in low-power mode longer than the refresh period (64 ms). As no auto-refresh operations are performed in this mode, the DDRSDRC carries out the refresh operation. In order to exit low-power mode, a NOP command is required in the case of Low-power SDR-SDRAM and SDR-SDRAM devices. In the case of Low-power DDR1-SDRAM devices, the controller generates a NOP command during a delay of at least TXP. In addition, Low-power DDR1-SDRAM and DDR2-SDRAM must remain in Power-down mode for a minimum period of TCKE periods.

The exit procedure is faster than in Self Refresh mode. (See Figure 30-22.) The DDRSDRC returns to Power-down mode as soon as the SDRAM device is not selected. It is possible to define when Power-down mode is enabled by configuring the TIMEOUT command field in the DDRSDRC Low-power Register (DDRSDRC_LPR) (see Section 30.7.7 "DDRSDRC Low-power Register"):

- 0 = Power-down mode is enabled as soon as the SDRAM device is not selected
- 1 = Power-down mode is enabled 64 clock cycles after completion of the last access
- 2 = Power-down mode is enabled 128 clock cycles after completion of the last access

© 2017 Microchip Technology Inc.

Figure 30-22: Power-down Entry/Exit, DDRSDRC_LPR.TIMEOUT = 0

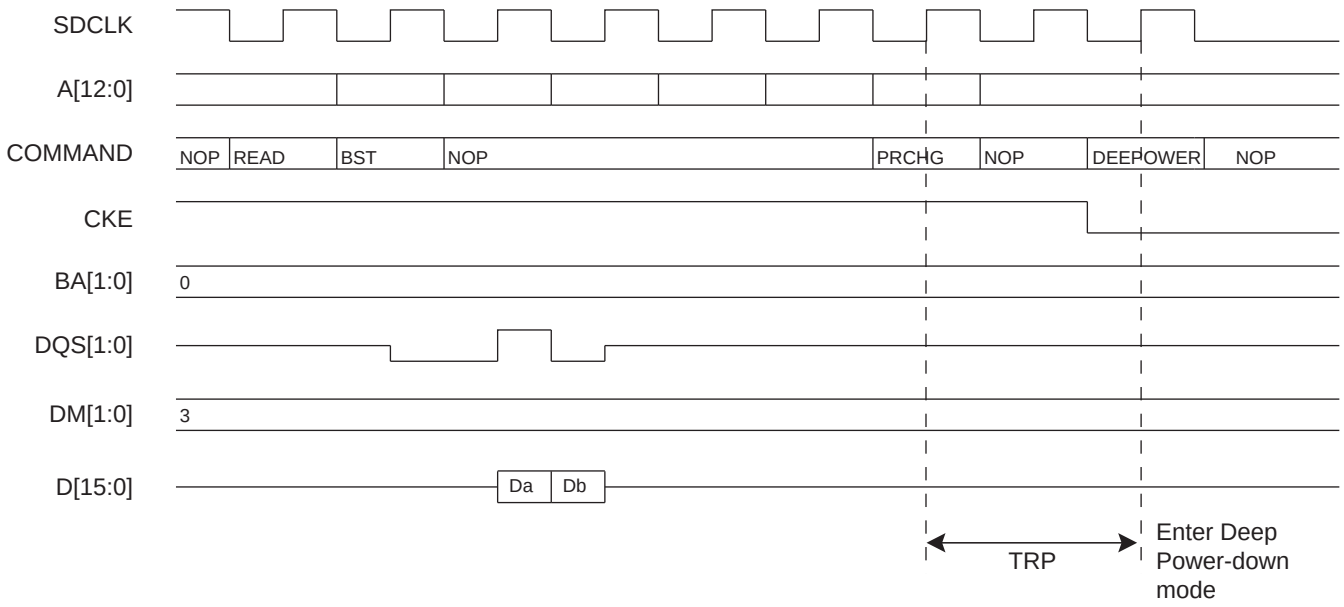


30.5.4.3 Deep Power-down Mode

The Deep Power-down mode is a feature of the Low-power SDRAM. When this mode is activated, all internal voltage generators inside the device are stopped and all data is lost.

This mode is activated by configuring the Low-power Command Bits (LPCB) field value to 3 in the DDRSDRC_LPR. When this mode is enabled, the DDRSDRC leaves Normal mode (DDRSDRC_MR.MODE = 0) and the controller is frozen. To exit Deep Power-down mode, DDRSDRC_LPR.LPCB must be configured to 0 and an initialization sequence must be generated by software. See Section 30.4.2 "Low-power DDR1-SDRAM Initialization".

Figure 30-23: Deep Power-down Mode Entry



30.5.4.4 Reset Mode

The reset mode is a feature of the DDR2-SDRAM. This mode is activated by configuring the Low-power Command Bits (LPCB) field value to 3 and the Clock Frozen Command Bit (CLK_FR) to 1 in the DDRSDRC_LPR.

When this mode is enabled, the DDRSDRC leaves Normal mode (DDRSDRC_MR.MODE = 0) and the controller is frozen. Before enabling this mode, the end user must assume there is not an access in progress.

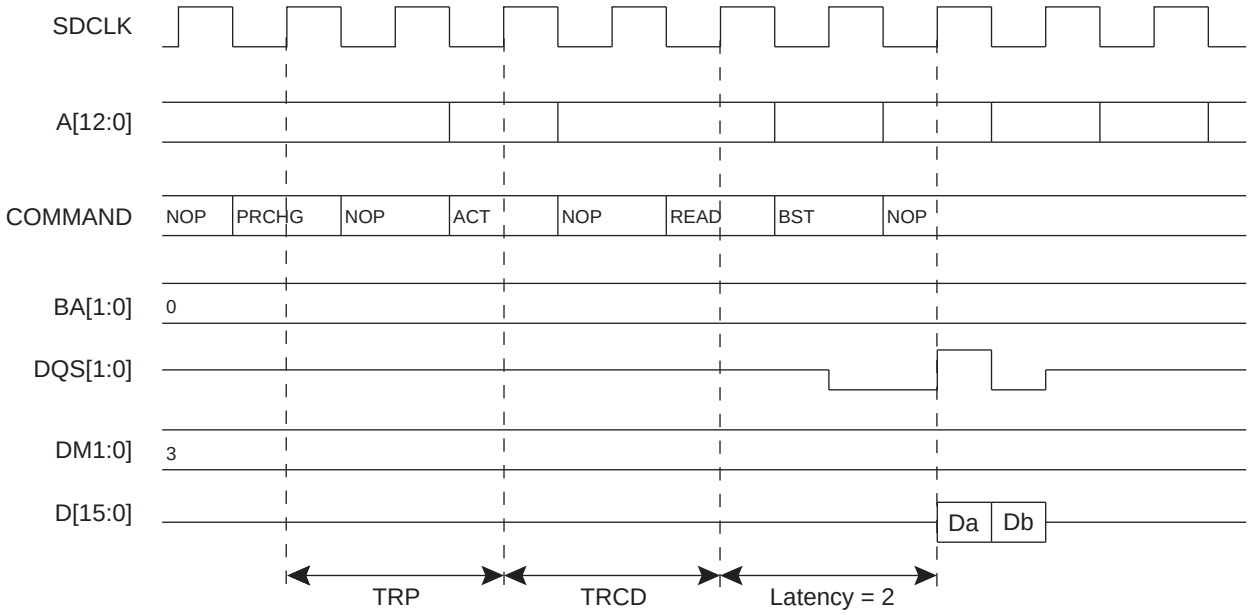
To exit reset mode, DDRSDRC_LPR.LPCB must be configured to 0, DDRSDRC_LPR.CLK_FR to 0, and an initialization sequence must be generated by software. See Section 30.4.3 "DDR2-SDRAM Initialization".

30.5.5 Multi-port Functionality

The SDRAM protocol imposes a check of timings prior to performing a read or a write access, thus decreasing the performance of systems. An access to SDRAM is performed if banks and rows are open (or active). To activate a row in a particular bank, it has to de-active the last open row and open the new row. Two SDRAM commands must be performed to open a bank: Precharge and Active command with respect to TRP timing. Before performing a read or write command, TRCD timing must be checked.

This operation represents a significative loss of performance. (See Figure 30-24.)

Figure 30-24: TRP and TRCD Timings



4 cycles before performing a read command

The multi-port controller has been designed to mask these timings and thus improve the bandwidth of the system.

DDRSDRC is a multi-port controller since four masters can simultaneously reach the controller. This feature improves the bandwidth of the system because it can detect four requests on the AHB slave inputs and thus anticipate the commands that follow, PRECHARGE and ACTIVE commands in bank X during current access in bank Y. This allows TRP and TRCD timings to be masked (see Figure 30-25). In the best case, all accesses are done as if the banks and rows were already open. The best condition is met when the four masters work in different banks. In the case of four simultaneous read accesses, when the four banks and associated rows are open, the controller reads with a continuous flow and masks the CAS latency for each different access. To allow a continuous flow, the read command must be set at 2 or 3 cycles (CAS latency) before the end of current access. This requires that the scheme of arbitration changes since the round-robin arbitration cannot be respected. If the controller anticipates a read access, and thus before the end of current access a master with a high priority arises, then this master will not serviced.

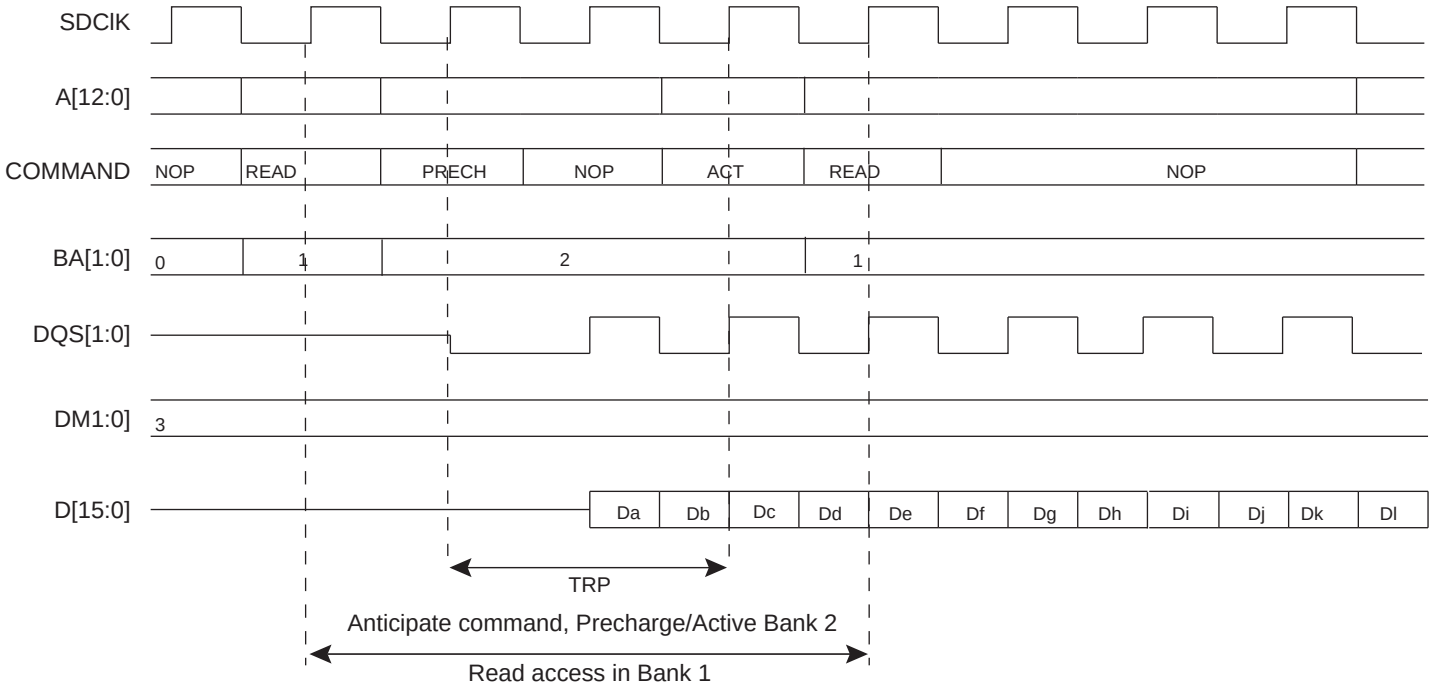
The arbitration mechanism reduces latency when conflicts occur, i.e., when two or more masters try to access the SDRAM device at the same time.

© 2017 Microchip Technology Inc.

The arbitration type is round-robin arbitration. This algorithm dispatches the requests from different masters to the SDRAM device in a round-robin manner. If two or more master requests arise at the same time, the master with the lowest number is serviced first, then the others are serviced in a round-robin manner. To avoid burst breaking and to provide the maximum throughput for the SDRAM device, arbitration may only take place during the following cycles:

1. Idle cycles: When no master is connected to the SDRAM device.
2. Single cycles: When a slave is currently doing a single access.
3. End of Burst cycles: When the current cycle is the last cycle of a burst transfer. For bursts of defined length, predicted end of burst matches the size of the transfer. For bursts of undefined length, predicted end of burst is generated at the end of each four beat boundary inside the INCR transfer.
4. Anticipated Access: When an anticipate read access is performed while current access is not complete, the arbitration scheme can be changed if the anticipated access is not the next access serviced by the arbitration scheme.

Figure 30-25: Anticipate Precharge/Active Command in Bank 2 During Read Access in Bank 1



30.5.6 Register Write Protection

To prevent any single software error from corrupting DDRSDRC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the DDRSDRC Write Protection Mode Register (DDRSDRC_WPMR).

If a write access in a write-protected register is detected, the WPVS flag in the DDRSDRC Write Protection Status Register (DDRSDRC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS flag is automatically cleared after reading the DDRSDRC_WPSR.

The following registers can be write-protected:

- DDRSDRC Mode Register
- DDRSDRC Refresh Timer Register
- DDRSDRC Configuration Register
- DDRSDRC Timing Parameter 0 Register
- DDRSDRC Timing Parameter 1 Register
- DDRSDRC Timing Parameter 2 Register
- DDRSDRC Memory Device Register
- DDRSDRC High Speed Register

30.6 Software Interface/SDRAM Organization, Address Mapping

The SDRAM address space is organized into banks, rows and columns. The DDRSDRC maps different memory types depending on the values set in the DDRSDRC Configuration Register (DDRSDRC_CR). The following tables illustrate the relation between CPU addresses and columns, rows and banks addresses for 16-bit memory data bus widths and 32-bit memory data bus widths.

The DDRSDRC supports address mapping in linear mode and interleaved mode.

Linear mode is a method for address mapping where banks alternate at each last SDRAM page of current bank.

Interleaved mode is a method for address mapping where banks alternate at each SDRAM end page of current bank.

The DDRSDRC makes the SDRAM devices access protocol transparent to the user. Table 30-1 to Table 30-15 illustrate the SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

30.6.1 SDRAM Address Mapping for 16-bit Memory Data Bus Width and Four Banks

Table 30-1: Linear Mapping for SDRAM Configuration, 2K Rows, 512/1024/2048/4096 Columns

CPU Address Line																										
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Bk[1:0]		Row[10:0]											Column[8:0]									M0
				Bk[1:0]			Row[10:0]											Column[9:0]									M0
			Bk[1:0]				Row[10:0]											Column[10:0]									M0
		Bk[1:0]					Row[10:0]											Column[11:0]									M0

Table 30-2: Linear Mapping for SDRAM Configuration: 4K Rows, 512/1024/2048/4096 Columns

CPU Address Line																										
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Bk[1:0]			Row[11:0]											Column[8:0]									M0
			Bk[1:0]				Row[11:0]											Column[9:0]									M0
		Bk[1:0]					Row[11:0]											Column[10:0]									M0
	Bk[1:0]						Row[11:0]											Column[11:0]									M0

© 2017 Microchip Technology Inc.

Table 30-3: Linear Mapping for SDRAM Configuration: 8K Rows, 512/1024/2048/4096 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
			Bk[1:0]		Row[12:0]													Column[8:0]									M0
		Bk[1:0]			Row[12:0]													Column[9:0]									M0
	Bk[1:0]				Row[12:0]													Column[10:0]									M0
Bk[1:0]					Row[12:0]													Column[11:0]									M0

Table 30-4: Linear Mapping for SDRAM Configuration: 16K Rows, 512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
		Bk[1:0]			Row[13:0]													Column[8:0]									M0
	Bk[1:0]				Row[13:0]													Column[9:0]									M0
Bk[1:0]					Row[13:0]													Column[10:0]									M0

Table 30-5: Interleaved Mapping for SDRAM Configuration, 2K Rows, 512/1024/2048/4096 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
					Row[10:0]											Bk[1:0]		Column[8:0]									M0
					Row[10:0]										Bk[1:0]			Column[9:0]									M0
					Row[10:0]									Bk[1:0]				Column[10:0]									M0
					Row[10:0]								Bk[1:0]					Column[11:0]									M0

Table 30-6: Interleaved Mapping for SDRAM Configuration: 4K Rows, 512/1024/2048/4096 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				Row[11:0]												Bk[1:0]		Column[8:0]									M0
			Row[11:0]												Bk[1:0]			Column[9:0]									M0
		Row[11:0]												Bk[1:0]				Column[10:0]									M0
	Row[11:0]												Bk[1:0]					Column[11:0]									M0

Table 30-7: Interleaved Mapping for SDRAM Configuration: 8K Rows, 512/1024/2048/4096 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			Row[12:0]													Bk[1:0]		Column[8:0]									M0
		Row[12:0]													Bk[1:0]		Column[9:0]										M0
	Row[12:0]													Bk[1:0]		Column[10:0]											M0
Row[12:0]													Bk[1:0]		Column[11:0]												M0

Table 30-8: Interleaved Mapping for SDRAM Configuration: 16K Rows, 512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Row[13:0]														Bk[1:0]		Column[8:0]									M0
	Row[13:0]														Bk[1:0]		Column[9:0]										M0
Row[13:0]														Bk[1:0]		Column[10:0]											M0

30.6.2 SDRAM Address Mapping for 16-bit Memory Data Bus Width and Eight Banks

Table 30-9: Linear Mapping for SDRAM Configuration: 8K Rows, 1024 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Bk[2:0]			Row[12:0]													Column[9:0]										M0

Table 30-10: Linear Mapping for SDRAM Configuration: 16K Rows, 1024 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bk[2:0]			Row[13:0]														Column[9:0]										M0

Table 30-11: Interleaved Mapping for SDRAM Configuration: 8K Rows, 1024 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Row[12:0]													Bk[2:0]			Column[9:0]										M0

Table 30-12: Interleaved Mapping for SDRAM Configuration: 16K Rows, 1024 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row[12:0]														Bk[2:0]			Column[9:0]										M0

© 2017 Microchip Technology Inc.

30.6.3 SDR-SDRAM Address Mapping for 32-bit Memory Data Bus Width

Table 30-13: SDR-SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
					Bk[1:0]		Row[10:0]											Column[7:0]									M[1:0]	
				Bk[1:0]			Row[10:0]											Column[8:0]									M[1:0]	
			Bk[1:0]				Row[10:0]											Column[9:0]									M[1:0]	
		Bk[1:0]					Row[10:0]											Column[10:0]									M[1:0]	

Table 30-14: SDR-SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				Bk[1:0]			Row[11:0]											Column[7:0]									M[1:0]	
			Bk[1:0]				Row[11:0]											Column[8:0]									M[1:0]	
		Bk[1:0]					Row[11:0]											Column[9:0]									M[1:0]	
	Bk[1:0]						Row[11:0]											Column[10:0]									M[1:0]	

Table 30-15: SDR-SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
			Bk[1:0]				Row[12:0]											Column[7:0]									M[1:0]	
		Bk[1:0]					Row[12:0]											Column[8:0]									M[1:0]	
	Bk[1:0]						Row[12:0]											Column[9:0]									M[1:0]	
Bk[1:0]							Row[12:0]											Column[10:0]									M[1:0]	

Note 1: M[1:0] is the byte address inside a 32-bit word.

2: Bk[2] = BA2, Bk[1] = BA1, Bk[0] = BA0

30.7 DDR SDR SDRAM Controller (DDRSDRC) User Interface

The User Interface is connected to the APB bus. The DDRSDRC is programmed using the registers listed in Table 30-16.

Table 30-16: Register Mapping

Offset	Register	Name	Access	Reset
0x00	DDRSDRC Mode Register	DDRSDRC_MR	Read/Write	0x00000000
0x04	DDRSDRC Refresh Timer Register	DDRSDRC_RTR	Read/Write	0x00000000
0x08	DDRSDRC Configuration Register	DDRSDRC_CR	Read/Write	0x7024
0x0C	DDRSDRC Timing Parameter 0 Register	DDRSDRC_TPR0	Read/Write	0x20227225
0x10	DDRSDRC Timing Parameter 1 Register	DDRSDRC_TPR1	Read/Write	0x3c80808
0x14	DDRSDRC Timing Parameter 2 Register	DDRSDRC_TPR2	Read/Write	0x2062
0x18	Reserved	–	–	–
0x1C	DDRSDRC Low-power Register	DDRSDRC_LPR	Read/Write	0x10000
0x20	DDRSDRC Memory Device Register	DDRSDRC_MD	Read/Write	0x10
0x24	DDRSDRC DLL Information Register	DDRSDRC_DLL	Read-only	0x00000001
0x28	Reserved	–	–	–
0x2C	DDRSDRC High Speed Register	DDRSDRC_HS	Read/Write	0x0
0x54–0xE0	Reserved	–	–	–
0xE4	DDRSDRC Write Protection Mode Register	DDRSDRC_WPMR	Read/Write	0x00000000
0xE8	DDRSDRC Write Protection Status Register	DDRSDRC_WPSR	Read-only	0x00000000
0xEC–0xFC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

30.7.1 DDRSDRC Mode Register

Name:DDRSDRC_MR

Address:0xFFFFE800

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	MODE		

This register can only be written if the WPEN bit is cleared in the DDRSDRC Write Protection Mode Register.

MODE: DDRSDRC Command Mode

This field defines the command issued by the DDRSDRC when the SDRAM device is accessed. This register is used to initialize the SDRAM device and to activate Deep Power-down mode.

Value	Name	Description
0	NORMAL	Normal mode: Any access to the DDRSDRC will be decoded normally. To activate this mode, command must be followed by a write to the SDRAM.
1	NOP	The DDRSDRC issues a NOP command when the SDRAM device is accessed regardless of the cycle. To activate this mode, command must be followed by a write to the SDRAM.
2	ALLBKPRECH	The DDRSDRC issues an “All Banks Precharge” command when the SDRAM device is accessed regardless of the cycle. To activate this mode, command must be followed by a write to the SDRAM.
3	LOADMODREG	The DDRSDRC issues a “Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. To activate this mode, command must be followed by a write to the SDRAM.
4	AUTOREFRESH	The DDRSDRC issues an “Auto-refresh” command when the SDRAM device is accessed regardless of the cycle. Previously, an “All Banks Precharge” command must be issued. To activate this mode, command must be followed by a write to the SDRAM.
5	EXTLOADMODREG	The DDRSDRC issues an “Extended Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. To activate this mode, the “Extended Load Mode Register” command must be followed by a write to the SDRAM. The write in the SDRAM must be done in the appropriate bank.
6	DEEPPOWER	Deep power mode: Access to Deep Power-down mode

30.7.2 DDRSDRC Refresh Timer Register

Name:DDRSDRC_RTR

Address:0xFFFFE804

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	COUNT			

7	6	5	4	3	2	1	0
COUNT							

This register can only be written if the WPEN bit is cleared in the DDRSDRC Write Protection Mode Register.

COUNT: DDRSDRC Refresh Timer Count

This 12-bit field is loaded into a timer which generates the refresh pulse. Each time the refresh pulse is generated, a refresh sequence is initiated.

SDRAM devices require a refresh of all rows every 64 ms. The value to be loaded depends on the DDRSDRC clock frequency (MCK: Master Clock) and the number of rows in the device.

For example, for an SDRAM with 8192 rows and a 100 MHz Master clock, the value of Refresh Timer Count bit is programmed as $((64 \times 10^{-3}) / 8192) \times 100 \times 10^{6} = 781$ (0x030D).

© 2017 Microchip Technology Inc.

30.7.3 DDRSDRC Configuration Register

Name:DDRSDRC_CR

Address:0xFFFFE808

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	DECOD	–	NB	–	ACTBST	–	EBISHARE

15	14	13	12	11	10	9	8
–	OCD			–	–	DIS_DLL	DIC

7	6	5	4	3	2	1	0
DLL	CAS			NR		NC	

This register can only be written if the if the WPEN bit is cleared in the DDRSDRC Write Protection Mode Register.

NC: Number of Column Bits

Reset value is 9 column bits.

SDR-SDRAM devices with eight columns in 16-bit mode are not supported.

Value	Name	Description
0	DDR9_SDR8	9-bit for DDR, 8-bit for SDR
1	DDR10_SDR9	10-bit for DDR, 9-bit for SDR
2	DDR11_SDR10	11-bit for DDR, 10-bit for SDR
3	DDR12_SDR11	12-bit for DDR, 11-bit for SDR

NR: Number of Row Bits

Reset value is 12 row bits.

Value	Name	Description
0	11_BIT	11 row bit
1	12_BIT	12 row bit
2	13_BIT	13 row bit
3	14_BIT	14 row bit

CAS: CAS Latency

Reset value is 2 cycles.

Value	DDR2 CAS Latency	SDR CAS Latency
0	Reserved	Reserved
1	Reserved	Reserved
2	Reserved	2
3	3	3
4	Reserved	Reserved
5	Reserved	Reserved

© 2017 Microchip Technology Inc.

Value	DDR2 CAS Latency	SDR CAS Latency
6	Reserved	Reserved
7	Reserved	Reserved

DLL: Reset DLL

Reset value is 0.

This field defines the value of Reset DLL.

0: Disable DLL reset.

1: Enable DLL reset.

This value is used during the power-up sequence.

Note: This field is found only in DDR2-SDRAM devices.

DIC: Output Driver Impedance Control

Reset value is 0. This field name may be described as “DS” in some memory datasheets and defines the output drive strength. This value is used during the power-up sequence.

Value	Name	Description
0	DDR1_2_NORMALSTRENGTH	Normal driver strength
1	DDR1_2_WEAKSTRENGTH	Weak driver strength

Note: This field is found only in DDR2-SDRAM devices.

DIS_DLL: Disable DLL

Reset value is 0.

0: Enable DLL

1: Disable DLL

Note: This field is found only in DDR2-SDRAM devices.

OCD: Off-chip Driver

Reset value is 7.

Note: The SDRAM controller supports only two values for OCD: 7 (calibration default) and 0 (exit from calibration). These values MUST always be programmed during the initialization sequence. The “calibration default” must be programmed before programming “exit calibration and maintain settings.”

Note: This field is found only in DDR2-SDRAM devices.

Value	Name	Description
0	DDR2_EXITCALIB	Exit from OCD calibration mode and maintain settings
7	DDR2_DEFAULT_CALIB	OCD calibration default

EBISHARE: External Bus Interface is Shared

The DDR controller embedded in the EBI is used at the same time as another memory controller (SMC, ...)

Reset value is 0.

0: Only the DDR controller function is used.

1: The DDR controller shares the EBI with another memory controller (SMC, NAND, ...)

© 2017 Microchip Technology Inc.

ACTBST: Active Bank X to Burst Stop Read Access Bank Y

Reset value is 0.

0: After an Active command in Bank X, Burst Stop command can be issued to another bank to stop current read access.

1: After an Active command in Bank X, Burst Stop command cannot be issued to another bank to stop current read access.

This field is unique to SDR-SDRAM, Low-power SDR-SDRAM and Low-power DDR1-SDRAM devices.

NB: Number of Banks

Reset value is four banks.

Value	Name	Description
0	4_BANKS	4 Banks
1	8_BANKS	8 Banks

Note: Only DDR-SDRAM 2 devices support eight internal banks.

DECOD: Type of Decoding

Reset value is 0: sequential decoding.

Value	Name	Description
0	SEQUENTIAL	Sequential Decoding
1	INTERLEAVED	Interleaved Decoding

30.7.4 DDRSDRC Timing Parameter 0 Register

Name:DDRSDRC_TPR0

Address:0xFFFFE80C

Access: Read/Write

31	30	29	28	27	26	25	24
TMRD				REDUCE_WRRD	TWTR		
23	22	21	20	19	18	17	16
TRRD				TRP			
15	14	13	12	11	10	9	8
TRC				TWR			
7	6	5	4	3	2	1	0
TRCD				TRAS			

This register can only be written if the WPEN bit is cleared in the DDRSDRC Write Protection Mode Register.

TRAS: Active to Precharge Delay

Reset value is 5 cycles.

This field defines the delay between an Activate command and a Precharge command in number of cycles. Number of cycles is between 0 and 15.

TRCD: Row to Column Delay

Reset value is 2 cycles.

This field defines the delay between an Activate command and a Read/Write command in number of cycles. Number of cycles is between 0 and 15.

TWR: Write Recovery Delay

Reset value is 2 cycles.

This field defines the Write Recovery Time in number of cycles. Number of cycles is between 1 and 15.

TRC: Row Cycle Delay

Reset value is 7 cycles.

This field defines the delay between an Activate command and Refresh command in number of cycles. Number of cycles is between 0 and 15

TRP: Row Precharge Delay

Reset value is 2 cycles.

This field defines the delay between a Precharge command and another command in number of cycles. Number of cycles is between 0 and 15.

TRRD: Active BankA to Active BankB

Reset value is 2 cycles.

This field defines the delay between an Active command in BankA and an active command in BankB in number of cycles. Number of cycles is between 1 and 15.

TWTR: Internal Write to Read Delay

Reset value is 0.

This field is relevant only for Low-power DDR1-SDRAM devices and DDR2-SDRAM devices.

© 2017 Microchip Technology Inc.

This field defines the internal write to read command Time in number of cycles. Number of cycles is between 1 and 7.

Value	Name	Description
1	ONE	1 SDCK clock cycle delay
2	TWO	2 SDCK clock cycle delay
3	THREE	3 SDCK clock cycle delay
4	FOUR	4 SDCK clock cycle delay
5	FIVE	5 SDCK clock cycle delay
6	SIX	6 SDCK clock cycle delay
7	SEVEN	7 SDCK clock cycle delay

In the case of low-power DDR1-SDRAM device the coding is different.

Value	Name	Description
0	ONE	Does 1
1	TWO	Does 2

REDUCE_WRRD: Reduce Write to Read Delay

Reset value is 0.

This field reduces the delay between write to read access for low-power DDR-SDRAM devices with a latency equal to 2. To use this feature, TWTR field must be equal to 0. Important to note is that some devices do not support this feature.

TMRD: Load Mode Register Command to Active or Refresh Command

Reset value is 2 cycles.

This field defines the delay between a Load Mode Register command and an active or refresh command in number of cycles. Number of cycles is between 0 and 15.

30.7.5 DDRSDRC Timing Parameter 1 Register

Name:DDRSDRC_TPR1

Address:0xFFFFE810

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	TXP			
23	22	21	20	19	18	17	16
TXSRD							
15	14	13	12	11	10	9	8
TXSNR							
7	6	5	4	3	2	1	0
–	–	–	TRFC				

This register can only be written if the WPEN bit is cleared in the DDRSDRC Write Protection Mode Register.

TRFC: Row Cycle Delay

Reset value is 8 cycles.

This field defines the delay between a Refresh and an Activate command or Refresh command in number of cycles. Number of cycles is between 0 and 31.

TXSNR: Exit Self Refresh Delay to Non-read Command

Reset value is 8 cycles.

This field defines the delay between CKE set high and a non Read command in number of cycles. Number of cycles is between 0 and 255. This field is used for SDR-SDRAM and DDR-SDRAM devices. In the case of SDR-SDRAM devices and Low-power DDR1-SDRAM, this field is equivalent to TXSR timing.

TXSRD: Exit Self Refresh Delay to Read Command

Reset value is 200 cycles.

This field defines the delay between CKE set high and a Read command in number of cycles. Number of cycles is between 0 and 255 cycles.This field is unique to DDR-SDRAM devices. In the case of a Low-power DDR1-SDRAM, this field must be written to 0.

TXP: Exit Power-down Delay to First Command

Reset value is 3 cycles.

This field defines the delay between CKE set high and a Valid command in number of cycles. Number of cycles is between 0 and 15 cycles. This field is unique to Low-power DDR1-SDRAM devices and DDR2-SDRAM devices.

© 2017 Microchip Technology Inc.

30.7.6 DDRSDRC Timing Parameter 2 Register

Name:DDRSDRC_TPR2

Address:0xFFFFE814

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	TFAW			

15	14	13	12	11	10	9	8
–	TRTP			TRPA			

7	6	5	4	3	2	1	0
TXARDS				TXARD			

This register can only be written if the WPEN bit is cleared in the DDRSDRC Write Protection Mode Register.

TXARD: Exit Active Power Down Delay to Read Command in Mode “Fast Exit”

Reset value is 2 cycles.

This field defines the delay between CKE set high and a Read command in number of cycles. Number of cycles is between 0 and 15.

Note: This field is found only in DDR2-SDRAM devices.

TXARDS: Exit Active Power Down Delay to Read Command in Mode “Slow Exit”

Reset value is 6 cycles.

This field defines the delay between CKE set high and a Read command in number of cycles. Number of cycles is between 0 and 15.

Note: This field is found only in DDR2-SDRAM devices.

TRPA: Row Precharge All Delay

Reset value is 0 cycles.

This field defines the delay between an All Banks Precharge command and another command in number of cycles. Number of cycles is between 0 and 15.

Note: This field is found only in DDR2-SDRAM devices.

TRTP: Read to Precharge

Reset value is 2 cycles.

This field defines the delay between Read command and a Precharge command in number of cycle.

Number of cycles is between 0 and 7.

TFAW: Four Active Window

Reset value is 4 cycles.

DDR2 devices with 8-banks (1 GB or larger) have an additional requirement: t_{FAW}. This requires that no more than four ACTIVATE commands may be issued in any given t_{FAW} (MIN) period.

Number of cycles is between 0 and 15.

Note: This field is found only in DDR-SDRAM 2 devices with eight internal banks

30.7.7 DDRSDRC Low-power Register

Name:DDRSDRC_LPR

Address:0xFFFFE81C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	UPD_MR		–	–	–	APDE

15	14	13	12	11	10	9	8
–	–	TIMEOUT		–	DS		

7	6	5	4	3	2	1	0
–	PASR				CLK_FR	LPCB	

LPCB: Low-power Command Bits

Reset value is 0.

Value	Name	Description
0	NOLOWPOWER	Low-power feature is inhibited: no power-down, Self Refresh and deep power mode are issued to the SDRAM device.
1	SELFREFRESH	The DDRSDRC issues a Self Refresh command to the SDRAM device, the clock(s) is/are de-activated and the CKE signal is set low. The SDRAM device leaves the Self Refresh mode when accessed and enters it after the access.
2	POWERDOWN	The DDRSDRC issues a Power-down command to the SDRAM device after each access, the CKE signal is set low. The SDRAM device leaves the Power-down mode when accessed and enters it after the access.
3	DEEPPOWERDOWN	The DDRSDRC issues a Deep Power-down command to the Low-power SDRAM device. **Note:** This mode is unique to Low-power SDRAM devices.

CLK_FR: Clock Frozen Command Bit

Reset value is 0.

This field sets the clock low during Power-down mode or during Deep Power-down mode. Some SDRAM devices do not support freezing the clock during Power-down mode or during Deep Power-down mode. Refer to the SDRAM device datasheet for details on this.

0: Clock(s) is/are not frozen.

1: Clock(s) is/are frozen.

PASR: Partial Array Self Refresh

Reset value is 0.

This field is unique to Low-power SDRAM. It is used to specify whether only one quarter, one half or all banks of the SDRAM array are enabled. Disabled banks are not refreshed in Self Refresh mode.

The values of this field are dependant on Low-power SDRAM devices.

After the initialization sequence, as soon as PASR field is modified, Extended Mode Register in the external device memory is accessed automatically and PASR field is updated. Depending on the value of the the UPD_MR field, the update is done before entering in Self Refresh mode or during a refresh command and a pending read or write access.

© 2017 Microchip Technology Inc.

DS: Drive Strength

Reset value is 0.

Note: This field is unique to Low-power SDRAM.

It selects the driver strength of SDRAM output (see memory devices datasheet for details).

After the initialization sequence, as soon as DS field is modified, Extended Mode Register is accessed automatically and DS bits are updated. Depending on the value of the UPD_MR field, the update is done before entering in Self Refresh mode or during a refresh command and a pending read or write access.

TIMEOUT: Time Between Last Transfer and Low Power Mode

Reset value is 0. This field defines when low-power mode is enabled.

Value	Name	Description
0	NONE	Self Refresh mode is enabled as soon as the SDRAM device is not selected
1	CLK64	Self Refresh mode is enabled 64 clock cycles after completion of the last access
2	CLK128	Self Refresh mode is enabled 128 clock cycles after completion of the last access

APDE: Active Power Down Exit Time

Reset value is 1.

Note: This mode is unique to DDR2-SDRAM devices.

This mode allows to determine the active power-down mode, which determines performance versus power saving.

Value	Name	Description
0	DDR2_FAST_EXIT	Fast Exit from Power Down. DDR2-SDRAM devices only.
1	DDR2_SLOW_EXIT	Slow Exit from Power Down. DDR2-SDRAM devices only.

After the initialization sequence, as soon as APDE bit is modified, Extended Mode Register (located in the memory of the external device) is accessed automatically and APDE bit is updated. Depending on the value of the UPD_MR field, the update is done before entering in Self Refresh mode or during a refresh command and a pending read or write access.

UPD_MR: Update Load Mode Register and Extended Mode Register

Reset value is 0.

This bit is used to enable or disable automatic update of the Load Mode Register and Extended Mode Register. This update depends on the DDRSDRC integration in a system. DDRSDRC can either share or not, an external bus with another controller.

Value	Name	Description
0	NO_UPDATE	Update is disabled.
1	UPDATE_SHAREDBUS	DDRSDRC shares external bus. Automatic update is done during a refresh command and a pending read or write access in SDRAM device.
2	UPDATE_NOSHAREDBUS	DDRSDRC does not share external bus. Automatic update is done before entering in Self Refresh mode.

30.7.8 DDRSDRC Memory Device Register

Name:DDRSDRC_MD

Address:0xFFFFE820

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	DBW	–	MD		

This register can only be written if the WPEN bit is cleared in the DDRSDRC Write Protection Mode Register.

MD: Memory Device

Indicates the type of memory used. Reset value is for SDR-SDRAM device.

Value	Name	Description
0	SDR_SDRAM	SDR-SDRAM memory device
1	LPSDR_SDRAM	Low-power SDR-SDRAM
2	DDR1_SDRAM	Reserved
3	LPDDR1_SDRAM	Low-power DDR1-SDRAM
4	DDR2_SDRAM	DDR2-SDRAM

DBW: Data Bus Width

Reset value is 16 bits.

Value	Name	Description
0	BUS_32BIT	Data bus width is 32 bits (reserved for SDR-SDRAM device)
1	BUS_16BIT	Data bus width is 16 bits

© 2017 Microchip Technology Inc.

30.7.9 DDRSDRC DLL Register

Name:DDRSDRC_DLL

Address:0xFFFFE824

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
MDVAL							
7	6	5	4	3	2	1	0
–	–	–	–	–	MDOVF	MDDEC	MDINC

The DLL logic is internally used by the controller in order to delay DQS inputs. This is necessary to center the strobe time and the data valid window.

MDINC: DLL Master Delay Increment

0: The DLL is not incrementing the Master delay counter.

1: The DLL is incrementing the Master delay counter.

MDDEC: DLL Master Delay Decrement

0: The DLL is not decrementing the Master delay counter.

1: The DLL is decrementing the Master delay counter.

MDOVF: **DLL Master Delay Overflow Flag**

0: The Master delay counter has not reached its maximum value, or the Master is not locked yet.

1: The Master delay counter has reached its maximum value, the Master delay counter increment is stopped and the DLL forces the Master lock. If this flag is set, it means the DDRSDRC clock frequency is too low compared to Master delay line number of elements.

MDVAL: **DLL Master Delay Value**

Value of the Master delay counter.

30.7.10 DDRSDRC High Speed Register

Name:DDRSDRC_HS

Address:0xFFFFE82C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	DIS_ANTICIP_READ	–	–

This register can only be written if the WPEN bit is cleared in the DDRSDRC Write Protection Mode Register.

DIS_ANTICIP_READ: Anticipated Read Access

0: Anticipated read access is enabled.

1: Anticipated read access is disabled (default).

DIS_ANTICIP_READ allows DDR2 read access optimization with multi-port.

This feature is based on the “bank open policy”. Therefore, software must map different buffers in different DDR2 banks to use the feature.

© 2017 Microchip Technology Inc.

30.7.11 DDRSDRC Write Protection Mode Register

Name:DDRSDRC_WPMR

Address:0xFFFFE8E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPEN

WPEN: Write Protection Enable

0: Disables the Write Protection if WPKEY corresponds to 0x444452 (“DDR” in ASCII).

1: Enables the Write Protection if WPKEY corresponds to 0x444452 (“DDR” in ASCII).

WPKEY: Write Protection Key

Value	Name	Description
0x444452	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

30.7.12 DDRSDRC Write Protection Status Register

Name:DDRSDRC_WPSR

Address:0xFFFFE8E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

WPVS: Write Protection Violation Status

0: No write protection violation has occurred since the last read of the DDRSDRC_WPSR.

1: A write protection violation has occurred since the last read of the DDRSDRC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

WPVSRC: Write Protection Violation Source

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Note: Reading DDRSDRC_WPSR automatically clears all fields.

© 2017 Microchip Technology Inc.

31. DMA Controller (DMAC)

31.1 Description

The DMA Controller (DMAC) is an AHB-central DMA controller core that transfers data from a source peripheral to a destination peripheral over one or more AMBA buses. One channel is required for each source/destination pair. In the most basic configuration, the DMAC has one master interface and one channel. The master interface reads the data from a source and writes it to a destination. Two AMBA transfers are required for each DMAC data transfer. This is also known as a dual-access transfer.

The DMAC is programmed via the APB interface.

31.2 Embedded Characteristics

- 2 AHB-Lite Master Interfaces
- DMA Module Supports the Following Transfer Schemes: Peripheral-to-Memory, Memory-to-Peripheral, Peripheral-to-Peripheral and Memory-to-Memory
- Source and Destination Operate independently on BYTE (8-bit), HALF-WORD (16-bit) and WORD (32-bit)
- Supports Hardware and Software Initiated Transfers
- Supports Multiple Buffer Chaining Operations
- Supports Incrementing/decrementing/fixed Addressing Mode Independently for Source and Destination
- Supports Programmable Address Increment/decrement on User-defined Boundary Condition to Enable Picture-in-Picture Mode
- Programmable Arbitration Policy, Modified Round Robin and Fixed Priority are Available
- Supports Specified Length and Unspecified Length AMBA AHB Burst Access to Maximize Data Bandwidth
- AMBA APB Interface Used to Program the DMA Controller
- 8 DMA Channels 29 External Request Lines
- Embedded FIFO
- Channel Locking and Bus Locking Capability
- Register Write Protection

© 2017 Microchip Technology Inc.

31.3 DMA Controller Peripheral Connections

The DMA Controller handles the transfer between peripherals and memory and receives triggers from the peripherals listed in the following table.

For each listed DMA channel number, the SIF and/or DIF fields in the Channel x Control B Register (DMAC_CTRLBx) must be programmed with a value compatible to the MATRIX “Master to Slave Access” definition provided in Section 25. ”Bus Matrix (MATRIX)”. See Section 31.8.17 ”DMAC Channel x [x = 0..7] Control B Register” (where x is the DMA Channel Number).

Depending on transfer descriptor location, the DSCR_IF field must be programmed with a value compatible to the MATRIX “Master to Slave Access” definition provided in Section 25. ”Bus Matrix (MATRIX)”. See Section 31.8.15 ”DMAC Channel x [x = 0..7] Descriptor Address Register” (where x is the DMA Channel Number).

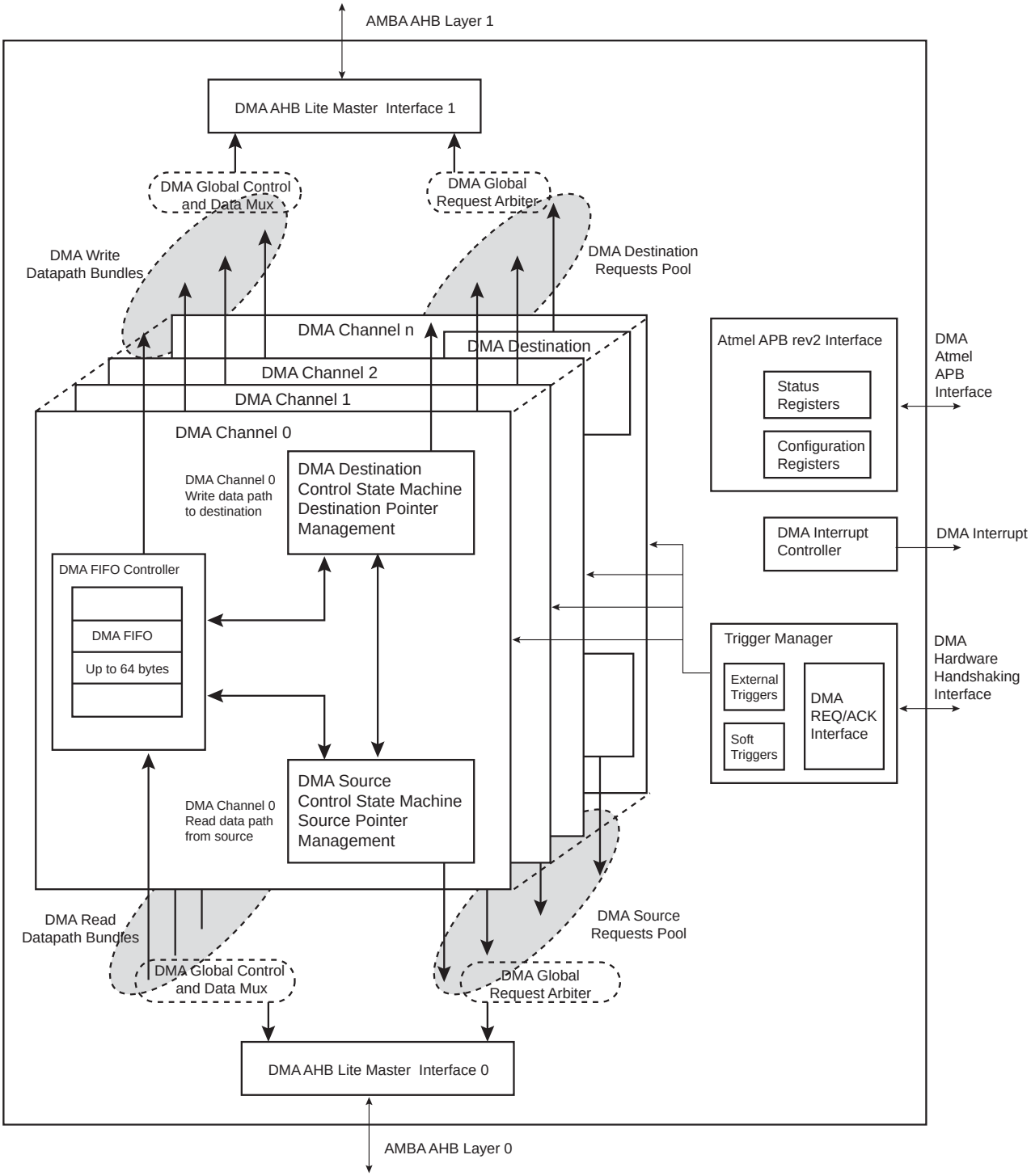
Table 31-1: DMA Channel Definition

Instance Name	Channel T/R	DMA Channel HW Interface Number
HSMCI	RX/TX	0
SPI0	TX	1
SPI0	RX	2
SPI1	TX	3
SPI1	RX	4
USART0	TX	5
USART0	RX	6
USART1	TX	7
USART1	RX	8
USART2	TX	9
USART2	RX	10
USART3	TX	11
USART3	RX	12
TWI0	TX	13
TWI0	RX	14
TWI1	TX	15
TWI1	RX	16
UART0	TX	17
UART0	RX	18
UART1	TX	19
UART1	RX	20
SSC	TX	21
SSC	RX	22
ADC	RX	23
DBGU	TX	24
DBGU	RX	25
AES	TX	26
AES	RX	27
SHA	TX	28

© 2017 Microchip Technology Inc.

31.4 Block Diagram

Figure 31-1: DMA Controller (DMAC) Block Diagram



31.5 Product Dependencies

31.5.1 Interrupt Sources

The DMAC interrupt line is connected to one of the internal sources of the interrupt controller. Using the DMAC interrupt requires prior programming of the interrupt controller.

Table 31-2: Peripheral IDs

Instance	ID
DMAC	20

31.6 Functional Description

31.6.1 Basic Definitions

Source peripheral: Device on an AMBA layer from where the DMAC reads data, which is then stored in the channel FIFO. The source peripheral teams up with a destination peripheral to form a channel.

Destination peripheral: Device to which the DMAC writes the stored data from the FIFO (previously read from the source peripheral).

Memory: Source or destination that is always “ready” for a DMAC transfer and does not require a handshaking interface to interact with the DMAC.

Programmable Arbitration Policy: Modified Round Robin and Fixed Priority are available by means of the ARB_CFG bit in the Global Configuration Register (DMAC_GCFG). The fixed priority is linked to the channel number. The highest DMAC channel number has the highest priority.

Channel: Read/write datapath between a source peripheral on one configured AMBA layer and a destination peripheral on the same or different AMBA layer that occurs through the channel FIFO. If the source peripheral is not memory, then a source handshaking interface is assigned to the channel. If the destination peripheral is not memory, then a destination handshaking interface is assigned to the channel. Source and destination handshaking interfaces can be assigned dynamically by programming the channel registers.

Master interface: DMAC is a master on the AHB bus reading data from the source and writing it to the destination over the AHB bus.

Slave interface: The APB interface over which the DMAC is programmed. The slave interface in practice could be on the same layer as any of the master interfaces or on a separate layer.

Handshaking interface: A set of signal registers that conform to a protocol and *handshake* between the DMAC and source or destination peripheral to control the transfer of a single or chunk transfer between them. This interface is used to request, acknowledge, and control a DMAC transaction. A channel can receive a request through one of two types of handshaking interface: hardware or software.

Hardware handshaking interface: Uses hardware signals to control the transfer of a single or chunk transfer between the DMAC and the source or destination peripheral.

Software handshaking interface: Uses software registers to control the transfer of a single or chunk transfer between the DMAC and the source or destination peripheral. No special DMAC handshaking signals are needed on the I/O of the peripheral. This mode is useful for interfacing an existing peripheral to the DMAC without modifying it.

Transfer hierarchy: Figure 31-2 illustrates the hierarchy between DMAC transfers, buffer transfers, chunk or single, and AMBA transfers (single or burst) for non-memory peripherals. Figure 31-3 shows the transfer hierarchy for memory.

© 2017 Microchip Technology Inc.

Figure 31-2: DMAC Transfer Hierarchy for Non-Memory Peripheral

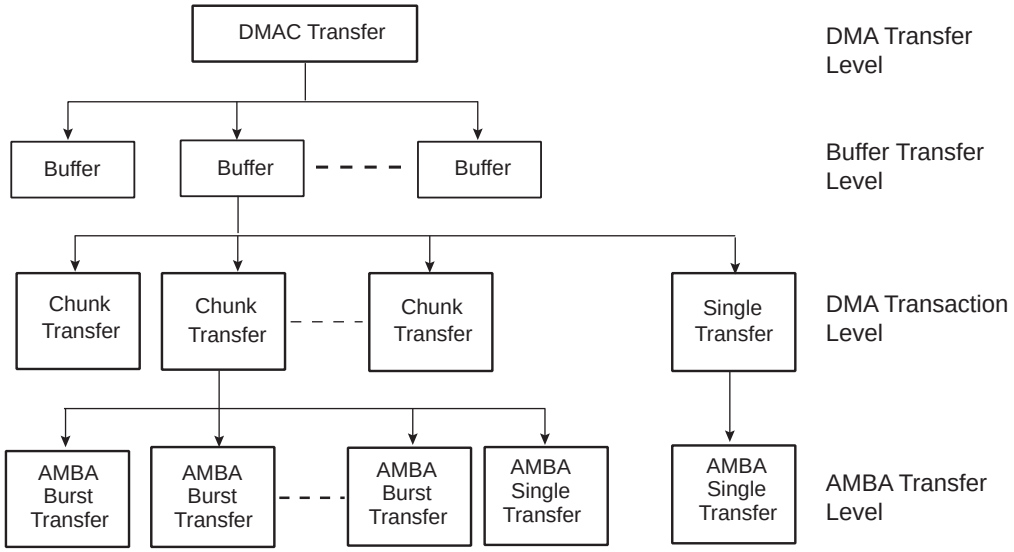
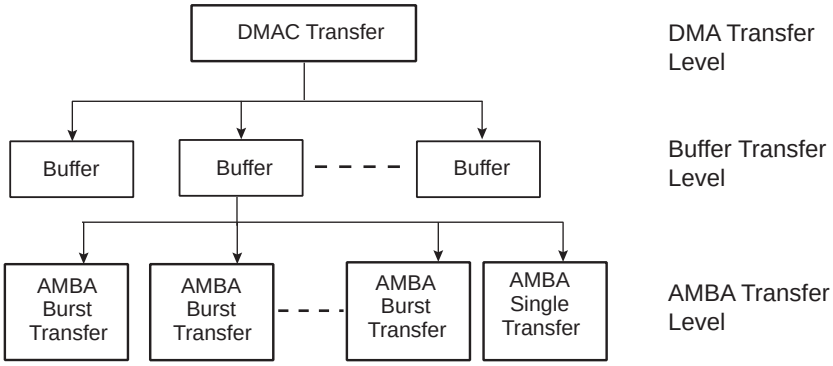


Figure 31-3: DMAC Transfer Hierarchy for Memory



Buffer: A buffer of DMAC data. The amount of data (length) is determined by the flow controller. For transfers between the DMAC and memory, a buffer is broken directly into a sequence of AMBA bursts and AMBA single transfers.

For transfers between the DMAC and a non-memory peripheral, a buffer is broken into a sequence of DMAC transactions (single and chunks). These are in turn broken into a sequence of AMBA transfers.

Transaction: A basic unit of a DMAC transfer as determined by either the hardware or software handshaking interface. A transaction is only relevant for transfers between the DMAC and a source or destination peripheral if the source or destination peripheral is a non-memory device. There are two types of transactions: single transfer and chunk transfer.

- **Single transfer:** The length of a single transaction is always 1 and is converted to a single AMBA access.
- **Chunk transfer:** The length of a chunk is programmed into the DMAC. The chunk is then converted into a sequence of AHB access.DMAC executes each AMBA burst transfer by performing incremental bursts that are no longer than 16 beats.

DMAC transfer: Software controls the number of buffers in a DMAC transfer. Once the DMAC transfer has completed, then hardware within the DMAC disables the channel and can generate an interrupt to signal the completion of the DMAC transfer. It is then possible to re-program the channel for a new DMAC transfer.

Single-buffer DMAC transfer: Consists of a single buffer.

Multi-buffer DMAC transfer: A DMAC transfer may consist of multiple DMAC buffers. Multi-buffer DMAC transfers are supported through buffer chaining (linked list pointers), auto-reloading of channel registers, and contiguous buffers. The source and destination can independently select which method to use.

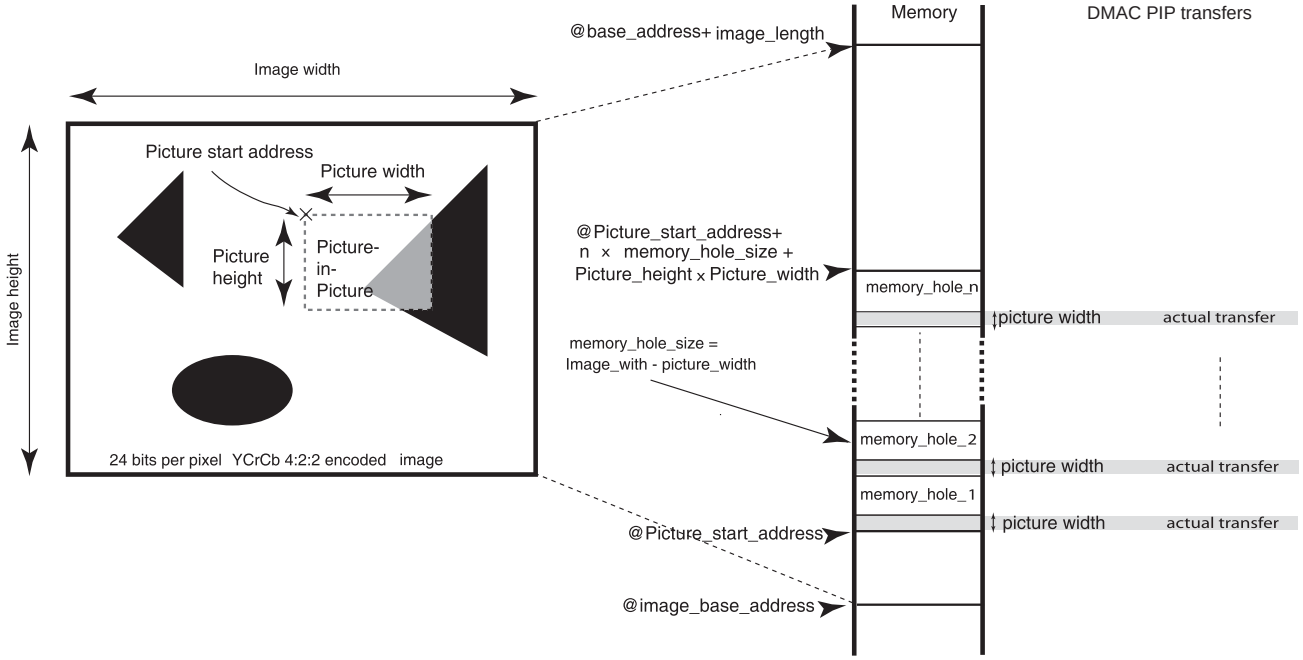
- **Linked lists (buffer chaining) –** A descriptor pointer (DSCR) points to the location in system memory where the next linked list item (LLI) exists. The LLI is a set of registers that describe the next buffer (buffer descriptor) and a descriptor pointer register. The DMAC fetches the LLI at the beginning of every buffer when buffer chaining is enabled.

- **Replay –** The DMAC automatically reloads the channel registers at the end of each buffers to the value when the channel was first enabled.
- **Contiguous buffers –** Where the address of the next buffer is selected to be a continuation from the end of the previous buffer.

Picture-in-Picture Mode: DMAC contains a Picture-in-Picture mode support. When this mode is enabled, addresses are automatically incremented by a programmable value when the DMAC channel transfer count reaches a user defined boundary.

Figure 31-4 illustrates a memory mapped image 4:2:2 encoded located at image_base_address in memory. A user defined start address is defined at Picture_start_address. The incremented value is set to memory_hole_size = image_width - picture_width, and the boundary is set to picture_width.

Figure 31-4: Picture-In-Picture Mode Support



Channel locking: Software can program a channel to keep the AHB master interface by locking the arbitration for the master bus interface for the duration of a DMAC transfer, buffer, or chunk.

Bus locking: Software can program a channel to maintain control of the AMBA bus by asserting hmastlock for the duration of a DMAC transfer, buffer, or transaction (single or chunk). Channel locking is asserted for the duration of bus locking at a minimum.

31.6.2 Memory Peripherals

Figure 31-3 shows the DMAC transfer hierarchy of the DMAC for a memory peripheral. There is no handshaking interface with the DMAC, and therefore the memory peripheral can never be a flow controller. Once the channel is enabled, the transfer proceeds immediately without waiting for a transaction request. The alternative to not having a transaction-level handshaking interface is to allow the DMAC to attempt AMBA transfers to the peripheral once the channel is enabled. If the peripheral slave cannot accept these AMBA transfers, it inserts wait states onto the bus until it is ready; it is not recommended that more than 16 wait states be inserted onto the bus. By using the handshaking interface, the peripheral can signal to the DMAC that it is ready to transmit/receive data, and then the DMAC can access the peripheral without the peripheral inserting wait states onto the bus.

31.6.3 Handshaking Interface

Handshaking interfaces are used at the transaction level to control the flow of single or chunk transfers. The operation of the handshaking interface is different and depends on whether the peripheral or the DMAC is the flow controller.

The peripheral uses the handshaking interface to indicate to the DMAC that it is ready to transfer/accept data over the AMBA bus. A non-memory peripheral can request a DMAC transfer through the DMAC using one of two handshaking interfaces:

- Hardware handshaking
- Software handshaking

© 2017 Microchip Technology Inc.

Software selects between the hardware or software handshaking interface on a per-channel basis. Software handshaking is accomplished through memory-mapped registers, while hardware handshaking is accomplished using a dedicated handshaking interface.

31.6.3.1 Software Handshaking

When the slave peripheral requires the DMAC to perform a DMAC transaction, it communicates this request by sending an interrupt to the CPU or interrupt controller.

The interrupt service routine then uses the software registers to initiate and control a DMAC transaction. These software registers are used to implement the software handshaking interface.

The SRC_H2SEL/DST_H2SEL bit in the Channel Configuration Register (DMAC_CFGx) must be cleared to enable software handshaking.

When the peripheral is not the flow controller, then the Software Last Transfer Flag Register (DMAC_LAST) is not used, and the values in these registers are ignored.

Chunk Transactions

Writing a ‘1’ to the Software Chunk Transfer Request Register (DMAC_CREQ[2x]) starts a source chunk transaction request, where x is the channel number. Writing a ‘1’ to the DMAC_CREQ[2x+1] register starts a destination chunk transfer request, where x is the channel number.

Upon completion of the chunk transaction, the hardware clears the DMAC_CREQ[*2x*] or DMAC_CREQ[2x+1].

Single Transactions

Writing a ‘1’ to the Software Single Request Register (DMAC_SREQ[2x]) starts a source single transaction request, where x is the channel number. Writing a ‘1’ to the DMAC_SREQ[2x+1] register starts a destination single transfer request, where x is the channel number.

Upon completion of the chunk transaction, the hardware clears the DMAC_SREQ[x] or DMAC_SREQ[2x+1].

The software can poll the relevant channel bit in the DMAC_CREQ[2x]/DMAC_CREQ[2x+1] and DMAC_SREQ[x]/DMAC_SREQ[2x+1] registers. When both are 0, then either the requested chunk or single transaction has completed.

31.6.4 DMAC Transfer Types

A DMAC transfer may consist of single or multi-buffer transfers. On successive buffers of a multi-buffer transfer, DMAC_SADDRx/DMAC_DADDRx in the DMAC are reprogrammed using either of the following methods:

- Buffer chaining using linked lists
- Replay mode
- Contiguous address between buffers

On successive buffers of a multi-buffer transfer, the DMAC_CTRLAx and DMAC_CTRLBx registers in the DMAC are re-programmed using either of the following methods:

- Buffer chaining using linked lists
- Replay mode

When buffer chaining using linked lists is the multi-buffer method of choice, and on successive buffers, DMAC_DSCRx in the DMAC is reprogrammed using the following method:

- Buffer chaining using linked lists

A buffer descriptor (LLI) consists of the following registers: DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLAx, and DMAC_CTRLBx. These registers, along with DMAC_CFGx, are used by the DMAC to set up and describe the buffer transfer.

31.6.4.1 Multi-buffer Transfers

Buffer Chaining Using Linked Lists

In this case, the DMAC re-programs the channel registers prior to the start of each buffer by fetching the buffer descriptor for that buffer from system memory. This is known as an LLI update.

DMAC buffer chaining is supported by using a descriptor pointer register (DMAC_DSCRx) that stores the address in memory of the next buffer descriptor. Each buffer descriptor contains the corresponding buffer descriptor (DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLAx DMAC_CTRLBx).

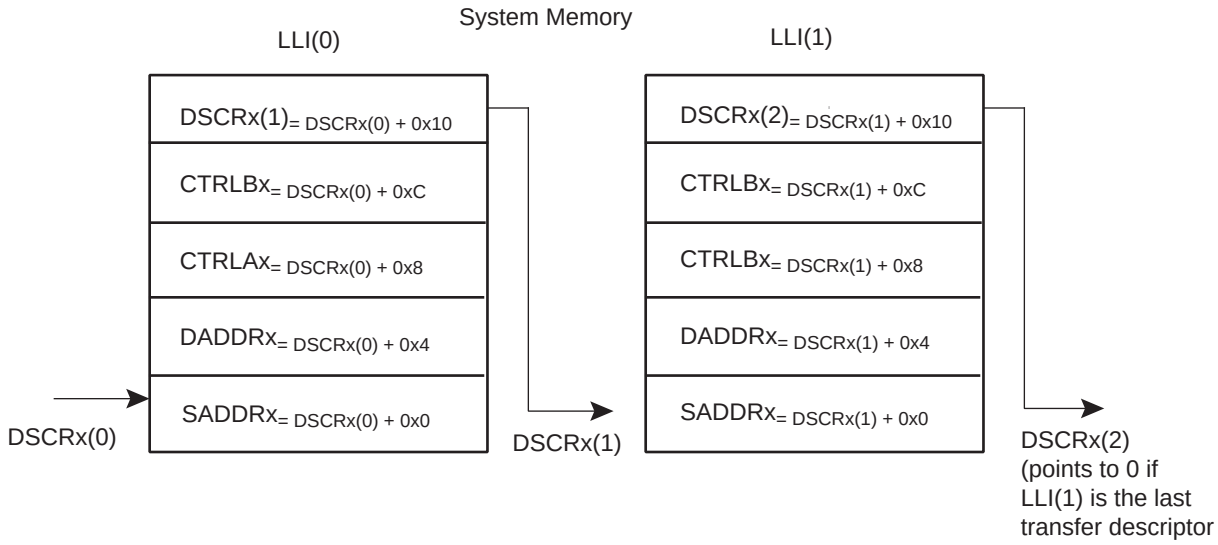
To set up buffer chaining, a sequence of linked lists must be programmed in memory.

DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLAx and DMAC_CTRLBx are fetched from system memory on an LLI update. The updated content of DMAC_CTRLAx is written back to memory on buffer completion. Figure 31-5 shows how to use chained linked lists in memory to define multi-buffer transfers using buffer chaining.

The Linked List multi-buffer transfer is initiated by programming DMAC_DSCRx with DSCRx(0) (LLI(0) base address) different from zero. Other fields and registers are ignored and overwritten when the descriptor is retrieved from memory.

The last transfer descriptor must be written to memory with its next descriptor address set to 0.

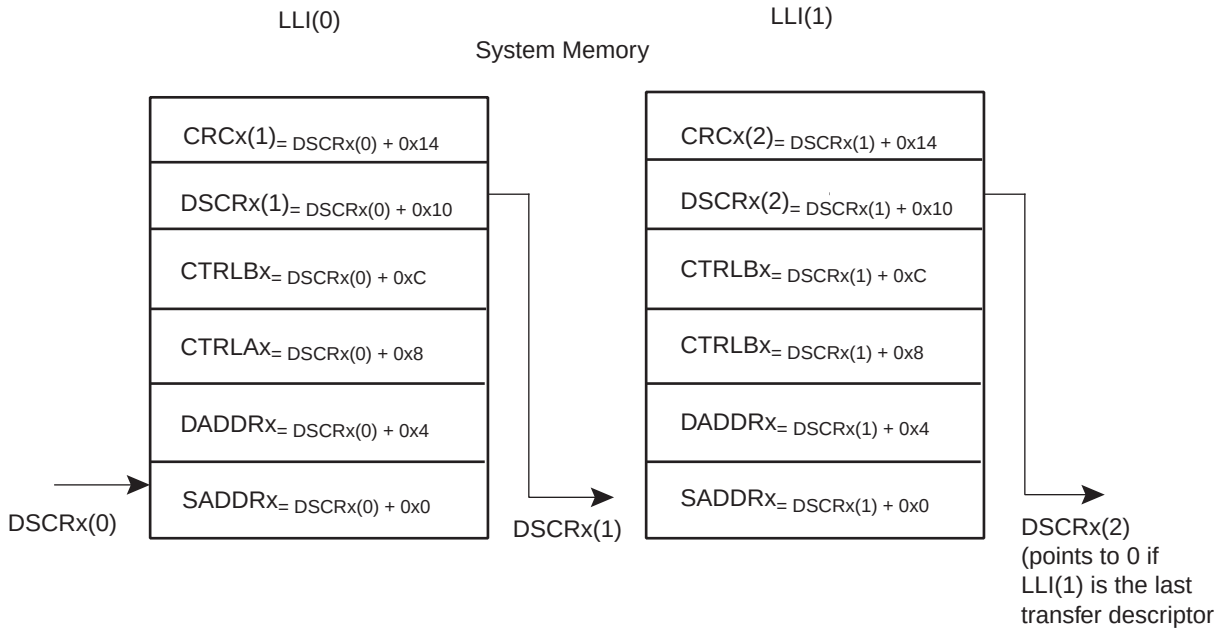
Figure 31-5: Multi-Buffer Transfer Using Linked List



Descriptor Integrity Check

When the Descriptor Integrity Check is enabled, a cyclic redundancy check information is attached to the descriptor. When fetched from the memory, the descriptor is verified through the use of a CRC16-CCIT (0x1021 polynom) by the DMAC channel. If a CRC error is detected, then the DICERR flag is set in the DMAC Error, Buffer Transfer and Chained Buffer Transfer Status Register (DMAC_EBCISR). The CRC16 is computed from MSB to LSB. The BTSIZE field and DONE bit in DMAC_CTRLAx are ignored and cleared.

Figure 31-6: Linked List with CRC16 Attached



© 2017 Microchip Technology Inc.

31.6.4.2 Programming DMAC for Multiple Buffer Transfers

Table 31-3: Multiple Buffers Transfer Management Table

Transfer Type	AUTO	SRC_REP	DST_REP	SRC_DSCR	DST_DSCR	BTSIZE	DSCR	SADDR	DADDR	Other Fields
1) Single Buffer or Last Buffer of a multiple buffer transfer	0	–	–	–	–	USR	0	USR	USR	USR
2) Multi-buffer transfer with contiguous DADDR	0	–	0	0	1	LLI	USR	LLI	CONT	LLI
3) Multi-buffer transfer with contiguous SADDR	0	0	–	1	0	LLI	USR	CONT	LLI	LLI
4) Multi-buffer transfer with LLI support	0	–	–	0	0	LLI	USR	LLI	LLI	LLI
5) Multi-buffer transfer with DADDR reloaded	0	–	1	0	1	LLI	USR	LLI	REP	LLI
6) Multi-buffer transfer with SADDR reloaded	0	1	–	1	0	LLI	USR	REP	LLI	LLI
7) Multi-buffer transfer with BTSIZE reloaded and contiguous DADDR	1	–	0	0	1	REP	USR	LLI	CONT	LLI
8) Multi-buffer transfer with BTSIZE reloaded and contiguous SADDR	1	0	–	1	0	REP	USR	CONT	LLI	LLI
9) Automatic mode channel is stalling BTsize is reloaded	1	0	0	1	1	REP	USR	CONT	CONT	REP
10) Automatic mode BTSIZE, SADDR and DADDR reloaded	1	1	1	1	1	REP	USR	REP	REP	REP
11) Automatic mode BTSIZE, SADDR reloaded and DADDR contiguous	1	1	0	1	1	REP	USR	REP	CONT	REP

Note 1: USR means that the register field is manually programmed by the user.

2: CONT means that address are contiguous.

3: REP means that the register field is updated with its previous value. If the transfer is the first one, then the user must manually program the value.

4: Channel stalled is true if the relevant BTC interrupt is not masked.

5: LLI means that the register field is updated with the content of the linked list item.

Replay Mode of Channel Registers

During automatic replay mode, the channel registers are reloaded with their initial values at the completion of each buffer and the new values used for the new buffer. Depending on the row number in Table 31-3, some or all of the DMAC_SADDRx, DMAC_DADDRx, DMAC_CTRLAx and DMAC_CTRLBx channel registers are reloaded from their initial value at the start of a buffer transfer.

Contiguous Address Between Buffers

In this case, the address between successive buffers is selected to be a continuation from the end of the previous buffer. Enabling the source or destination address to be contiguous between buffers is a function of DMAC_CTRLAx.SRC_DSCR, DMAC_CFGx.DST_REP, DMAC_CFGx.SRC_REP and DMAC_CTRLAx.DST_DSCR registers.

Suspension of Transfers Between Buffers

At the end of every buffer transfer, an end of buffer interrupt is asserted if:

- the channel buffer interrupt is unmasked, DMAC_EBCIMR.BTCx = '1', where x is the channel number.

Note: The Buffer Transfer Completed Interrupt is generated at the completion of the buffer transfer to the destination.

At the end of a chain of multiple buffers, an end of linked list interrupt is asserted if:

- the channel end of the Chained Buffer Transfer Completed Interrupt is unmasked, DMAC_EBCIMR.CBTCx = '1', when n is the channel number.

31.6.4.3 Ending Multi-buffer Transfers

All multi-buffer transfers must end as shown in Row 1 of Table 31-3. At the end of every buffer transfer, the DMAC samples the row number, and if the DMAC is in Row 1 state, then the previous buffer transferred was the last buffer and the DMAC transfer is terminated.

For rows 9, 10 and 11 of Table 31-3, (DMAC_DSCRx = 0 and DMAC_CTRLBx.AUTO is set), multi-buffer DMAC transfers continue until the automatic mode is disabled by clearing the DMAC_CTRLBx.AUTO bit. This bit should be programmed to zero in the end of buffer interrupt service routine that services the next-to-last buffer transfer. This puts the DMAC into Row 1 state.

For rows 2, 3, 4, 5, and 6 (DMAC_CRTLBx.AUTO cleared), the user must set up the last buffer descriptor in memory so that LLI.DMAC_DSCRx is set to 0.

31.6.5 Programming a Channel

Four registers, DMAC_DSCRx, DMAC_CTRLAx, DMAC_CTRLBx and DMAC_CFGx, need to be programmed to set up whether single or multi-buffer transfers take place, and which type of multi-buffer transfer is used. The different transfer types are shown in Table 31-3.

The "BTSIZE", "SADDR" and "DADDR" columns indicate where the values of DMAC_SARx, DMAC_DARx, DMAC_CTLx, and DMAC_LLPx are obtained for the next buffer transfer when multi-buffer DMAC transfers are enabled.

31.6.5.1 Programming Examples

Single-buffer Transfer (Row 1)

1. Read the ENAx bit in the DMAC Channel Handler Status Register (DMAC_CHSR) to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the DMAC Error, Buffer Transfer and Chained Buffer Transfer Status Register (DMAC_EBCISR).
3. Program the following channel registers:
   a) Write the starting source address in DMAC_SADDRx for channel x.
   b) Write the starting destination address in DMAC_DADDRx for channel x.
   c) Write the next descriptor address in DMA_DSCRx for channel x with 0x0.
   d) Program DMAC_CTRLAx, DMAC_CTRLBx and DMAC_CFGx according to Row 1 as shown in Table 31-3. Program DMAC_CTRLBx with both AUTO bits cleared.
   e) Write the control information for the DMAC transfer in DMAC_CTRLAx and DMAC_CTRLBx for channel x. For example, in the register, you can program the following:

- i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC field in DMAC_CTRLBx.
- ii. Set up the transfer characteristics, such as:

  Transfer width for the source in the SRC_WIDTH field.

  Transfer width for the destination in the DST_WIDTH field.

  Source AHB Master interface layer in the SIF field where source resides.

  Destination AHB Master Interface layer in the DIF field where destination resides.

  Incrementing/decrementing or fixed address for source in SRC_INCR field.

  Incrementing/decrementing or fixed address for destination in DST_INCR field.

© 2017 Microchip Technology Inc.

f) Write the channel configuration information into DMAC_CFGx for channel x.

- i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests. Writing a '0' activates the software handshaking interface to handle source/destination requests.
- ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign a handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

g) If source Picture-in-Picture mode is enabled (DMAC_CTRLBx.SRC_PIP is enabled), program DMAC_SPIPx for channel x.

h) If destination Picture-in-Picture mode is enabled (DMAC_CTRLBx.DST_PIP is enabled), program DMAC_DPIPx for channel x.

4. After the DMAC selected channel has been programmed, enable the channel by setting the ENAx bit in the DMAC Channel Handler Enable Register (DMAC_CHER), where x is the channel number. Make sure that the ENABLE bit (register bit 0) in DMAC_EN is set.
5. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carries out the buffer transfer.
6. Once the transfer completes, the hardware sets the interrupts and disables the channel. At this time, you can either respond to the Buffer Transfer Completed Interrupt or Chained Buffer Transfer Completed Interrupt, or poll for the DMAC_CHSR.ENAx bit until it is cleared by hardware, to detect when the transfer is complete.

Multi-buffer Transfer with Linked List for Source and Linked List for Destination (Row 4)

1. Read the DMAC_CHSR to choose a free (disabled) channel.
2. Set up the chain of Linked List Items (otherwise known as buffer descriptors) in memory. Write the control information in the LLI.DMAC_CTRLAx and LLI.DMAC_CTRLBx registers location of the buffer descriptor for each LLI in memory (see Figure 31-7) for channel x. For example, in the register, you can program the following:

   a) Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC field in DMAC_CTRLBx.

   b) Set up the transfer characteristics, such as:

- i. Transfer width for the source in the SRC_WIDTH field.
- ii. Transfer width for the destination in the DST_WIDTH field.
- iii. Source AHB master interface layer in the SIF field where source resides.
- iv. Destination AHB master interface layer in the DIF field where destination resides.
- v. Incrementing/decrementing or fixed address for source in SRC_INCR field.
- vi. Incrementing/decrementing or fixed address for destination DST_INCR field.

3. Write the channel configuration information into DMAC_CFGx for channel x.

   a) Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '0' activates the software handshaking interface to handle source/destination requests.

   b) If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

4. Make sure that the LLI.DMAC_CTRLBx register locations of all LLI entries in memory (except the last) are set as shown in Row 4 of Table 31-3. The LLI.DMAC_CTRLBx register of the last Linked List Item must be set as described in Row 1 of Table 31-3. Figure 31-5 shows a Linked List example with two list items.
5. Make sure that the LLI.DMAC_DSCRx register locations of all LLI entries in memory (except the last) are non-zero and point to the base address of the next Linked List Item.
6. Make sure that the LLI.DMAC_SADDRx/LLI.DMAC_DADDRx register locations of all LLI entries in memory point to the start source/destination buffer address preceding that LLI fetch.
7. Make sure that the LLI.DMAC_CTRLAx.DONE bit of the LLI.DMAC_CTRLAx register locations of all LLI entries in memory are cleared.
8. If source Picture-in-Picture mode is enabled (DMAC_CTRLBx.SRC_PIP is enabled), program DMAC_SPIPx for channel x.
9. If destination Picture-in-Picture is enabled (DMAC_CTRLBx.DST_PIP is enabled), program DMAC_DPIPx for channel x.
10. Clear any pending interrupts on the channel from the previous DMAC transfer by reading DMAC_EBCISR.
11. Program DMAC_CTRLBx and DMAC_CFGx according to Row 4 as shown in Table 31-3.
12. Program DMAC_DSCRx with DMAC_DSCRx(0), the pointer to the first Linked List item.
13. Finally, enable the channel by setting the DMAC_CHER.ENAx bit, where x is the channel number. The transfer is performed.
14. The DMAC fetches the first LLI from the location pointed to by DMAC_DSCRx(0).

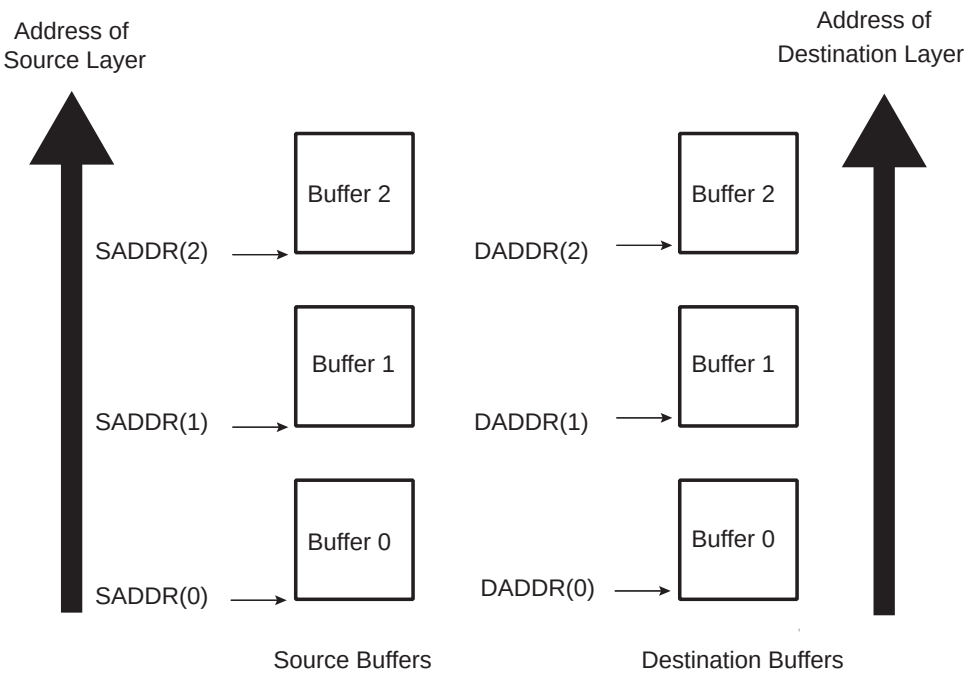
Note: The LLI.DMAC_SADDRx, LLI. DMAC_DADDRx, LLI.DMAC_DSCRx, LLI.DMAC_CTRLAx and LLI.DMAC_CTRLBx registers are fetched. The DMAC automatically reprograms the DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLBx and DMAC_CTRLAx channel registers from the DMAC_DSCRx(0).

15. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripheral). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carries out the buffer transfer.
16. Once the buffer of data is transferred, the DMAC_CTRLAx register is written out to system memory at the same location and on the same layer (DMAC_DSCRx.DSCR_IF) where it was originally fetched, that is, the location of the DMAC_CTRLAx register of the linked list item fetched prior to the start of the buffer transfer. Only DMAC_CTRLAx register is written out because only the DMAC_CTRLAx.BTSIZE and DMAC_CTRLAX.DONE bits have been updated by DMAC hardware. Additionally, the DMAC_CTRLAx.DONE bit is asserted when the buffer transfer has completed.

Note: Do not poll the DMAC_CTRLAx.DONE bit in the DMAC memory map. Instead, poll the LLI.DMAC_CTRLAx.DONE bit in the LLI for that buffer. If the poll LLI.DMAC_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC_CTRLAx.DONE bit was cleared at the start of the transfer.

17. The DMAC does not wait for the buffer interrupt to be cleared, but continues fetching the next LLI from the memory location pointed to by current DMAC_DSCRx and automatically reprograms the DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLAx and DMAC_CTRLBx channel registers. The DMAC transfer continues until the DMAC determines that the DMAC_CTRLBx and DMAC_DSCRx registers at the end of a buffer transfer match as described in Row 1 of Table 31-3. The DMAC then knows that the previous buffer transferred was the last buffer in the DMAC transfer. The DMAC transfer might look like that shown in Figure 31-7.

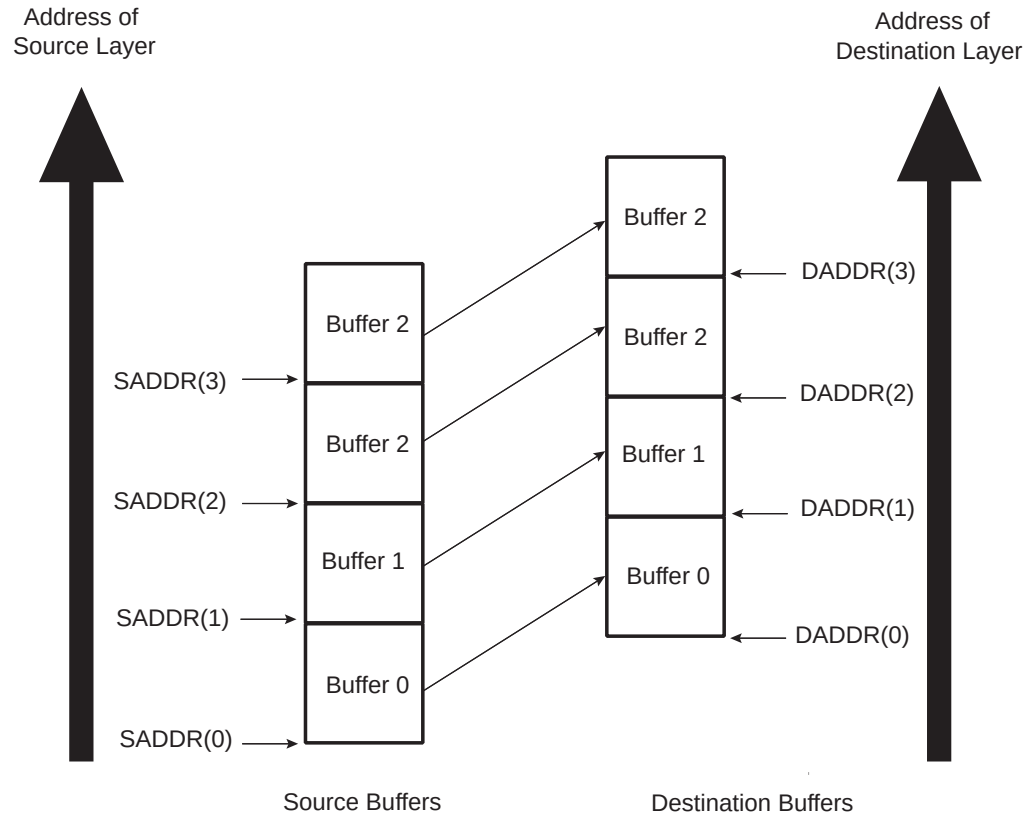
Figure 31-7: Multi-buffer with Linked List Address for Source and Destination



If the user needs to execute a DMAC transfer where the source and destination address are contiguous but the amount of data to be transferred is greater than the maximum buffer size DMAC_CTRLAx.BTSIZE, then this can be achieved using the type of multi-buffer transfer as shown in Figure 31-8.

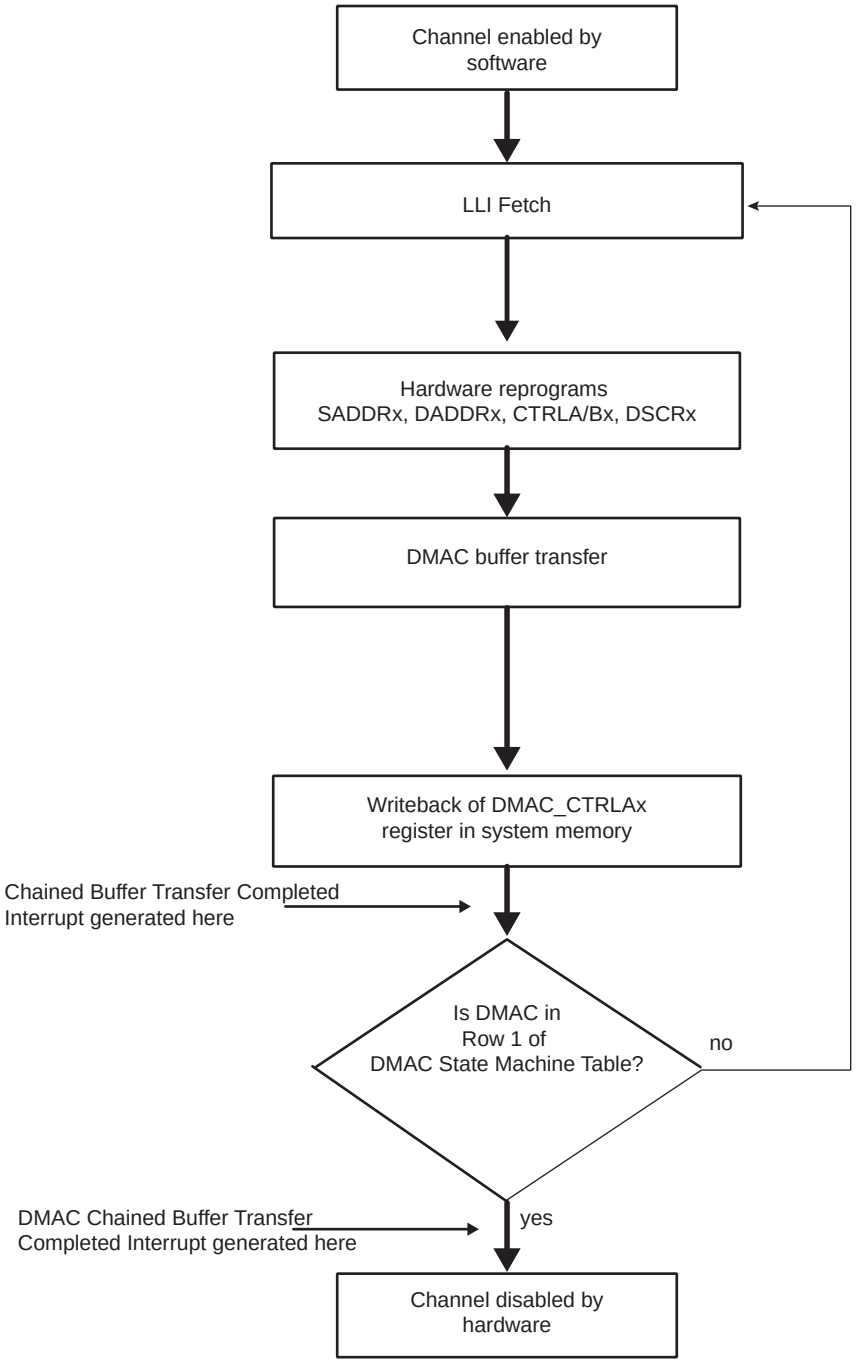
© 2017 Microchip Technology Inc.

Figure 31-8: Multi-buffer with Linked Address for Source and Destination Buffers are Contiguous



The DMAC transfer flow is shown in Figure 31-9.

Figure 31-9: DMAC Transfer Flow for Source and Destination Linked List Address



Multi-buffer Transfer with Source Address Auto-reloaded and Destination Address Auto-reloaded (Row 10)

1. Read the DMAC_CHSR to choose an available (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the interrupt status register. Program the following channel registers:
   a) Write the starting source address in DMAC_SADDRx for channel x.
   b) Write the starting destination address in DMAC_DADDRx for channel x.
   c) Program DMAC_CTRLAx, DMAC_CTRLBx and DMAC_CFGx according to Row 10 as shown in Table 31-3. Program DMAC_DSCRx with 0.

© 2017 Microchip Technology Inc.

d) Write the control information for the DMAC transfer in DMAC_CTRLAx and DMAC_CTRLBx for channel x. For example, in the register, you can program the following:

- i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC field in DMAC_CTRLBx.
- ii. Set up the transfer characteristics, such as:

  Transfer width for the source in the SRC_WIDTH field.

  Transfer width for the destination in the DST_WIDTH field.

  Source AHB master interface layer in the SIF field where source resides.

  Destination AHB master interface layer in the DIF field where destination resides.

  Incrementing/decrementing or fixed address for source in SRC_INCR field.

  Incrementing/decrementing or fixed address for destination in DST_INCR field.

e) If source Picture-in-Picture mode is enabled (DMAC_CTRLBx.SPIP is enabled), program DMAC_SPIPx for channel x.

f) If destination Picture-in-Picture is enabled (DMAC_CTRLBx.DPIP), program DMAC_DPIPx for channel x.

g) Write the channel configuration information into DMAC_CFGx for channel x. Ensure that the reload bits, DMAC_CFGx.SRC_REP, DMAC_CFGx.DST_REP and DMAC_CTRLBx.AUTO are enabled.

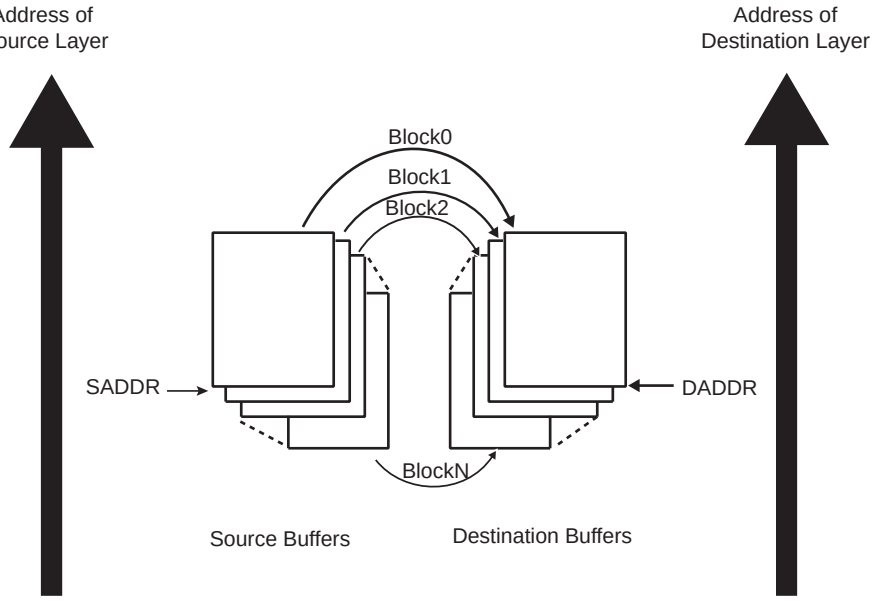
- i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_h2SEL bits, respectively. Writing a ‘1’ activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a ‘0’ activates the software handshaking interface to handle source/destination requests.
- ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

3. After the DMAC selected channel has been programmed, enable the channel by setting the DMAC_CHER.ENAx bit where x is the channel number. Make sure that the ENABLE bit (register bit 0) in DMAC_EN is set.
4. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges on completion of each chunk/single transaction and carries out the buffer transfer.
5. When the buffer transfer has completed, the DMAC reloads DMAC_SADDRx, DMAC_DADDRx and DMAC_CTRLAx. The hardware sets the Buffer Transfer Completed Interrupt. The DMAC then samples the row number as shown in Table 31-3. If the DMAC is in Row 1, then the DMAC transfer has completed. The hardware sets the Chained Buffer Transfer Completed Interrupt and disables the channel. You can either respond to the Buffer Transfer Completed Interrupt or Chained Buffer Transfer Completed Interrupt, or poll for the DMAC_CHSR.ENAx until it is disabled, to detect when the transfer is complete. If the DMAC is not in Row 1, the next step is performed.
6. The DMAC transfer proceeds as follows:

   a) If the Buffer Transfer Completed Interrupt is unmasked (DMAC_EBCIMR.BTCx = ‘1’, where x is the channel number), the hardware sets the Buffer Transfer Completed Interrupt when the buffer transfer has completed. It then stalls until the STALx bit of DMAC_CHSR is cleared by software, writing ‘1’ to DMAC_CHER.KEEPx bit, where x is the channel number. If the next buffer is to be the last buffer in the DMAC transfer, then the buffer complete ISR (interrupt service routine) should clear the automatic mode bit in the DMAC_CTRLBx.AUTO bit. This puts the DMAC into Row 1 as shown in Table 31-3. If the next buffer is not the last buffer in the DMAC transfer, then the reload bits should remain enabled to keep the DMAC in Row 4.

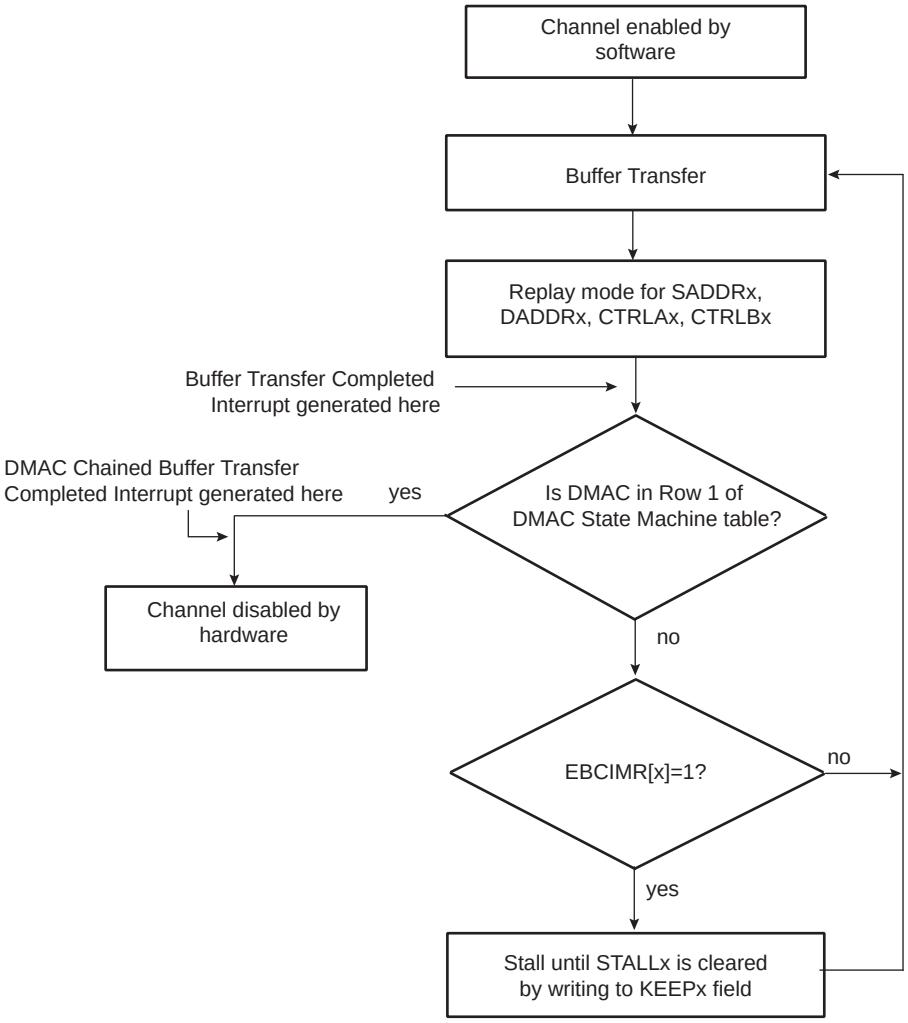
   b) If the Buffer Transfer Completed Interrupt is masked (DMAC_EBCIMR.BTCx = ‘0’, where x is the channel number), the hardware does not stall until it detects a write to the Buffer Transfer Completed Interrupt Enable register DMAC_EBCIER register, but starts the next buffer transfer immediately. In this case, the software must clear the automatic mode bit in the DMAC_CTRLB to put the DMAC into ROW 1 of Table 31-3 before the last buffer of the DMAC transfer has completed. The transfer is similar to that shown in Figure 31-10. The DMAC transfer flow is shown in Figure 31-11.

Figure 31-10: Multi-buffer DMAC Transfer with Source and Destination Address Auto-reloaded



© 2017 Microchip Technology Inc.

Figure 31-11: DMAC Transfer Flow for Source and Destination Address Auto-reloaded



Multi-buffer Transfer with Source Address Auto-reloaded and Linked List Destination Address (Row 6)

1. Read the DMAC_CHSR to choose a free (disabled) channel.
2. Set up the chain of linked list items (otherwise known as buffer descriptors) in memory. Write the control information in the LLI.DMAC_CTRLAx and DMAC_CTRLBx registers location of the buffer descriptor for each LLI in memory for channel x. For example, in the register, you can program the following:
   a) Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control peripheral by programming the FC field in DMAC_CTRLBx.
   b) Set up the transfer characteristics, such as:
   - i. Transfer width for the source in the SRC_WIDTH field.
   - ii. Transfer width for the destination in the DST_WIDTH field.
   - iii. Source AHB master interface layer in the SIF field where source resides.
   - iv. Destination AHB master interface layer in the DIF field where destination resides.
   - v. Incrementing/decrementing or fixed address for source in SRC_INCR field.
   - vi. Incrementing/decrementing or fixed address for destination DST_INCR field.
3. Write the starting source address in DMAC_SADDRx for channel x.

   Note: The values in the LLI.DMAC_SADDRx register locations of each of the Linked List Items (LLIs) set up in memory, although fetched during an LLI fetch, are not used.

4. Write the channel configuration information into DMAC_CFGx for channel x.

a) Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '0' activates the software handshaking interface source/destination requests.

b) If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

5. Make sure that the LLI.DMAC_CTRLBx register locations of all LLIs in memory (except the last one) are set as shown in Row 6 of Table 31-3 while the LLI.DMAC_CTRLBx register of the last Linked List item must be set as described in Row 1 of Table 31-3. Figure 31-5 shows a Linked List example with two list items.
6. Make sure that the LLI.DMAC_DSCRx register locations of all LLIs in memory (except the last one) are non-zero and point to the next Linked List Item.
7. Make sure that the LLI.DMAC_DADDRx register locations of all LLIs in memory point to the start destination buffer address proceeding that LLI fetch.
8. Make sure that the LLI.DMAC_CTLx.DONE bit of the LLI.DMAC_CTRLA register locations of all LLIs in memory is cleared.
9. If source Picture-in-Picture is enabled (DMAC_CTRLBx.SPIP is enabled), program DMAC_SPIPx for channel x.
10. If destination Picture-in-Picture is enabled (DMAC_CTRLBx.DPIP is enabled), program DMAC_DPIPx for channel x.
11. Clear any pending interrupts on the channel from the previous DMAC transfer by reading DMAC_EBCISR.
12. Program DMAC_CTLx and DMAC_CFGx according to Row 6 as shown in Table 31-3.
13. Program DMAC_DSCRx with DMAC_DSCRx(0), the pointer to the first Linked List item.
14. Finally, enable the channel by setting the DMAC_CHER.ENAx bit, where x is the channel number. The transfer is performed. Make sure that the ENABLE bit (register bit 0) in DMAC_EN is set.
15. The DMAC fetches the first LLI from the location pointed to by DMAC_DSCRx(0).

Note: The LLI.DMAC_SADDRx, LLI.DMAC_DADDRx, LLI. DMAC_LLPx LLI.DMAC_CTRLAx and LLI.DMAC_CTRLBx registers are fetched. The LLI.DMAC_SADDRx register, although fetched, is not used.

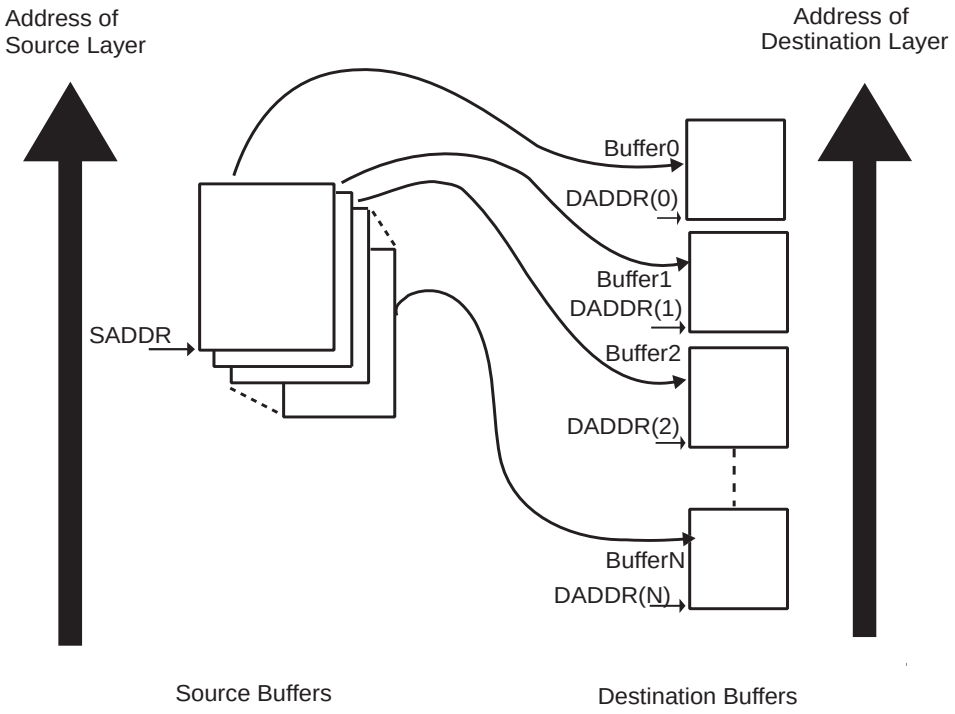
16. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carries out the buffer transfer.
17. The DMAC_CTRLAx register is written out to the system memory. The DMAC_CTRLAx register is written out to the same location on the same layer (DMAC_DSCRx.DSCR_IF) where it was originally fetched, that is the location of the DMAC_CTRLAx register of the linked list item fetched prior to the start of the buffer transfer. Only DMAC_CTRLAx register is written out, because only the DMAC_CTRLAx.BTSIZE and DMAC_CTRLAx.DONE fields have been updated by hardware within the DMAC. The LLI.DMAC_CTRLAx.DONE bit is asserted to indicate buffer completion. Therefore, the software can poll the LLI.DMAC_CTRLAx.DONE bit of the DMAC_CTRLAx register in the LLi to ascertain when a buffer transfer has completed.

Note: Do not poll the DMAC_CTRLAx.DONE bit in the DMAC memory map. Instead, poll the LLI.DMAC_CTRLAx.DONE bit in the LLI for that buffer. If the polled LLI.DMAC_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC_CTRLA.DONE bit was cleared at the start of the transfer.

18. The DMAC reloads DMAC_SADDRx from the initial value. The hardware sets the Buffer Transfer Completed Interrupt. The DMAC samples the row number as shown in Table 31-3. If the DMAC is in Row 1, then the DMAC transfer has completed. The hardware sets the Chained Buffer Transfer Completed Interrupt and disables the channel. You can either respond to the Buffer Transfer Completed Interrupt or Chained Buffer Transfer Completed Interrupt, or poll for the DMAC_CHSR.ENAx bit until it is cleared by hardware, to detect when the transfer is complete. If the DMAC is not in Row 1 as shown in Table 31-3, the following step is performed.
19. The DMAC fetches the next LLI from the memory location pointed to by the current DMAC_DSCRx register, and automatically reprograms the DMAC_DADDRx, DMAC_CTRLAx, DMAC_CTRLBx and DMAC_DSCRx channel registers. Note that the DMAC_SADDRx is not re-programmed as the reloaded value is used for the next DMAC buffer transfer. If the next buffer is the last buffer of the DMAC transfer, then the DMAC_CTRLBx and DMAC_DSCRx registers just fetched from the LLI should match Row 1 of Table 31-3. The DMAC transfer might look like that shown in Figure 31-12.

© 2017 Microchip Technology Inc.

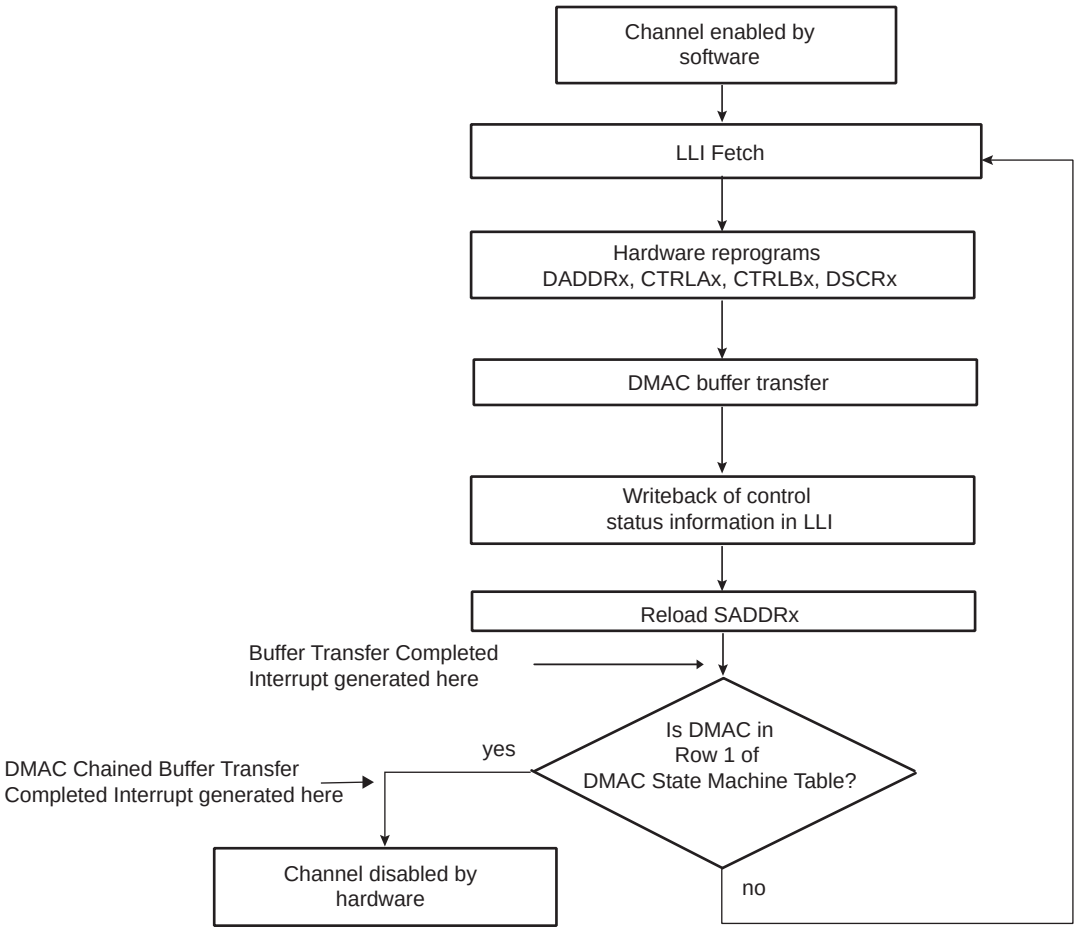
Figure 31-12: Multi-buffer DMAC Transfer with Source Address Auto-reloaded and Linked List Destination



Address

The DMAC Transfer flow is shown in Figure 31-13.

Figure 31-13: DMAC Transfer Flow for Replay Mode at Source and Linked List Destination Address



Multi-buffer Transfer with Source Address Auto-reloaded and Contiguous Destination Address (Row 11)

1. Read the DMAC_CHSR to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading to the Interrupt Status Register.
3. Program the following channel registers:
   a) Write the starting source address in DMAC_SADDRx for channel x.
   b) Write the starting destination address in DMAC_DADDRx for channel x.
   c) Program DMAC_CTRLAx, DMAC_CTRLBx and DMAC_CFGx according to Row 11 as shown in Table 31-3. Program DMAC_DSCRx with ‘0’. DMAC_CTRLBx.AUTO bit is set to ‘1’ to enable automatic mode support.
   d) Write the control information for the DMAC transfer in DMAC_CTRLBx and DMAC_CTRLAx for channel x. For example, in this register, you can program the following:

- i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC field in DMAC_CTRLBx.
- ii. Set up the transfer characteristics, such as:

  Transfer width for the source in the SRC_WIDTH field.

  Transfer width for the destination in the DST_WIDTH field.

  Source AHB master interface layer in the SIF field where source resides.

  Destination AHB master interface master layer in the DIF field where destination resides.

  Incrementing/decrementing or fixed address for source in SRC_INCR field.

  Incrementing/decrementing or fixed address for destination in DST_INCR field.

   e) If source Picture-in-Picture is enabled (DMAC_CTRLBx.SPIP is enabled), program DMAC_SPIPx for channel x.

© 2017 Microchip Technology Inc.

f) If destination Picture-in-Picture is enabled (DMAC_CTRLBx.DPIP), program DMAC_DPIPx for channel x.

g) Write the channel configuration information into DMAC_CFGx for channel x.

- i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '0' activates the software handshaking interface to handle source/destination requests.
- ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

4. After the DMAC channel has been programmed, enable the channel by setting the DMAC_CHER.ENAx bit, where x is the channel number. Make sure that the ENABLE bit (register bit 0) in DMAC_EN is set.
5. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carries out the buffer transfer.
6. When the buffer transfer has completed, the DMAC reloads DMAC_SADDRx. DMAC_DADDRx remains unchanged. The hardware sets the Buffer Transfer Completed Interrupt. The DMAC then samples the row number as shown in Table 31-3. If the DMAC is in Row 1, then the DMAC transfer has completed. The hardware sets the Chained Buffer Transfer Completed Interrupt and disables the channel. You can either respond to the Buffer Transfer Completed Interrupt or Chained Buffer Transfer Completed Interrupt, or poll for the DMAC_CHSR.ENAx bit until it is cleared by hardware, to detect when the transfer is complete. If the DMAC is not in Row 1, the next step is performed.
7. The DMAC transfer proceeds as follows:
   a) If the Buffer Transfer Completed Interrupt is unmasked (DMAC_EBCIMR.BTCx = '1', where x is the channel number), the hardware sets the Buffer Transfer Completed Interrupt when the buffer transfer has completed. It then stalls until STALx bit of DMAC_CHSR is cleared by writing in the KEEPx bit of DMAC_CHER, where x is the channel number. If the next buffer is to be the last buffer in the DMAC transfer, then the buffer complete ISR (interrupt service routine) should clear the automatic mode bit, DMAC_CTRLBx.AUTO. This puts the DMAC into Row 1 as shown in Table 31-3. If the next buffer is not the last buffer in the DMAC transfer, then the automatic transfer mode bit should remain enabled to keep the DMAC in Row 11 as shown in Table 31-3.
   b) If the Buffer Transfer Completed Interrupt is masked (DMAC_EBCIMR.BTCx = '0', where x is the channel number), the hardware does not stall until it detects a write to the Buffer Transfer Completed Interrupt Enable register, but starts the next buffer transfer immediately. In this case, the software must clear the automatic mode bit, DMAC_CTRLBx.AUTO, to put the device into ROW 1 of Table 31-3 before the last buffer of the DMAC transfer has completed.

The transfer is similar to that shown in Figure 31-14.

The DMAC Transfer flow is shown in Figure 31-15.

Figure 31-14: Multi-buffer Transfer with Source Address Auto-reloaded and Contiguous Destination Address

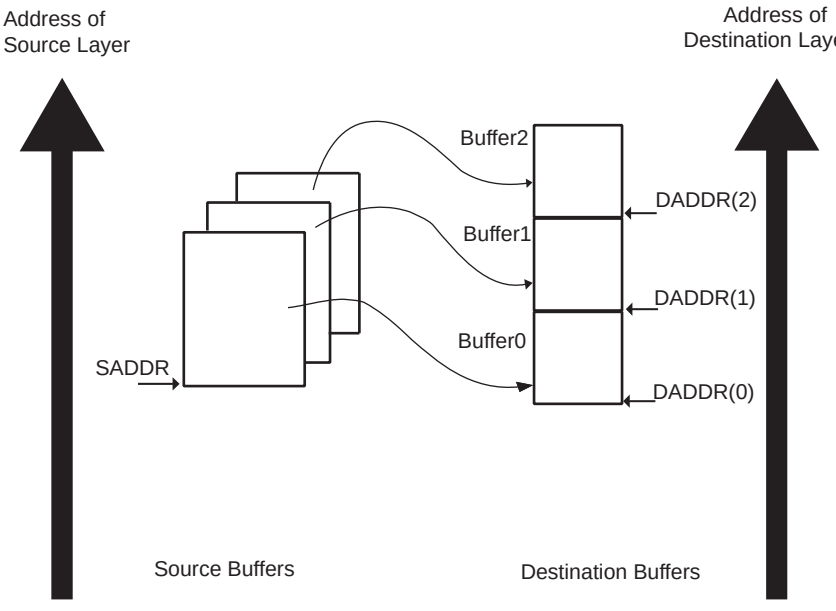
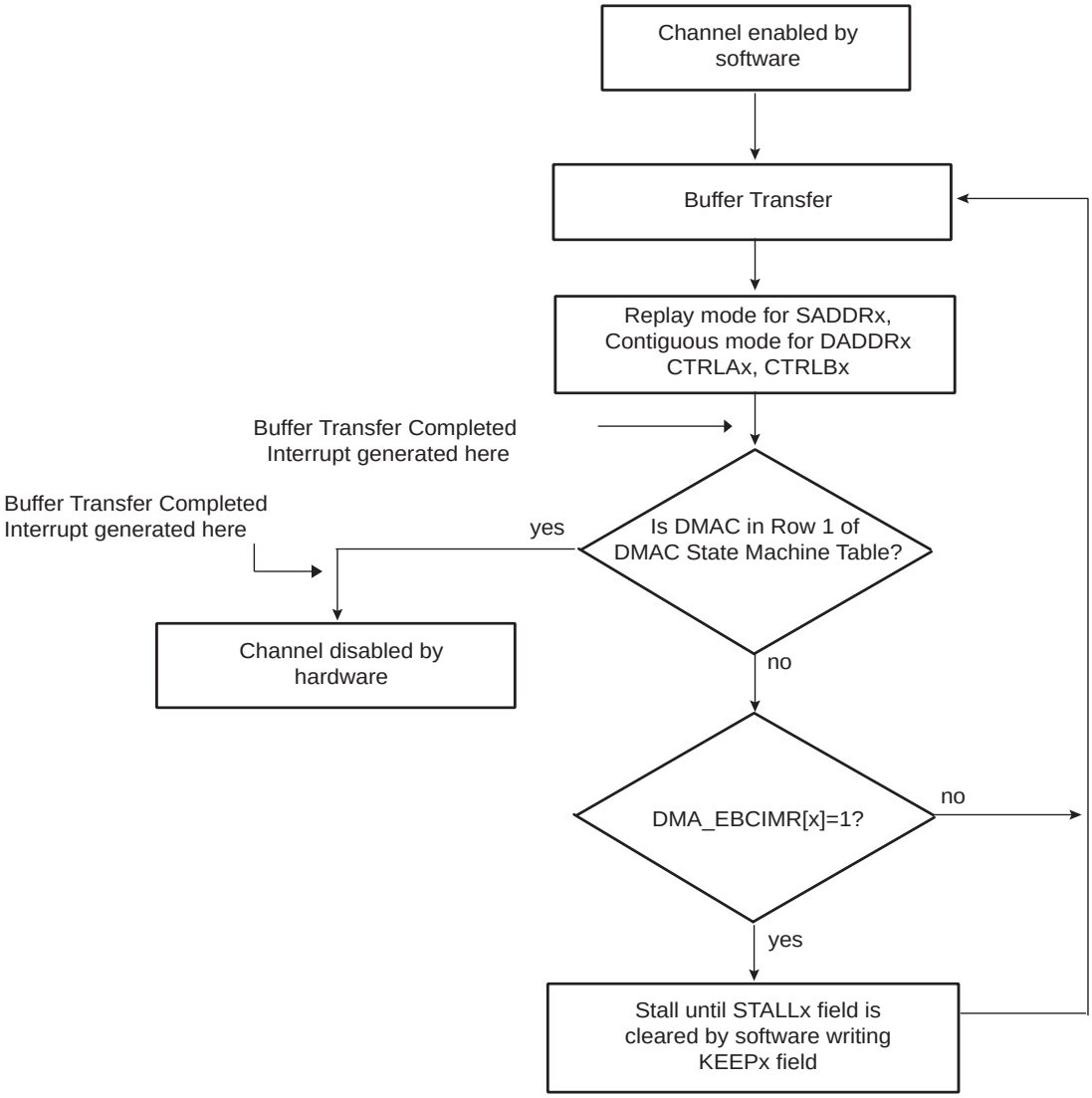


Figure 31-15: DMAC Transfer Replay Mode is Enabled for the Source and Contiguous Destination Address



Multi-buffer DMAC Transfer with Linked List for Source and Contiguous Destination Address (Row 2)

1. Read the DMAC_CHSR to choose a free (disabled) channel.
2. Set up the linked list in memory. Write the control information in the LLI.DMAC_CTRLAx and LLI.DMAC_CTRLBx register location of the buffer descriptor for each LLI in memory for channel x. For example, in the register, you can program the following:
   a) Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC field in DMAC_CTRLBx.
   b) Set up the transfer characteristics, such as:
   - i. Transfer width for the source in the SRC_WIDTH field.
   - ii. Transfer width for the destination in the DST_WIDTH field.
   - iii. Source AHB master interface layer in the SIF field where source resides.
   - iv. Destination AHB master interface layer in the DIF field where destination resides.
   - v. Incrementing/decrementing or fixed address for source in SRC_INCR field.
   - vi. Incrementing/decrementing or fixed address for destination DST_INCR field.
3. Write the starting destination address in DMAC_DADDRx for channel x.

Note: The values in the LLI.DMAC_DADDRx register location of each Linked List Item (LLI) in memory, although fetched during an LLI fetch, are not used.

© 2017 Microchip Technology Inc.

4. Write the channel configuration information into DMAC_CFGx for channel x.
   a) Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '0' activates the software handshaking interface to handle source/destination requests.
   b) If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripherals. This requires programming the SRC_PER and DST_PER bits, respectively.
5. Make sure that all LLI.DMAC_CTRLBx register locations of the LLI (except the last) are set as shown in Row 2 of Table 31-3, while the LLI.DMAC_CTRLBx register of the last Linked List item must be set as described in Row 1 of Table 31-3. Figure 31-5 shows a Linked List example with two list items.
6. Make sure that the LLI.DMAC_DSCRx register locations of all LLIs in memory (except the last) are non-zero and point to the next Linked List Item.
7. Make sure that the LLI.DMAC_SADDRx register locations of all LLIs in memory point to the start source buffer address proceeding that LLI fetch.
8. Make sure that the LLI.DMAC_CTRLAx.DONE bit of the LLI.DMAC_CTRLAx register locations of all LLIs in memory is cleared.
9. If source Picture-in-Picture is enabled (DMAC_CTRLBx.SPIP is enabled), program DMAC_SPIPx for channel x.
10. If destination Picture-in-Picture is enabled (DMAC_CTRLBx.DPIP is enabled), program DMAC_DPIPx for channel x.
11. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the interrupt status register.
12. Program DMAC_CTRLAx, DMAC_CTRLBx and DMAC_CFGx according to Row 2 as shown in Table 31-3
13. Program DMAC_DSCRx with DMAC_DSCRx(0), the pointer to the first Linked List item.
14. Finally, enable the channel by setting the DMAC_CHER.ENAx bit. The transfer is performed. Make sure that the ENABLE bit (register bit 0) in DMAC_EN is set.
15. The DMAC fetches the first LLI from the location pointed to by DMAC_DSCRx(0).

    Note: The LLI.DMAC_SADDRx, LLI.DMAC_DADDRx, LLI.DMAC_DSCRx and LLI.DMAC_CTRLA/Bx registers are fetched. The LLI.DMAC_DADDRx register location of the LLI, although fetched, is not used. The DMAC_DADDRx register in the DMAC remains unchanged.

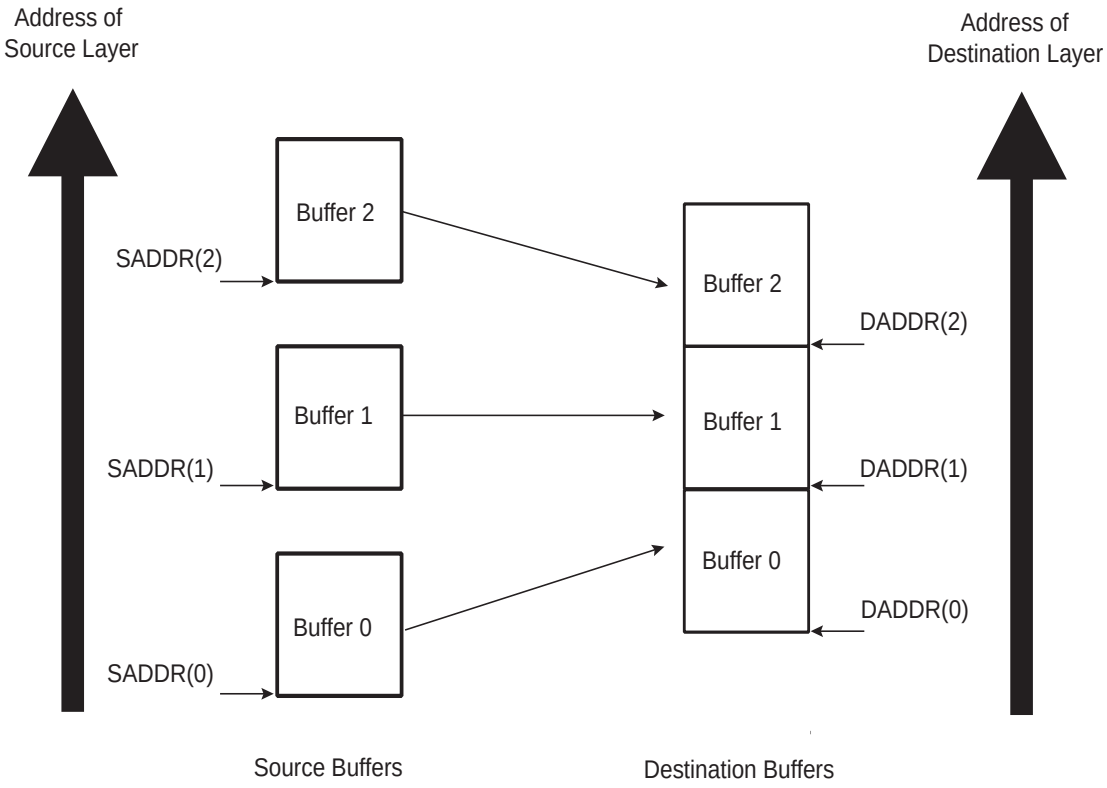
16. Source and destination requests single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carries out the buffer transfer.
17. Once the buffer of data is transferred, the DMAC_CTRLAx register is written out to the system memory at the same location and on the same layer (DMAC_DSCRx.DSCR_IF) where it was originally fetched, that is, the location of the DMAC_CTRLAx register of the linked list item fetched prior to the start of the buffer transfer. Only DMAC_CTRLAx register is written out because only the DMAC_CTRLAx.BTSIZE and DMAC_CTRLAX.DONE fields have been updated by DMAC hardware. Additionally, the DMAC_CTRLAx.DONE bit is asserted when the buffer transfer has completed.

    Note: Do not poll the DMAC_CTRLAx.DONE bit in the DMAC memory map. Instead, poll the LLI.DMAC_CTRLAx.DONE bit in the LLI for that buffer. If the poll LLI.DMAC_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC_CTRLAx.DONE bit was cleared at the start of the transfer.

18. The DMAC does not wait for the buffer interrupt to be cleared, but continues and fetches the next LLI from the memory location pointed to by the current DMAC_DSCRx register, then automatically reprograms the DMAC_SADDRx, DMAC_CTRLAx, DMAC_CTRLBx and DMAC_DSCRx channel registers. DMAC_DADDRx is left unchanged. The DMAC transfer continues until the DMAC samples the DMAC_CTRLAx, DMAC_CTRLBx and DMAC_DSCRx registers at the end of a buffer transfer match that described in Row 1 of Table 31-3. The DMAC then knows that the previous buffer transferred was the last buffer in the DMAC transfer.

The DMAC transfer might look like that shown in Figure 31-16. Note that the destination address is decrementing.

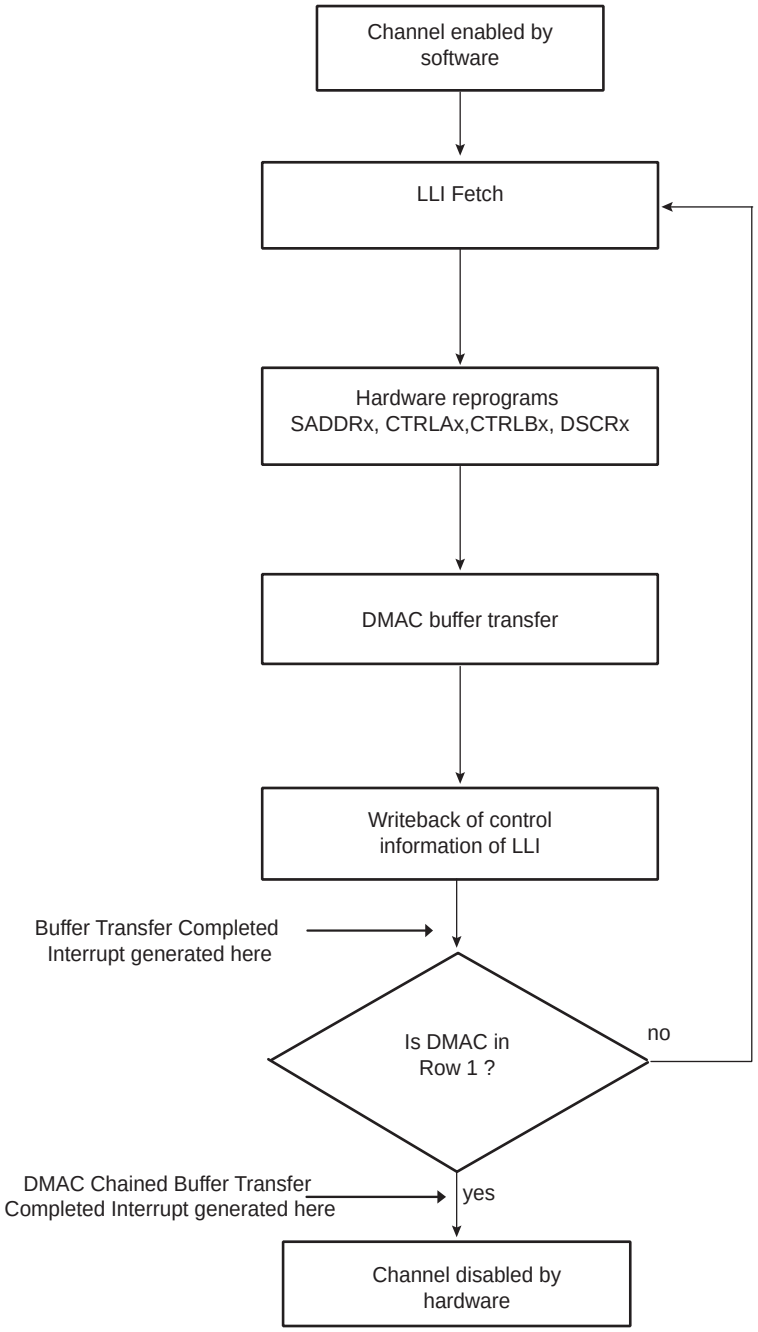
Figure 31-16: DMAC Transfer with Linked List Source Address and Contiguous Destination Address



The DMAC transfer flow is shown in Figure 31-17.

© 2017 Microchip Technology Inc.

Figure 31-17: DMAC Transfer Flow for Linked List Source Address and Contiguous Destination Address



31.6.6 Disabling a Channel Prior to Transfer Completion

Under normal operation, the software enables a channel by setting the DMAC_CHER.ENAx bit, and the hardware disables a channel on transfer completion by clearing the DMAC_CHSR.ENAx bit.

The recommended way for software to disable a channel without losing data is to use the SUSPx bit in conjunction with the EMPTx bit in the DMAC_CHSR.

1. If the software chooses to disable a channel n prior to the DMAC transfer completion, then it can set the DMAC_CHER.SUSPx bit to instruct the DMAC to halt all transfers from the source peripheral. Therefore, the channel FIFO receives no new data.
2. The software can now poll the DMAC_CHSR.EMPTx bit until it indicates that the channel n FIFO is empty, where n is the channel number.

3. The DMAC_CHER.ENAx bit can then be cleared by software once the channel n FIFO is empty, where n is the channel number.

When DMAC_CTRLAx.SRC_WIDTH is less than DMAC_CTRLAx.DST_WIDTH and the DMAC_CHSRx.SUSPx bit is high, the DMAC_CHSRx.EMPTx is asserted once the contents of the FIFO does not permit a single word of DMAC_CTRLAx.DST_WIDTH to be formed. However, there may still be data in the channel FIFO but not enough to form a single transfer of DMAC_CTLx.DST_WIDTH width. In this configuration, once the channel is disabled, the remaining data in the channel FIFO are not transferred to the destination peripheral. It is permitted to remove the channel from the suspension state by by setting the DMAC_CHDR.RESx bit. The DMAC transfer completes in the normal manner. n defines the channel number.

Note: If a channel is disabled by software, an active single or chunk transaction is not guaranteed to receive an acknowledgement.

31.6.6.1 Abnormal Transfer Termination

A DMAC transfer may be terminated abruptly by software by clearing the channel enable bit, DMAC_CHER.ENAx, where x is the channel number. This does not mean that the channel is disabled immediately after the DMAC_CHSR.ENAx bit is cleared over the APB interface. Consider this as a request to disable the channel. The DMAC_CHSR.ENAx must be polled and then it must be confirmed that the channel is disabled by reading back 0.

The software may terminate all channels abruptly by clearing the general enable bit in the DMAC Enable Register (DMAC_EN.ENABLE). Again, this does not mean that all channels are disabled immediately after the DMAC_EN.ENABLE bit is cleared over the APB slave interface. Consider this as a request to disable all channels. The DMAC_CHSR.ENABLE must be polled and then it must be confirmed that all channels are disabled by reading back ‘0’.

Note: If the channel enable bit is cleared while there is data in the channel FIFO, this data is not sent to the destination peripheral and is not present when the channel is re-enabled. For read sensitive source peripherals, such as a source FIFO, this data is therefore lost. When the source is not a read sensitive device (i.e., memory), disabling a channel without waiting for the channel FIFO to empty may be acceptable as the data is available from the source peripheral upon request and is not lost.

Note: If a channel is disabled by software, an active single or chunk transaction is not guaranteed to receive an acknowledgement.

31.6.7 Register Write Protection

To prevent any single software error from corrupting DMAC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the “DMAC Write Protection Mode Register” (DMAC_WPMR).

If a write access to a write-protected register is detected, the WPVS bit in the “DMAC Write Protection Status Register” (DMAC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the DMAC_WPSR.

The following registers can be write-protected:

- “DMAC Global Configuration Register”
- “DMAC Enable Register”
- “DMAC Channel x [x = 0..7] Source Address Register”
- “DMAC Channel x [x = 0..7] Destination Address Register”
- “DMAC Channel x [x = 0..7] Descriptor Address Register”
- “DMAC Channel x [x = 0..7] Control A Register”
- “DMAC Channel x [x = 0..7] Control B Register”
- “DMAC Channel x [x = 0..7] Configuration Register”

31.7 DMAC Software Requirements

- There must not be any write operation to channel registers in an active channel after the channel enable is made HIGH. If any channel parameters must be reprogrammed, this can only be done after disabling the DMAC channel.
- The channel registers DMAC_SADDRx and DMAC_DADDRx must be programmed with a byte, half-word and word aligned address depending on the source width and destination width.
- After the software disables a channel by writing into the DMAC Channel Handler Disable Register, it must re-enable the channel only after it has polled a ‘0’ in the DMAC Channel Handler Status Register. This is because the current AHB Burst must terminate properly.
- If the value of field DMAC_CTRLAx.BTSIZE is configured to zero and the DMAC has been defined as the flow controller, the channel is automatically disabled.

© 2017 Microchip Technology Inc.

- Multiple transfers involving the same peripheral must not be programmed and enabled on different channels, unless this peripheral integrates several hardware handshaking interfaces.
- When a peripheral has been defined as the flow controller, the targeted DMAC channel must be enabled before the peripheral. If you do not ensure this and the first DMAC request is also the last transfer, the DMAC channel might miss a Last Transfer Flag.
- When the AUTO bit is set to TRUE, the BTSIZE field is automatically reloaded from its previous value. BTSIZE must be initialized to a non zero value if the first transfer is initiated with the AUTO bit set to TRUE, even if LLI mode is enabled, because the LLI fetch operation will not update this field.

31.8 DMA Controller (DMAC) User Interface

Table 31-4: Register Mapping

Offset	Register	Name	Access	Reset
0x000	DMAC Global Configuration Register	DMAC_GCFG	Read/Write	0x10
0x004	DMAC Enable Register	DMAC_EN	Read/Write	0x0
0x008	DMAC Software Single Request Register	DMAC_SREQ	Read/Write	0x0
0x00C	DMAC Software Chunk Transfer Request Register	DMAC_CREQ	Read/Write	0x0
0x010	DMAC Software Last Transfer Flag Register	DMAC_LAST	Read/Write	0x0
0x014	Reserved	–	–	–
0x018	DMAC Error, Chained Buffer Transfer Completed Interrupt and Buffer Transfer Completed Interrupt Enable Register	DMAC_EBCIER	Write-only	–
0x01C	DMAC Error, Chained Buffer Transfer Completed Interrupt and Buffer Transfer Completed Interrupt Disable Register	DMAC_EBCIDR	Write-only	–
0x020	DMAC Error, Chained Buffer Transfer Completed Interrupt and Buffer transfer completed Mask Register	DMAC_EBCIMR	Read-only	0x0
0x024	DMAC Error, Chained Buffer Transfer Completed Interrupt and Buffer transfer completed Status Register	DMAC_EBCISR	Read-only	0x0
0x028	DMAC Channel Handler Enable Register	DMAC_CHER	Write-only	–
0x02C	DMAC Channel Handler Disable Register	DMAC_CHDR	Write-only	–
0x030	DMAC Channel Handler Status Register	DMAC_CHSR	Read-only	0x00FF0000
0x034–0x038	Reserved	–	–	–
0x03C+ch_num*(0x28)+(0x0)	DMAC Channel Source Address Register	DMAC_SADDR	Read/Write	0x0
0x03C+ch_num*(0x28)+(0x4)	DMAC Channel Destination Address Register	DMAC_DADDR	Read/Write	0x0
0x03C+ch_num*(0x28)+(0x8)	DMAC Channel Descriptor Address Register	DMAC_DSCR	Read/Write	0x0
0x03C+ch_num*(0x28)+(0xC)	DMAC Channel Control A Register	DMAC_CTRLA	Read/Write	0x0
0x03C+ch_num*(0x28)+(0x10)	DMAC Channel Control B Register	DMAC_CTRLB	Read/Write	0x0
0x03C+ch_num*(0x28)+(0x14)	DMAC Channel Configuration Register	DMAC_CFG	Read/Write	0x01000000
0x03C+ch_num*(0x28)+(0x18)	DMAC Channel Source Picture-in-Picture Configuration Register	DMAC_SPIP	Read/Write	0x0
0x03C+ch_num*(0x28)+(0x1C)	DMAC Channel Destination Picture-in-Picture Configuration Register	DMAC_DPIP	Read/Write	0x0
0x03C+ch_num*(0x28)+(0x20)	Reserved	–	–	–
0x03C+ch_num*(0x28)+(0x24)	Reserved	–	–	–
0x1E4	DMAC Write Protection Mode Register	DMAC_WPMR	Read/Write	0x0
0x1E8	DMAC Write Protection Status Register	DMAC_WPSR	Read-only	0x0
0x1EC–0x1FC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

31.8.1 DMAC Global Configuration Register

Name: DMAC_GCFG

Address:0xFFFFEC00

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	DICEN
7	6	5	4	3	2	1	0
–	–	–	ARB_CFG	–	–	–	–

Note: Bit fields 0, 1, 2, and 3 have a default value of 0. This should not be changed.

This register can only be written if the WPEN bit is cleared in “DMAC Write Protection Mode Register”.

ARB_CFG: Arbiter Configuration

Value	Name	Description
0	FIXED	Fixed priority arbiter (see “Basic Definitions”)
1	ROUND_ROBIN	Modified round robin arbiter.

DICEN: Descriptor Integrity Check

0: Descriptor Integrity Check Interface is Disabled.

1: Descriptor Integrity Check Interface is Enabled.

© 2017 Microchip Technology Inc.

31.8.2 DMAC Enable Register

Name: DMAC_EN

Address:0xFFFFEC04

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	ENABLE

This register can only be written if the WPEN bit is cleared in “DMAC Write Protection Mode Register”.

ENABLE: General Enable of DMA

0: DMA Controller is disabled.

1: DMA Controller is enabled.

© 2017 Microchip Technology Inc.

31.8.3 DMAC Software Single Request Register

Name:DMAC_SREQ

Address:0xFFFFEC08

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
DSREQ7	SSREQ7	DSREQ6	SSREQ6	DSREQ5	SSREQ5	DSREQ4	SSREQ4
7	6	5	4	3	2	1	0
DSREQ3	SSREQ3	DSREQ2	SSREQ2	DSREQ1	SSREQ1	DSREQ0	SSREQ0

DSREQx: Destination Request

Request a destination single transfer on channel i.

SSREQx: Source Request

Request a source single transfer on channel i.

31.8.4 DMAC Software Chunk Transfer Request Register

Name:DMAC_CREQ

Address:0xFFFFEC0C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
DCREQ7	SCREQ7	DCREQ6	SCREQ6	DCREQ5	SCREQ5	DCREQ4	SCREQ4
7	6	5	4	3	2	1	0
DCREQ3	SCREQ3	DCREQ2	SCREQ2	DCREQ1	SCREQ1	DCREQ0	SCREQ0

DCREQx: Destination Chunk Request

Request a destination chunk transfer on channel i.

SCREQx: Source Chunk Request

Request a source chunk transfer on channel i.

© 2017 Microchip Technology Inc.

31.8.5 DMAC Software Last Transfer Flag Register

Name:DMAC_LAST

Address:0xFFFFEC10

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
DLAST7	SLAST7	DLAST6	SLAST6	DLAST5	SLAST5	DLAST4	SLAST4

7	6	5	4	3	2	1	0
DLAST3	SLAST3	DLAST2	SLAST2	DLAST1	SLAST1	DLAST0	SLAST0

DLASTx: Destination Last

Writing one to DLASTx prior to writing one to DSREQx or DCREQx indicates that this destination request is the last transfer of the buffer.

SLASTx: Source Last

Writing one to SLASTx prior to writing one to SSREQx or SCREQx indicates that this source request is the last transfer of the buffer.

31.8.6 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Enable Register

Name: DMAC_EBCIER

Address:0xFFFFEC18

Access: Write-only

31	30	29	28	27	26	25	24
DICERR7	DICERR6	DICERR5	DICERR4	DICERR3	DICERR2	DICERR1	DICERR0
23	22	21	20	19	18	17	16
ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
15	14	13	12	11	10	9	8
CBTC7	CBTC6	CBTC5	CBTC4	CBTC3	CBTC2	CBTC1	CBTC0
7	6	5	4	3	2	1	0
BTC7	BTC6	BTC5	BTC4	BTC3	BTC2	BTC1	BTC0

BTCx: Buffer Transfer Completed [7:0]

Buffer Transfer Completed Interrupt Enable Register. Set the relevant bit in the BTC field to enable the interrupt for channel i.

CBTCx: Chained Buffer Transfer Completed [7:0]

Chained Buffer Transfer Completed Interrupt Enable Register. Set the relevant bit in the CBTC field to enable the interrupt for channel i.

ERRx: Access Error [7:0]

Access Error Interrupt Enable Register. Set the relevant bit in the ERR field to enable the interrupt for channel i.

DICERRx: Descriptor Integrity Check Error [7:0]

Descriptor Integrity Check Error Interrupt Enable Register. Set the relevant bit in the DICERR field to enable the interrupt for channel i.

© 2017 Microchip Technology Inc.

31.8.7 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Disable Register

Name: DMAC_EBCIDR

Address:0xFFFFEC1C

Access: Write-only

31	30	29	28	27	26	25	24
DICERR7	DICERR6	DICERR5	DICERR4	DICERR3	DICERR2	DICERR1	DICERR0
23	22	21	20	19	18	17	16
ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
15	14	13	12	11	10	9	8
CBTC7	CBTC6	CBTC5	CBTC4	CBTC3	CBTC2	CBTC1	CBTC0
7	6	5	4	3	2	1	0
BTC7	BTC6	BTC5	BTC4	BTC3	BTC2	BTC1	BTC0

BTCx: Buffer Transfer Completed [7:0]

Buffer transfer completed Disable Interrupt Register. When set, a bit of the BTC field disables the interrupt from the relevant DMAC channel.

CBTCx: Chained Buffer Transfer Completed [7:0]

Chained Buffer transfer completed Disable Register. When set, a bit of the CBTC field disables the interrupt from the relevant DMAC channel.

ERRx: Access Error [7:0]

Access Error Interrupt Disable Register. When set, a bit of the ERR field disables the interrupt from the relevant DMAC channel.

DICERRx: Descriptor Integrity Check Error [7:0]

Descriptor Integrity Check Error Interrupt Disable Register, When set, a bit of the DICERR field disables the interrupt from the relevant DMAC channel.

31.8.8 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Mask Register

Name:DMAC_EBCIMR

Address:0xFFFFEC20

Access: Read-only

31	30	29	28	27	26	25	24
DICERR7	DICERR6	DICERR5	DICERR4	DICERR3	DICERR2	DICERR1	DICERR0
23	22	21	20	19	18	17	16
ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
15	14	13	12	11	10	9	8
CBTC7	CBTC6	CBTC5	CBTC4	CBTC3	CBTC2	CBTC1	CBTC0
7	6	5	4	3	2	1	0
BTC7	BTC6	BTC5	BTC4	BTC3	BTC2	BTC1	BTC0

BTCx: Buffer Transfer Completed [7:0]

0: Buffer Transfer Completed Interrupt is disabled for channel i.

1: Buffer Transfer Completed Interrupt is enabled for channel i.

CBTCx: Chained Buffer Transfer Completed [7:0]

0: Chained Buffer Transfer interrupt is disabled for channel i.

1: Chained Buffer Transfer interrupt is enabled for channel i.

ERRx: Access Error [7:0]

0: Transfer Error Interrupt is disabled for channel i.

1: Transfer Error Interrupt is enabled for channel i.

DICERRx: Descriptor Integrity Check Error [7:0]

0: Descriptor Integrity Check Error Interrupt is disabled for channel i.

1: Descriptor Integrity Check Error Interrupt is enabled for channel i.

© 2017 Microchip Technology Inc.

31.8.9 DMAC Error, Buffer Transfer and Chained Buffer Transfer Status Register

Name:DMAC_EBCISR

Address:0xFFFFEC24

Access: Read-only

31	30	29	28	27	26	25	24
DICERR7	DICERR6	DICERR5	DICERR4	DICERR3	DICERR2	DICERR1	DICERR0
23	22	21	20	19	18	17	16
ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
15	14	13	12	11	10	9	8
CBTC7	CBTC6	CBTC5	CBTC4	CBTC3	CBTC2	CBTC1	CBTC0
7	6	5	4	3	2	1	0
BTC7	BTC6	BTC5	BTC4	BTC3	BTC2	BTC1	BTC0

BTCx: Buffer Transfer Completed [7:0]

When BTC[*i*] is set, Channel *i* buffer transfer has terminated.

CBTCx: Chained Buffer Transfer Completed [7:0]

When CBTC[*i*] is set, Channel *i* Chained buffer has terminated. LLI Fetch operation is disabled.

ERRx: Access Error [7:0]

When ERR[*i*] is set, Channel *i* has detected an AHB Read or Write Error Access.

DICERRx: Descriptor Integrity Check Error [7:0]

When DICERR[*i*] is set, Channel *i* has detected a Descriptor Integrity Check Error.

31.8.10 DMAC Channel Handler Enable Register

Name: DMAC_CHER

Address:0xFFFFEC28

Access: Write-only

31	30	29	28	27	26	25	24
KEEP7	KEEP6	KEEP5	KEEP4	KEEP3	KEEP2	KEEP1	KEEP0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
SUSP7	SUSP6	SUSP5	SUSP4	SUSP3	SUSP2	SUSP1	SUSP0
7	6	5	4	3	2	1	0
ENA7	ENA6	ENA5	ENA4	ENA3	ENA2	ENA1	ENA0

ENAx: Enable [7:0]

When set, a bit of the ENA field enables the relevant channel.

SUSPx: Suspend [7:0]

When set, a bit of the SUSP field freezes the relevant channel and its current context.

KEEPx: Keep on [7:0]

When set, a bit of the KEEP field resumes the current channel from an automatic stall state.

© 2017 Microchip Technology Inc.

31.8.11 DMAC Channel Handler Disable Register

Name: DMAC_CHDR

Address:0xFFFFEC2C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RES7	RES6	RES5	RES4	RES3	RES2	RES1	RES0
7	6	5	4	3	2	1	0
DIS7	DIS6	DIS5	DIS4	DIS3	DIS2	DIS1	DIS0

DISx: Disable [7:0]

Write one to this field to disable the relevant DMAC Channel. The content of the FIFO is lost and the current AHB access is terminated. Software must poll DIS[7:0] field in the DMAC_CHSR register to be sure that the channel is disabled.

RESx: Resume [7:0]

Write one to this field to resume the channel transfer restoring its context.

31.8.12 DMAC Channel Handler Status Register

Name:DMAC_CHSR

Address:0xFFFFEC30

Access: Read-only

31	30	29	28	27	26	25	24
STAL7	STAL6	STAL5	STAL4	STAL3	STAL2	STAL1	STAL0
23	22	21	20	19	18	17	16
EMPT7	EMPT6	EMPT5	EMPT4	EMPT3	EMPT2	EMPT1	EMPT0
15	14	13	12	11	10	9	8
SUSP7	SUSP6	SUSP5	SUSP4	SUSP3	SUSP2	SUSP1	SUSP0
7	6	5	4	3	2	1	0
ENA7	ENA6	ENA5	ENA4	ENA3	ENA2	ENA1	ENA0

ENAx: Enable [7:0]

A one in any position of this field indicates that the relevant channel is enabled.

SUSPx: Suspend [7:0]

A one in any position of this field indicates that the channel transfer is suspended.

EMPTx: Empty [7:0]

A one in any position of this field indicates that the relevant channel is empty.

STALx: Stalled [7:0]

A one in any position of this field indicates that the relevant channel is stalling.

© 2017 Microchip Technology Inc.

31.8.13 DMAC Channel x [x = 0..7] Source Address Register

Name:DMAC_SADDRx [x = 0..7]

Address:0xFFFFEC3C [0], 0xFFFFEC64 [1], 0xFFFFEC8C [2], 0xFFFFECB4 [3], 0xFFFFECDC [4], 0xFFFFED04 [5], 0xFFFFED2C [6], 0xFFFFED54 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
SADDR							

23	22	21	20	19	18	17	16
SADDR							

15	14	13	12	11	10	9	8
SADDR							

7	6	5	4	3	2	1	0
SADDR							

This register can only be written if the WPEN bit is cleared in “DMAC Write Protection Mode Register”.

SADDR: Channel x Source Address

This register must be aligned with the source transfer width.

© 2017 Microchip Technology Inc.

31.8.14 DMAC Channel x [x = 0..7] Destination Address Register

Name:DMAC_DADDRx [x = 0..7]

Address:0xFFFFEC40 [0], 0xFFFFEC68 [1], 0xFFFFEC90 [2], 0xFFFFECB8 [3], 0xFFFFECE0 [4], 0xFFFFED08 [5], 0xFFFFED30 [6], 0xFFFFED58 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
DADDR							

23	22	21	20	19	18	17	16
DADDR							

15	14	13	12	11	10	9	8
DADDR							

7	6	5	4	3	2	1	0
DADDR							

This register can only be written if the WPEN bit is cleared in “DMAC Write Protection Mode Register”.

DADDR: Channel x Destination Address

This register must be aligned with the destination transfer width.

© 2017 Microchip Technology Inc.

31.8.15 DMAC Channel x [x = 0..7] Descriptor Address Register

Name: DMAC_DSCRx [x = 0..7]

Address:0xFFFFEC44 [0], 0xFFFFEC6C [1], 0xFFFFEC94 [2], 0xFFFFECBC [3], 0xFFFFECE4 [4], 0xFFFFED0C [5], 0xFFFFED34 [6], 0xFFFFED5C [7]

Access: Read/Write

31	30	29	28	27	26	25	24
DSCR							

23	22	21	20	19	18	17	16
DSCR							

15	14	13	12	11	10	9	8
DSCR							

7	6	5	4	3	2	1	0
DSCR						DSCR_IF	

This register can only be written if the WPEN bit is cleared in “DMAC Write Protection Mode Register”.

DSCR_IF: Descriptor Interface Selection

Value	Name	Description
0	AHB_IF0	The buffer transfer descriptor is fetched via AHB-Lite Interface 0 (first DMA Master Interface)
1	AHB_IF1	The buffer transfer descriptor is fetched via AHB-Lite Interface 1 (second DMA Master Interface)

DSCR: Buffer Transfer Descriptor Address

This address is word aligned.

31.8.16 DMAC Channel x [x = 0..7] Control A Register

Name: DMAC_CTRLAx [x = 0..7]

Address:0xFFFFEC48 [0], 0xFFFFEC70 [1], 0xFFFFEC98 [2], 0xFFFFECC0 [3], 0xFFFFECE8 [4], 0xFFFFED10 [5], 0xFFFFED38 [6], 0xFFFFED60 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
DONE	–	DST_WIDTH		–	–	SRC_WIDTH	

23	22	21	20	19	18	17	16
–	DCSIZE			–	SCSIZE		

15	14	13	12	11	10	9	8
BTSIZE							

7	6	5	4	3	2	1	0
BTSIZE							

This register can only be written if the WPEN bit is cleared in “DMAC Write Protection Mode Register”.

BTSIZE: Buffer Transfer Size

The transfer size relates to the number of transfers to be performed, that is, for writes it refers to the number of source width transfers to perform when DMAC is flow controller. For reads, BTSIZE refers to the number of transfers completed on the Source Interface. When this field is cleared, the DMAC module is automatically disabled when the relevant channel is enabled.

SCSIZE: Source Chunk Transfer Size

Value	Name	Description
000	CHK_1	1 data transferred
001	CHK_4	4 data transferred
010	CHK_8	8 data transferred
011	CHK_16	16 data transferred

DCSIZE: Destination Chunk Transfer Size

Value	Name	Description
000	CHK_1	1 data transferred
001	CHK_4	4 data transferred
010	CHK_8	8 data transferred
011	CHK_16	16 data transferred

SRC_WIDTH: Transfer Width for the Source

Value	Name	Description
00	BYTE	The transfer size is set to 8-bit width
01	HALF_WORD	The transfer size is set to 16-bit width
1X	WORD	The transfer size is set to 32-bit width

© 2017 Microchip Technology Inc.

DST_WIDTH: Transfer Width for the Destination

Value	Name	Description
00	BYTE	The transfer size is set to 8-bit width
01	HALF_WORD	The transfer size is set to 16-bit width
1X	WORD	The transfer size is set to 32-bit width

DONE: Current Descriptor Stop Command and Transfer Completed Memory Indicator

0: The transfer is performed.

1: If SOD bit in DMAC_CFG is set to true, then the DMAC is automatically disabled when an LLI updates the content of this register.

The DONE bit is written back to memory at the end of the current descriptor transfer.

31.8.17 DMAC Channel x [x = 0..7] Control B Register

Name:DMAC_CTRLBx [x = 0..7]

Address:0xFFFFEC4C [0], 0xFFFFEC74 [1], 0xFFFFEC9C [2], 0xFFFFECC4 [3], 0xFFFFECEC [4], 0xFFFFED14 [5], 0xFFFFED3C [6], 0xFFFFED64 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
AUTO	IEN	DST_INCR		–	–	SRC_INCR	

23	22	21	20	19	18	17	16
–	FC		DST_DSCR	–	–	–	SRC_DSCR

15	14	13	12	11	10	9	8
–	–	–	DST_PIP	–	–	–	SRC_PIP

7	6	5	4	3	2	1	0
–	–	DIF		–	–	SIF	

This register can only be written if the WPEN bit is cleared in “DMAC Write Protection Mode Register”.

SIF: Source Interface Selection Field

Value	Name	Description
0	AHB_IF0	The source transfer is done via AHB_Lite Interface 0 (first DMA Master Interface)
1	AHB_IF1	The source transfer is done via AHB_Lite Interface 1 (second DMA Master Interface)

DIF: Destination Interface Selection Field

Value	Name	Description
0	AHB_IF0	The destination transfer is done via AHB_Lite Interface 0 (first DMA Master Interface)
1	AHB_IF1	The destination transfer is done via AHB_Lite Interface 1 (second DMA Master Interface)

SRC_PIP: Source Picture-in-Picture Mode

0 (DISABLE): Picture-in-Picture mode is disabled. The source data area is contiguous.

1 (ENABLE): Picture-in-Picture mode is enabled. When the source PIP counter reaches the programmable boundary, the address is automatically incremented by a user defined amount.

DST_PIP: Destination Picture-in-Picture Mode

0 (DISABLE): Picture-in-Picture mode is disabled. The Destination data area is contiguous.

1 (ENABLE): Picture-in-Picture mode is enabled. When the Destination PIP counter reaches the programmable boundary the address is automatically incremented by a user-defined amount.

SRC_DSCR: Source Address Descriptor

0 (FETCH_FROM_MEM): Source address is updated when the descriptor is fetched from the memory.

1 (FETCH_DISABLE): Buffer Descriptor Fetch operation is disabled for the source.

DST_DSCR: Destination Address Descriptor

0 (FETCH_FROM_MEM): Destination address is updated when the descriptor is fetched from the memory.

1 (FETCH_DISABLE): Buffer Descriptor Fetch operation is disabled for the destination.

© 2017 Microchip Technology Inc.

FC: Flow Control

This field defines which device controls the size of the buffer transfer, also referred to as the Flow Controller.

Value	Name	Description
00	MEM2MEM_DMA_FC	Memory-to-Memory Transfer DMAC is flow controller
01	MEM2PER_DMA_FC	Memory-to-Peripheral Transfer DMAC is flow controller
10	PER2MEM_DMA_FC	Peripheral-to-Memory Transfer DMAC is flow controller
11	PER2PER_DMA_FC	Peripheral-to-Peripheral Transfer DMAC is flow controller

SRC_INCR: Incrementing, Decrementing or Fixed Address for the Source

Value	Name	Description
00	INCREMENTING	The source address is incremented
01	DECREMENTING	The source address is decremented
10	FIXED	The source address remains unchanged

DST_INCR: Incrementing, Decrementing or Fixed Address for the Destination

Value	Name	Description
00	INCREMENTING	The destination address is incremented
01	DECREMENTING	The destination address is decremented
10	FIXED	The destination address remains unchanged

IEN: Interrupt Enable Not

0: When the buffer transfer is completed, the BTCx flag is set in the DMAC_EBCISR. This bit is active low.

1: When the buffer transfer is completed, the BTCx flag is not set.

If this bit is cleared, when the buffer transfer is completed, the BTCx flag is set in the DMAC_EBCISR.

AUTO: Automatic Multiple Buffer Transfer

0 (DISABLE): Automatic multiple buffer transfer is disabled.

1 (ENABLE): Automatic multiple buffer transfer is enabled. This bit enables replay mode or contiguous mode when several buffers are transferred.

31.8.18 DMAC Channel x [x = 0..7] Configuration Register

Name: DMAC_CFGx [x = 0..7]

Address:0xFFFFEC50 [0], 0xFFFFEC78 [1], 0xFFFFECA0 [2], 0xFFFFECC8 [3], 0xFFFFECF0 [4], 0xFFFFED18 [5], 0xFFFFED40 [6], 0xFFFFED68 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	FIFOCFG		–	AHB_PROT		

23	22	21	20	19	18	17	16
–	LOCK_IF_L	LOCK_B	LOCK_IF	–	–	–	SOD

15	14	13	12	11	10	9	8
DST_PER_MSB		DST_H2SEL	DST_REP	SRC_PER_MSB		SRC_H2SEL	SRC_REP

7	6	5	4	3	2	1	0
DST_PER				SRC_PER			

This register can only be written if the WPEN bit is cleared in "DMAC Write Protection Mode Register".

SRC_PER: Source with Peripheral identifier

Channel x Source Request is associated with peripheral identifier coded SRC_PER handshaking interface.

DST_PER: Destination with Peripheral identifier

Channel x Destination Request is associated with peripheral identifier coded DST_PER handshaking interface.

SRC_REP: Source Reloaded from Previous

0 (CONTIGUOUS_ADDR): When automatic mode is activated, source address is contiguous between two buffers.

1 (RELOAD_ADDR): When automatic mode is activated, the source address and the control register are reloaded from previous transfer.

SRC_H2SEL: Software or Hardware Selection for the Source

0 (SW): Software handshaking interface is used to trigger a transfer request.

1 (HW): Hardware handshaking interface is used to trigger a transfer request.

SRC_PER_MSB: SRC_PER Most Significant Bits

This field indicates the Most Significant bits of the SRC_PER field.

DST_REP: Destination Reloaded from Previous

0 (CONTIGUOUS_ADDR): When automatic mode is activated, destination address is contiguous between two buffers.

1 (RELOAD_ADDR): When automatic mode is activated, the destination and the control register are reloaded from the previous transfer.

DST_H2SEL: Software or Hardware Selection for the Destination

0 (SW): Software handshaking interface is used to trigger a transfer request.

1 (HW): Hardware handshaking interface is used to trigger a transfer request.

DST_PER_MSB: DST_PER Most Significant Bits

This field indicates the Most Significant bits of the DST_PER field.

SOD: Stop On Done

0 (DISABLE): STOP ON DONE disabled, the descriptor fetch operation ignores the DMAC_CTRLAx.DONE bit.

1 (ENABLE): STOP ON DONE activated, the DMAC module is automatically disabled if DMAC_CTRLAx.DONE bit is set.

© 2017 Microchip Technology Inc.

LOCK_IF: Interface Lock

0 (DISABLE): Interface Lock capability is disabled

1 (ENABLE): Interface Lock capability is enabled

LOCK_B: Bus Lock

0 (DISABLE): AHB Bus Locking capability is disabled.

1(ENABLE): AHB Bus Locking capability is enabled.

LOCK_IF_L: Master Interface Arbiter Lock

0 (CHUNK): The Master Interface Arbiter is locked by the channel x for a chunk transfer.

1 (BUFFER): The Master Interface Arbiter is locked by the channel x for a buffer transfer.

AHB_PROT: AHB Protection

AHB_PROT field provides additional information about a bus access and is primarily used to implement some level of protection.

HPROT[3]	HPROT[2]	HPROT[1]	HPROT[0]	Description
			1	Data access
		AHB_PROT[0]		0: User Access 1: Privileged Access
	AHB_PROT[1]			0: Not Bufferable 1: Bufferable
AHB_PROT[2]				0: Not cacheable 1: Cacheable

FIFOCFG: FIFO Configuration

Value	Name	Description
00	ALAP_CFG	The largest defined length AHB burst is performed on the destination AHB interface.
01	HALF_CFG	When half FIFO size is available/filled, a source/destination request is serviced.
10	ASAP_CFG	When there is enough space/data available to perform a single AHB access, then the request is serviced.

31.8.19 DMAC Channel x [x = 0..7] Source Picture-in-Picture Configuration Register

Name: DMAC_SPIPx [x = 0..7]

Address:0xFFFFEC54 [0], 0xFFFFEC7C [1], 0xFFFFECA4 [2], 0xFFFFECCC [3], 0xFFFFECF4 [4], 0xFFFFED1C [5], 0xFFFFED44 [6], 0xFFFFED6C [7]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	SPIP_BOUNDARY	

23	22	21	20	19	18	17	16
SPIP_BOUNDARY							

15	14	13	12	11	10	9	8
SPIP_HOLE							

7	6	5	4	3	2	1	0
SPIP_HOLE							

SPIP_HOLE: Source Picture-in-Picture Hole

This field indicates the value to add to the address when the programmable boundary has been reached.

SPIP_BOUNDARY: Source Picture-in-Picture Boundary

This field indicates the number of source transfers to perform before the automatic address increment operation.

© 2017 Microchip Technology Inc.

31.8.20 DMAC Channel x [x = 0..7] Destination Picture-in-Picture Configuration Register

Name:DMAC_DPIPx [x = 0..7]

Address:0xFFFFEC58 [0], 0xFFFFEC80 [1], 0xFFFFECA8 [2], 0xFFFFECD0 [3], 0xFFFFECF8 [4], 0xFFFFED20 [5], 0xFFFFED48 [6], 0xFFFFED70 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	DPIP_BOUNDARY	

23	22	21	20	19	18	17	16
DPIP_BOUNDARY							

15	14	13	12	11	10	9	8
DPIP_HOLE							

7	6	5	4	3	2	1	0
DPIP_HOLE							

DPIP_HOLE: Destination Picture-in-Picture Hole

This field indicates the value to add to the address when the programmable boundary has been reached.

DPIP_BOUNDARY: Destination Picture-in-Picture Boundary

This field indicates the number of source transfers to perform before the automatic address increment operation.

31.8.21 DMAC Write Protection Mode Register

Name:DMAC_WPMR

Address:0xFFFFEDE4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

WPEN: Write Protection Enable

0: Disables the Write Protection if WPKEY corresponds to 0x444D41 (“DMA” in ASCII).

1: Enables the Write Protection if WPKEY corresponds to 0x444D41 (“DMA” in ASCII).

See Section 31.6.7 ”Register Write Protection” for the list of registers that can be write-protected.

WPKEY: Write Protection Key

Value	Name	Description
0x444D41	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

© 2017 Microchip Technology Inc.

31.8.22 DMAC Write Protection Status Register

Name:DMAC_WPSR

Address:0xFFFFEDE8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

WPVS: Write Protection Violation Status

0: No write protection violation has occurred since the last read of the DMAC_WPSR.

1: A write protection violation has occurred since the last read of the DMAC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

WPVSRC: Write Protection Violation Source

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

32. USB Device Port (UDP)

32.1 Description

The USB Device Port (UDP) is compliant with the Universal Serial Bus (USB) 2.0 full-speed device specification.

Each endpoint can be configured in one of several USB transfer types. It can be associated with one or two banks of a dual-port RAM used to store the current data payload. If two banks are used, one DPR bank is read or written by the processor, while the other is read or written by the USB device peripheral. This feature is mandatory for isochronous endpoints. Thus the device maintains the maximum bandwidth (1 Mbyte/s) by working with endpoints with two banks of DPR.

Table 32-1: USB Endpoint Description

Endpoint No.	Mnemonic	Dual-Bank(1)	Max. Endpoint Size	Endpoint Type
0	EP0	No	64	Control/Bulk/Interrupt
1	EP1	Yes	64	Bulk/Iso/Interrupt
2	EP2	Yes	64	Bulk/Iso/Interrupt
3	EP3	No	64	Control/Bulk/Interrupt
4	EP4	Yes	512	Bulk/Iso/Interrupt
5	EP5	Yes	512	Bulk/Iso/Interrupt

Note 1: The Dual-Bank function provides two banks for an endpoint. This feature is used for ping-pong mode.

Suspend and resume are automatically detected by the USB device, which notifies the processor by raising an interrupt. Depending on the product, an external signal can be used to send a wakeup request to the USB host controller.

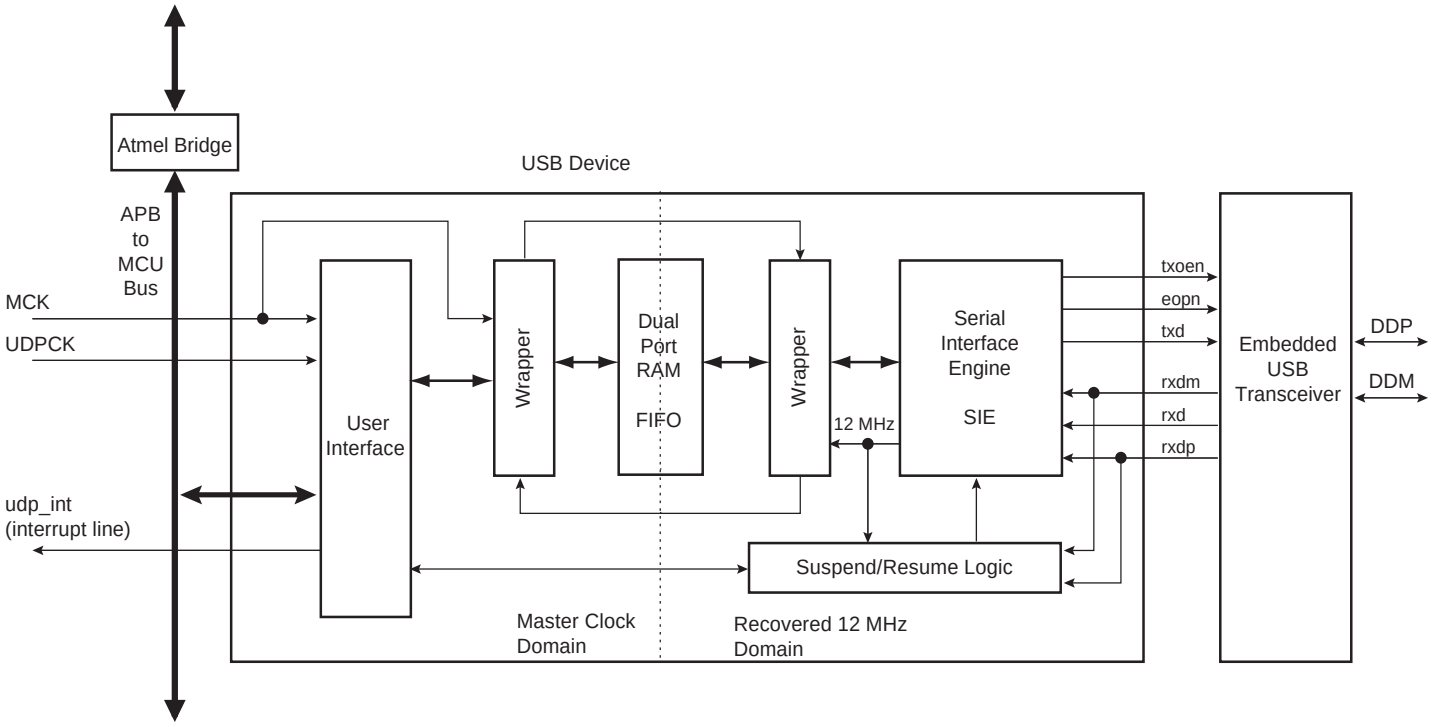
32.2 Embedded Characteristics

- USB 2.0 Full-speed Compliant, 12 Mbit/s
- Embedded USB 2.0 Full-speed Transceiver
- Integrated Pull-up on DDP
- 6 Endpoints
- Embedded Dual-port RAM for Endpoints
- Suspend/Resume Logic
- Ping-pong Mode (2 Memory Banks) for Isochronous and Bulk Endpoints

© 2017 Microchip Technology Inc.

32.3 Block Diagram

Figure 32-1: Block Diagram



Access to the UDP is via the APB bus interface. Read and write to the data FIFO are done by reading and writing 8-bit values to APB registers.

The UDP peripheral requires two clocks: one peripheral clock used by the Master Clock domain (MCK) and a 48 MHz clock (UDPCK) used by the 12 MHz domain.

A USB 2.0 full-speed pad is embedded and controlled by the Serial Interface Engine (SIE).

32.3.1 Signal Description

Table 32-2: Signal Names

Signal Name	Description	Type
UDPCK	48 MHz clock	Input
MCK	Master clock	Input
udp_int	Interrupt line connected to the Interrupt Controller	Input
DDP	USB D+ line	I/O
DDM	USB D- line	I/O

32.4 Product Dependencies

For further details on the USB Device hardware implementation, see the specific Product Properties document.

The USB physical transceiver is integrated into the product. The bidirectional differential signals DDP and DDM are available from the product boundary.

One I/O line may be used by the application to check that VBUS is still available from the host. Self-powered devices may use this entry to be notified that the host has been powered off. In this case, the pull-up on DDP must be disabled in order to prevent feeding current to the host. The application should disconnect the transceiver, then remove the pull-up.

32.4.1 I/O Lines

DDP and DDM are not controlled by any PIO controllers. The embedded USB physical transceiver is controlled by the USB device peripheral.

To reserve an I/O line to check VBUS, the programmer must first program the PIO controller to assign this I/O in input PIO mode.

32.4.2 Power Management

The USB device peripheral requires a 48 MHz clock. This clock must be generated by a PLL driven by a clock source with an accuracy of ± 0.25% (note that the fast RC oscillator cannot be used).

Thus, the USB device receives two clocks from the Power Management Controller (PMC):

- Master clock, MCK, used to drive the peripheral user interface
- UDPCK, used to interface with the bus USB signals (recovered 12 MHz domain)

WARNING: The UDP peripheral clock in the PMC must be enabled before any read/write operations to the UDP registers including the Transceiver Control Register (UDP_TXVC).

32.4.3 Interrupt Sources

The USB device interface has an interrupt line connected to the Interrupt Controller.

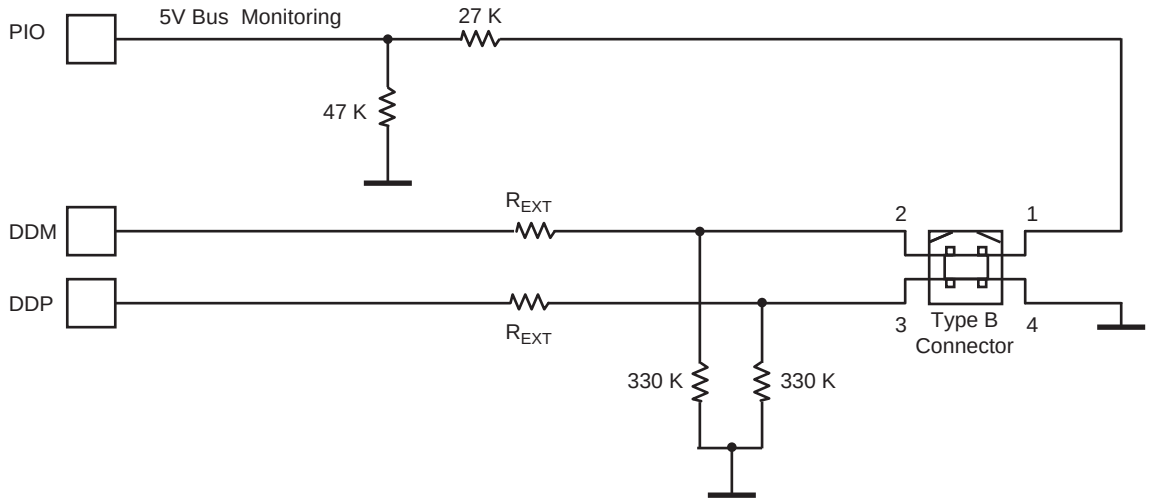
Handling the USB device interrupt requires programming the Interrupt Controller before configuring the UDP.

Table 32-3: Peripheral IDs

Instance	ID
UDP	23

32.5 Typical Connection

Figure 32-2: Board Schematic to Interface Device Peripheral



32.5.1 USB Device Transceiver

The USB device transceiver is embedded in the product. However, discrete components are required for each of the following actions:

- to monitor VBUS voltage
- for line termination
- to disconnect the host for reduced power consumption

© 2017 Microchip Technology Inc.

32.5.2 VBUS Monitoring

VBUS monitoring is required to detect host connection. VBUS monitoring is done using a standard PIO with internal pull-up disabled. When the host is switched off, it should be considered as a disconnect, the pull-up must be disabled in order to prevent powering the host through the pull-up resistor.

When the host is disconnected and the transceiver is enabled, then DDP and DDM are floating. This may lead to over consumption. A solution is to connect 330 KΩ pull-downs on DDP and DDM. These pull-downs do not alter DDP and DDM signal integrity.

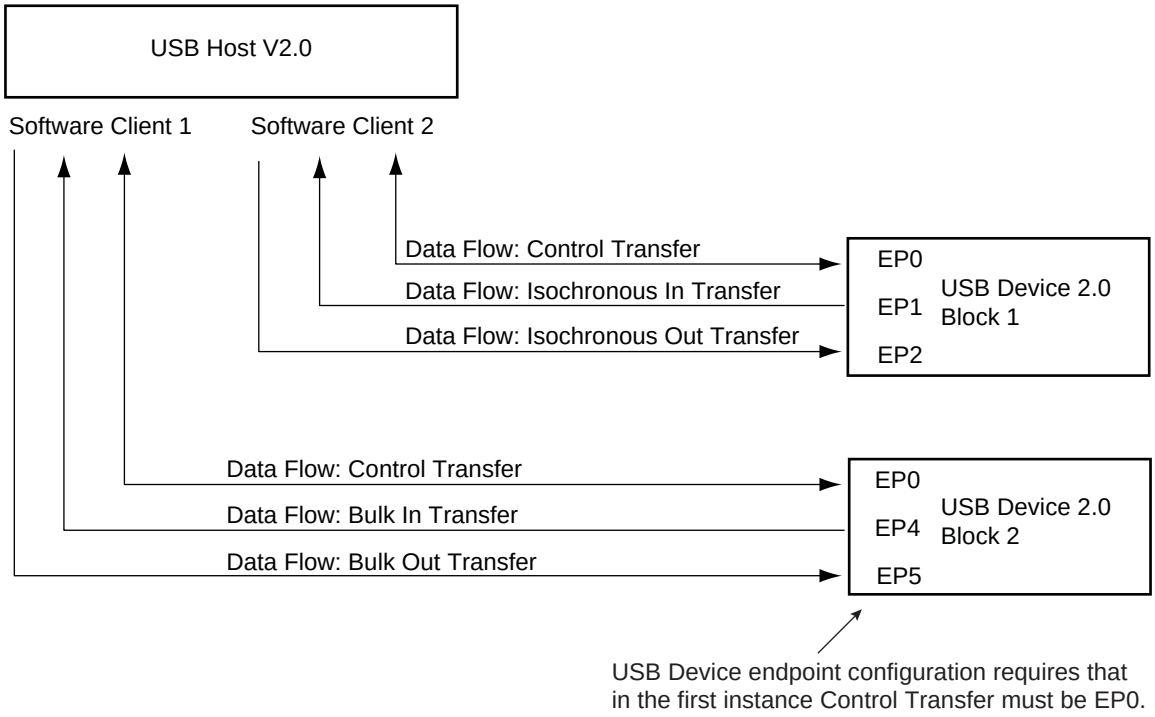
A termination serial resistor must be connected to DDP and DDM. The resistor value is defined in the electrical characteristics of the product (R_{EXT}).

32.6 Functional Description

32.6.1 USB 2.0 Full-speed Introduction

The USB 2.0 full-speed provides communication services between host and attached USB devices. Each device is offered with a collection of communication flows (pipes) associated with each endpoint. Software on the host communicates with a USB device through a set of communication flows.

Figure 32-3: Example of USB 2.0 Full-speed Communication Control



The Control Transfer endpoint EP0 is always used when a USB device is first configured (USB 2.0 specifications).

32.6.1.1 USB 2.0 Full-speed Transfer Types

A communication flow is carried over one of four transfer types defined by the USB device.

Table 32-4: USB Communication Flow

Transfer	Direction	Bandwidth	Supported Endpoint Size	Error Detection	Retrying
Control	Bidirectional	Not guaranteed	8, 16, 32, 64	Yes	Automatic
Isochronous	Unidirectional	Guaranteed	512	Yes	No
Interrupt	Unidirectional	Not guaranteed	≤ 64	Yes	Yes
Bulk	Unidirectional	Not guaranteed	8, 16, 32, 64	Yes	Yes

32.6.1.2 USB Bus Transactions

Each transfer results in one or more transactions over the USB bus. There are three kinds of transactions flowing across the bus in packets:

- Setup Transaction
- Data IN Transaction
- Data OUT Transaction

32.6.1.3 USB Transfer Event Definitions

As indicated below, transfers are sequential events carried out on the USB bus.

Table 32-5: USB Transfer Events

Transfer		Transaction
Direction	**Type**	
CONTROL (bidirectional)	Control(1)(3)	Setup transaction → Data IN transactions → Status OUT transaction
		Setup transaction → Data OUT transactions → Status IN transaction
		Setup transaction → Status IN transaction
IN (device toward host)	Interrupt IN	Data IN transaction → Data IN transaction
	Isochronous IN(2)	
	Bulk IN	
OUT (host toward device)	Interrupt OUT	Data OUT transaction → Data OUT transaction
	Isochronous OUT(2)	
	Bulk OUT	

Note 1: Control transfer must use endpoints with no ping-pong attributes.

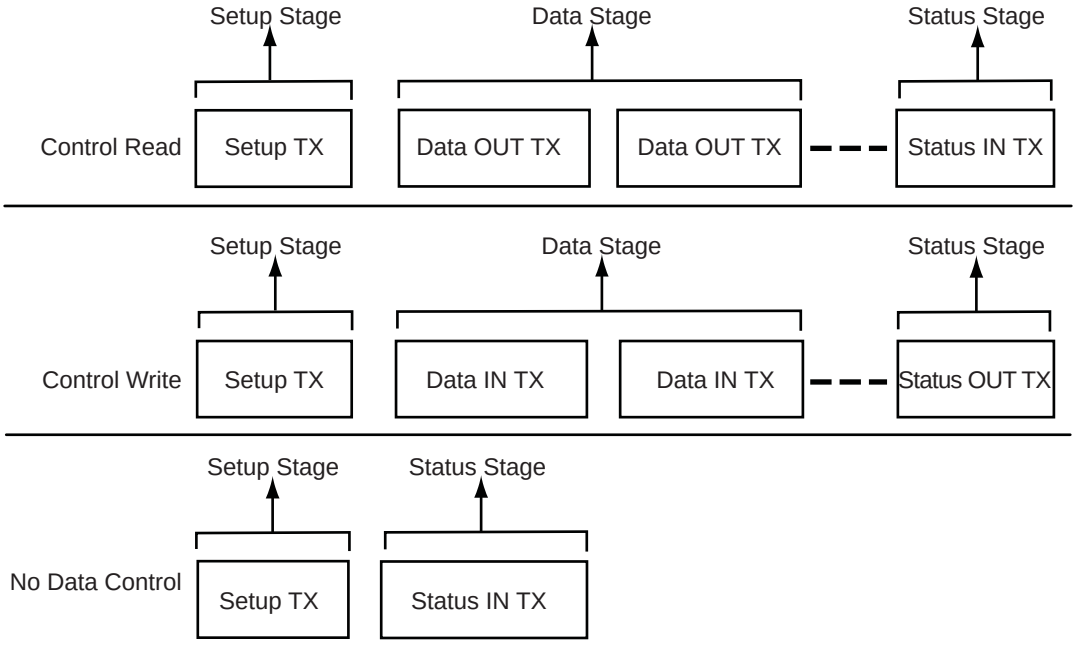
2: Isochronous transfers must use endpoints with ping-pong attributes.

3: Control transfers can be aborted using a stall handshake.

A status transaction is a special type of host-to-device transaction used only in a control transfer. The control transfer must be performed using endpoints with no ping-pong attributes. According to the control sequence (read or write), the USB device sends or receives a status transaction.

© 2017 Microchip Technology Inc.

Figure 32-4: Control Read and Write Sequences



Note 1: During the Status IN stage, the host waits for a zero length packet (Data IN transaction with no data) from the device using DATA1 PID. Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0,* for more information on the protocol layer.

2: During the Status OUT stage, the host emits a zero length packet to the device (Data OUT transaction with no data).

32.6.2 Handling Transactions with USB 2.0 Device Peripheral

32.6.2.1 Setup Transaction

Setup is a special type of host-to-device transaction used during control transfers. Control transfers must be performed using endpoints with no ping-pong attributes. A setup transaction needs to be handled as soon as possible by the firmware. It is used to transmit requests from the host to the device. These requests are then handled by the USB device and may require more arguments. The arguments are sent to the device by a Data OUT transaction which follows the setup transaction. These requests may also return data. The data is carried out to the host by the next Data IN transaction which follows the setup transaction. A status transaction ends the control transfer.

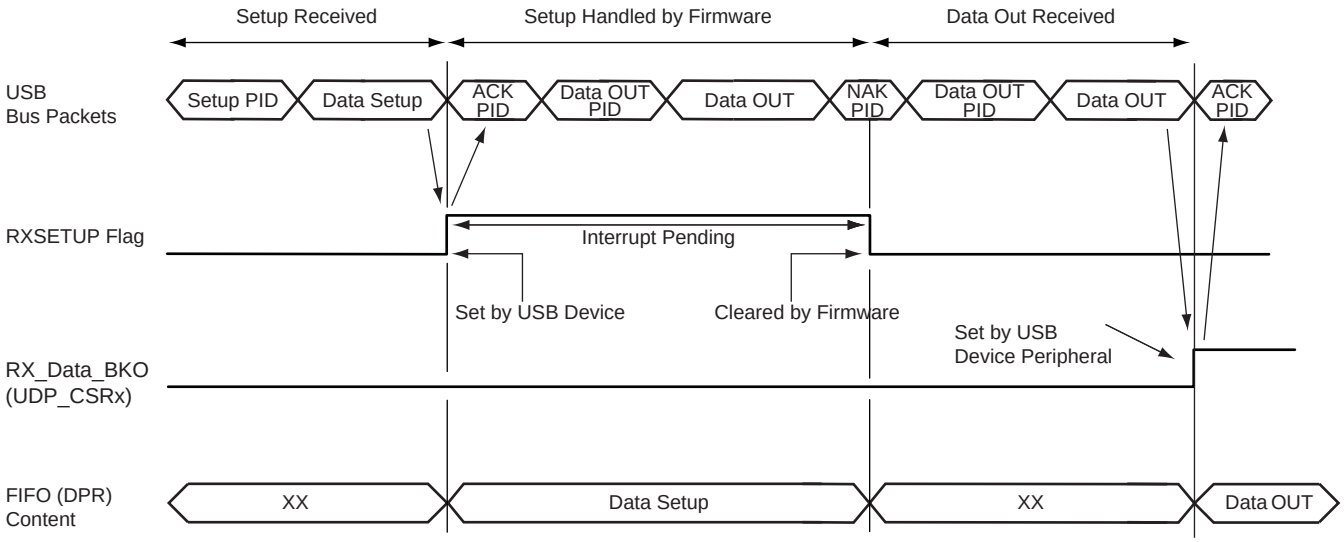
When a setup transfer is received by the USB endpoint:

- The USB device automatically acknowledges the setup packet
- RXSETUP is set in the corresponding Endpoint Control and Status Register x (UDP_CSRx)
- An endpoint interrupt is generated while the RXSETUP is not cleared. This interrupt is carried out to the microcontroller if interrupts are enabled for this endpoint.

Thus, firmware must detect the RXSETUP polling the UDP_CSRx or catching an interrupt, read the setup packet in the FIFO, then clear the RXSETUP. RXSETUP cannot be cleared before the setup packet has been read in the FIFO. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the setup packet in the FIFO.

© 2017 Microchip Technology Inc.

Figure 32-5: Setup Transaction Followed by a Data OUT Transaction



32.6.2.2 Data IN Transaction

Data IN transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the device to the host. Data IN transactions in isochronous transfer must be done using endpoints with ping-pong attributes.

Using Endpoints Without Ping-pong Attributes

To perform a Data IN transaction using a non ping-pong endpoint:

1. The application checks if it is possible to write in the FIFO by polling TXPKTRDY in the endpoint's UDP_CSRx (TXPKTRDY must be cleared).
2. The application writes the first packet of data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's FIFO Data Register x (UDP_FDRx).
3. The application notifies the USB peripheral it has finished by setting the TXPKTRDY in the endpoint's UDP_CSRx.
4. The application is notified that the endpoint's FIFO has been released by the USB device when TXCOMP in the endpoint's UDP_CSRx has been set. Then an interrupt for the corresponding endpoint is pending while TXCOMP is set.
5. The microcontroller writes the second packet of data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's UDP_FDRx.
6. The microcontroller notifies the USB peripheral it has finished by setting the TXPKTRDY in the endpoint's UDP_CSRx.
7. The application clears the TXCOMP in the endpoint's UDP_CSRx.

After the last packet has been sent, the application must clear TXCOMP once this has been set.

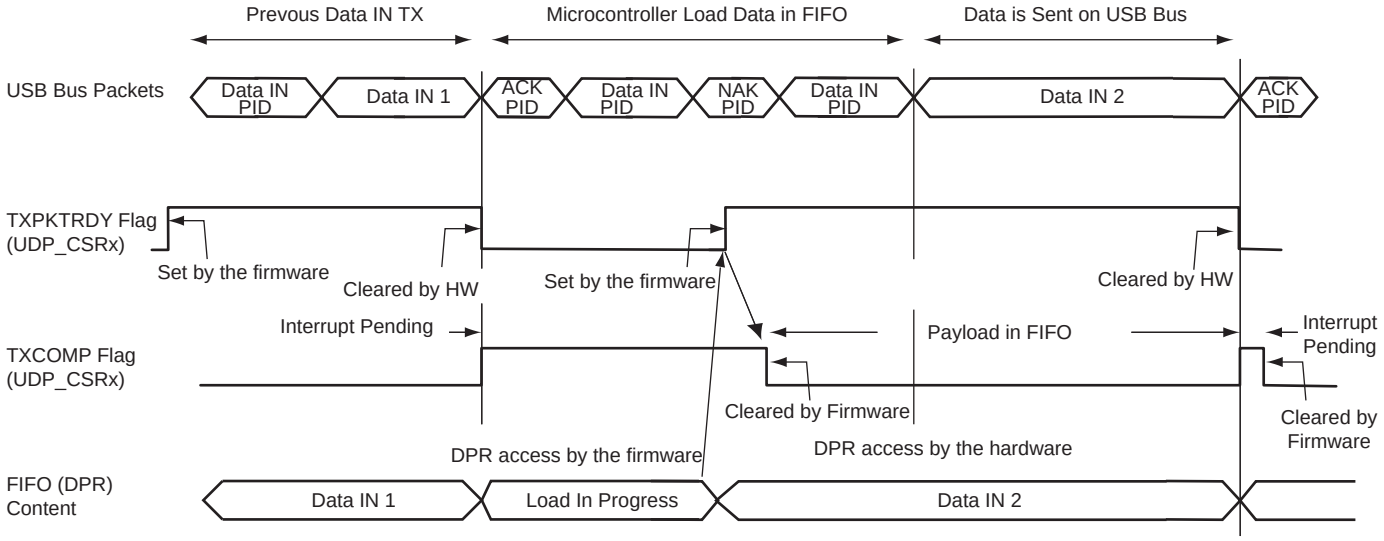
TXCOMP is set by the USB device when it has received an ACK PID signal for the Data IN packet. An interrupt is pending while TXCOMP is set.

Warning: TX_COMP must be cleared after TX_PKTRDY has been set.

Note: Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the Data IN protocol layer.

© 2017 Microchip Technology Inc.

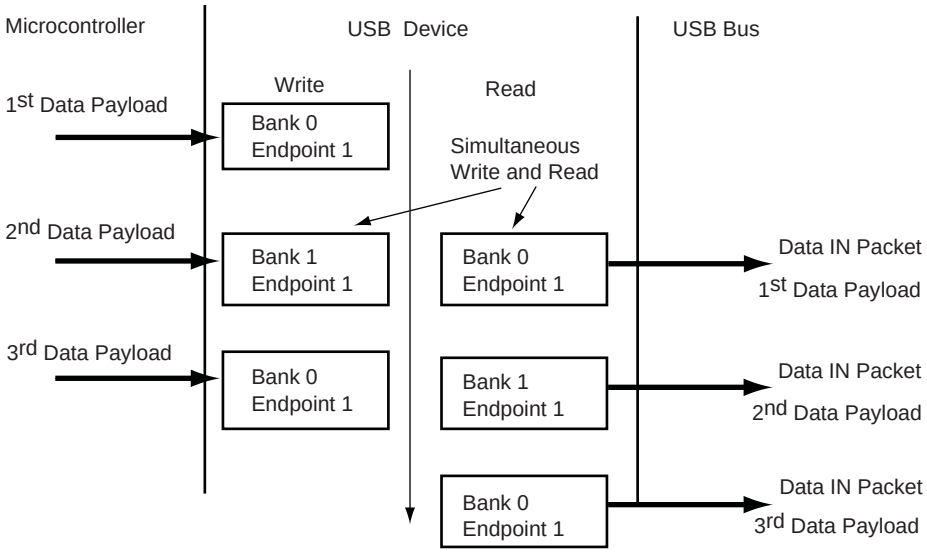
Figure 32-6: Data IN Transfer for Non Ping-pong Endpoint



Using Endpoints With Ping-pong Attribute

The use of an endpoint with ping-pong attributes is necessary during isochronous transfer. This also allows handling the maximum bandwidth defined in the USB specification during bulk transfer. To be able to guarantee a constant or the maximum bandwidth, the microcontroller must prepare the next data payload to be sent while the current one is being sent by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

Figure 32-7: Bank Swapping Data IN Transfer for Ping-pong Endpoints

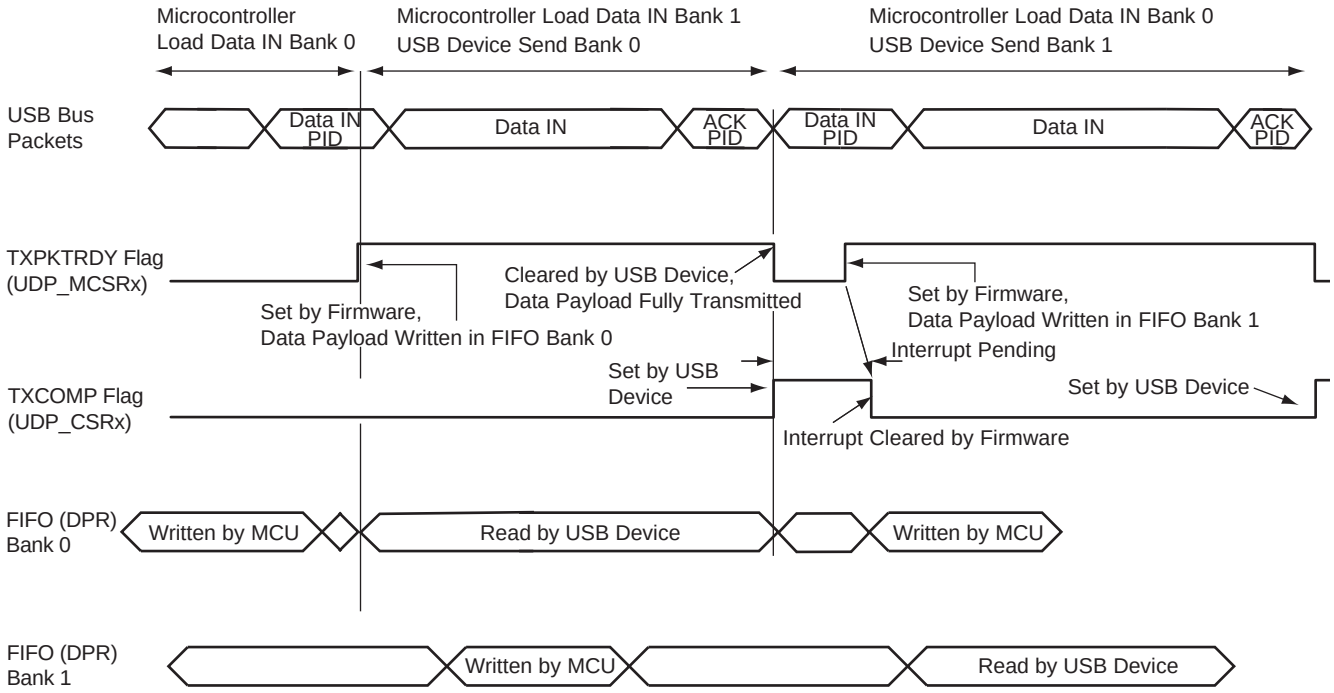


When using a ping-pong endpoint, the following procedures are required to perform Data IN transactions:

1. The microcontroller checks if it is possible to write in the FIFO by polling TXPKTRDY to be cleared in the endpoint's UDP_CSRx.
2. The microcontroller writes the first data payload to be sent in the FIFO (Bank 0), writing zero or more byte values in the endpoint's UDP_FDRx.
3. The microcontroller notifies the USB peripheral it has finished writing in Bank 0 of the FIFO by setting the TXPKTRDY in the endpoint's UDP_CSRx.
4. Without waiting for TXPKTRDY to be cleared, the microcontroller writes the second data payload to be sent in the FIFO (Bank 1), writing zero or more byte values in the endpoint's UDP_FDRx.

5. The microcontroller is notified that the first Bank has been released by the USB device when TXCOMP in the endpoint's UDP_CSRx is set. An interrupt is pending while TXCOMP is being set.
6. Once the microcontroller has received TXCOMP for the first Bank, it notifies the USB device that it has prepared the second Bank to be sent, raising TXPKTRDY in the endpoint's UDP_CSRx.
7. At this step, Bank 0 is available and the microcontroller can prepare a third data payload to be sent.

Figure 32-8: Data IN Transfer for Ping-pong Endpoint



Warning: There is software critical path due to the fact that once the second bank is filled, the driver has to wait for TX_COMP to set TX_PKTRDY. If the delay between receiving TX_COMP is set and TX_PKTRDY is set too long, some Data IN packets may be NACKed, reducing the bandwidth.

Warning: TX_COMP must be cleared after TX_PKTRDY has been set.

32.6.2.3 Data OUT Transaction

Data OUT transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the host to the device. Data OUT transactions in isochronous transfers must be done using endpoints with ping-pong attributes.

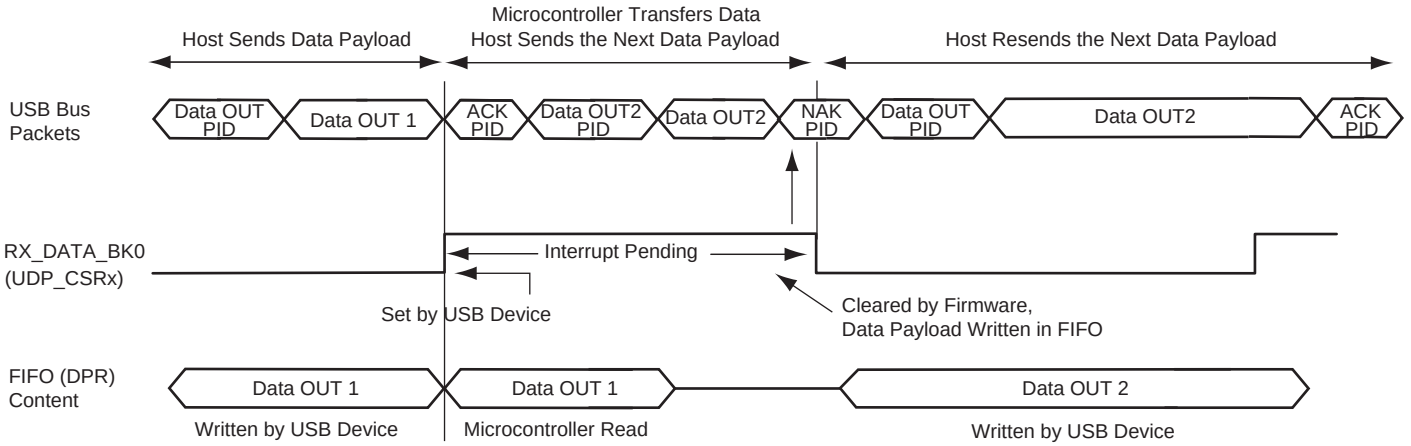
Data OUT Transaction Without Ping-pong Attributes

To perform a Data OUT transaction, using a non ping-pong endpoint:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. While the FIFO associated to this endpoint is being used by the microcontroller, a NAK PID is returned to the host. Once the FIFO is available, data are written to the FIFO by the USB device and an ACK is automatically carried out to the host.
3. The microcontroller is notified that the USB device has received a data payload polling RX_DATA_BK0 in the endpoint's UDP_CSRx. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
4. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_CSRx.
5. The microcontroller carries out data received from the endpoint's memory to its memory. Data received is available by reading the endpoint's UDP_FDRx.
6. The microcontroller notifies the USB device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_CSRx.
7. A new Data OUT packet can be accepted by the USB device.

© 2017 Microchip Technology Inc.

Figure 32-9: Data OUT Transfer for Non Ping-pong Endpoints

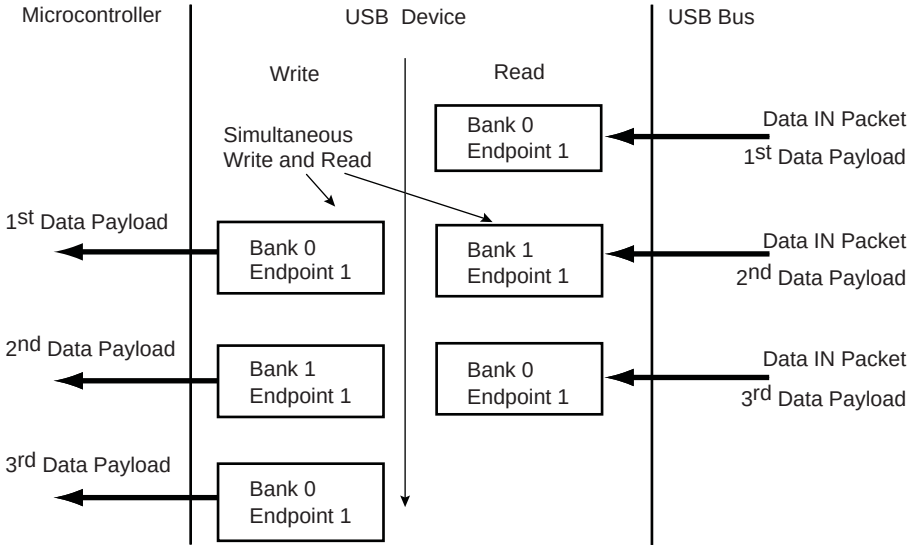


An interrupt is pending while the flag RX_DATA_BK0 is set. Memory transfer between the USB device, the FIFO and microcontroller memory is not possible after RX_DATA_BK0 has been cleared. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the current Data OUT packet in the FIFO.

Using Endpoints With Ping-pong Attributes

During isochronous transfer, using an endpoint with ping-pong attributes is obligatory. To be able to guarantee a constant bandwidth, the microcontroller must read the previous data payload sent by the host, while the current data payload is received by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

Figure 32-10: Bank Swapping in Data OUT Transfers for Ping-pong Endpoints

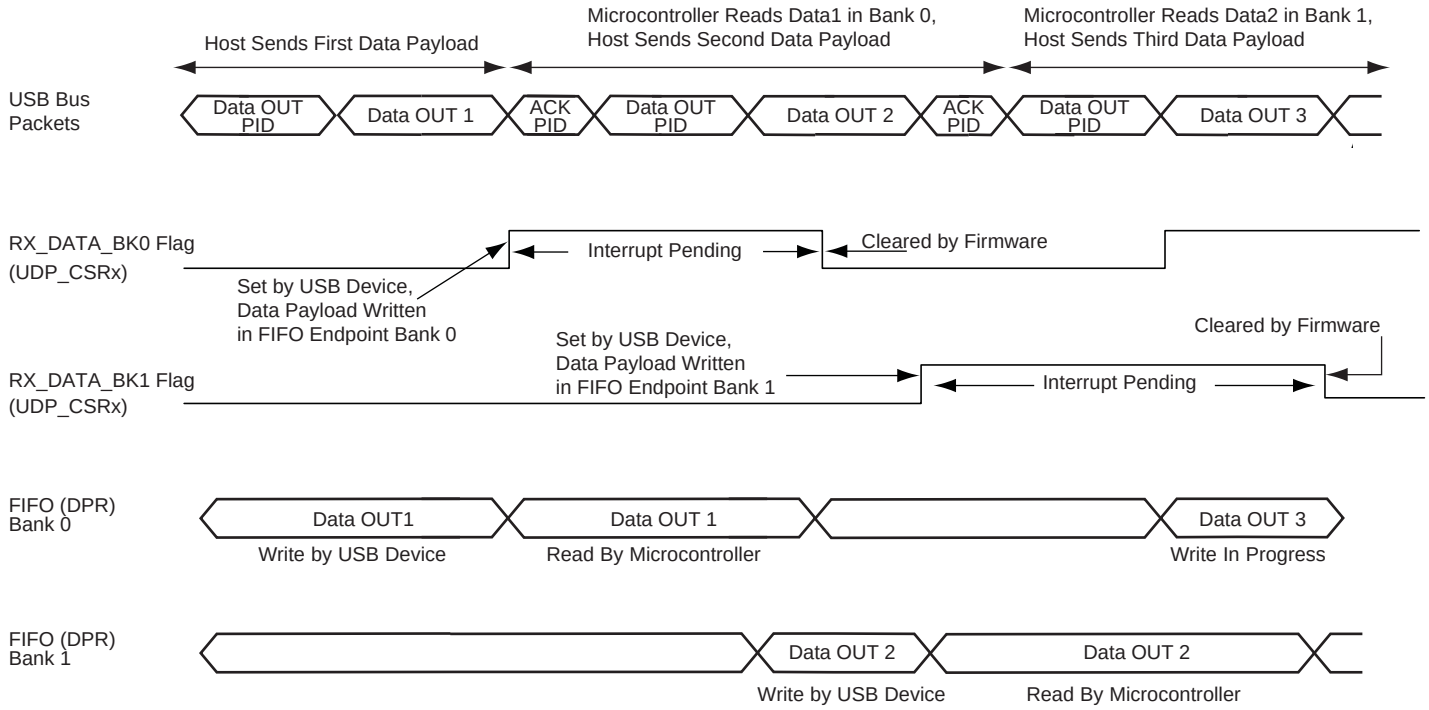


When using a ping-pong endpoint, the following procedures are required to perform Data OUT transactions:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. It is written in the endpoint's FIFO Bank 0.
3. The USB device sends an ACK PID packet to the host. The host can immediately send a second Data OUT packet. It is accepted by the device and copied to FIFO Bank 1.
4. The microcontroller is notified that the USB device has received a data payload, polling RX_DATA_BK0 in the endpoint's UDP_CSRx. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
5. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_CSRx.
6. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is made available by reading the endpoint's UDP_FDRx.

7. The microcontroller notifies the USB peripheral device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_CSRx.
8. A third Data OUT packet can be accepted by the USB peripheral device and copied in the FIFO Bank 0.
9. If a second Data OUT packet has been received, the microcontroller is notified by the flag RX_DATA_BK1 set in the endpoint's UDP_CSRx. An interrupt is pending for this endpoint while RX_DATA_BK1 is set.
10. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is available by reading the endpoint's UDP_FDRx.
11. The microcontroller notifies the USB device it has finished the transfer by clearing RX_DATA_BK1 in the endpoint's UDP_CSRx.
12. A fourth Data OUT packet can be accepted by the USB device and copied in the FIFO Bank 1.

Figure 32-11: Data OUT Transfer for Ping-pong Endpoint



Note: An interrupt is pending while the RX_DATA_BK0 or RX_DATA_BK1 flag is set.

Warning: When RX_DATA_BK0 and RX_DATA_BK1 are both set, there is no way to determine which one to clear first. Thus the software must keep an internal counter to be sure to clear alternatively RX_DATA_BK0 then RX_DATA_BK1. This situation may occur when the software application is busy elsewhere and the two banks are filled by the USB host. Once the application comes back to the USB driver, the two flags are set.

32.6.2.4 Stall Handshake

A stall handshake can be used in one of two distinct occasions. (For more information on the stall handshake, refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0.*)

- A functional stall is used when the halt feature associated with the endpoint is set. (Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the halt feature.)
- To abort the current request, a protocol stall is used, but uniquely with control transfer.

The following procedure generates a stall packet:

1. The microcontroller sets the FORCESTALL flag in the UDP_CSRx endpoint's register.
2. The host receives the stall packet.
3. The microcontroller is notified that the device has sent the stall by polling the STALLSENT to be set. An endpoint interrupt is pending while STALLSENT is set. The microcontroller must clear STALLSENT to clear the interrupt.

© 2017 Microchip Technology Inc.

When a setup transaction is received after a stall handshake, STALLSENT must be cleared in order to prevent interrupts due to STALLSENT being set.

Figure 32-12: Stall Handshake (Data IN Transfer)

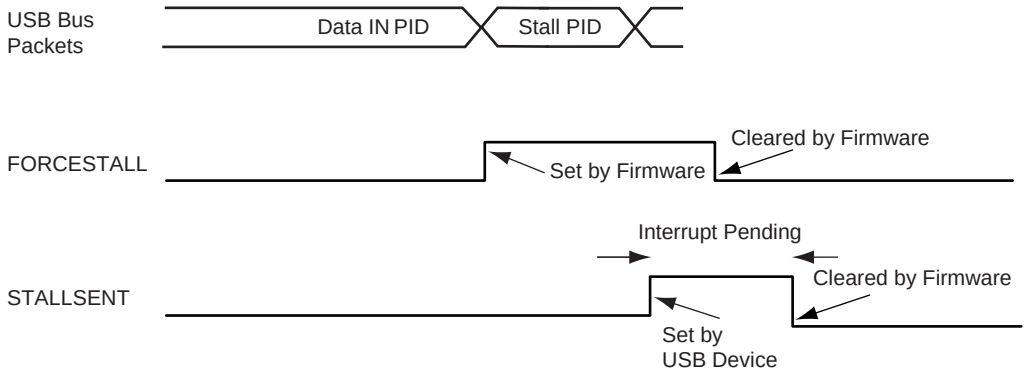
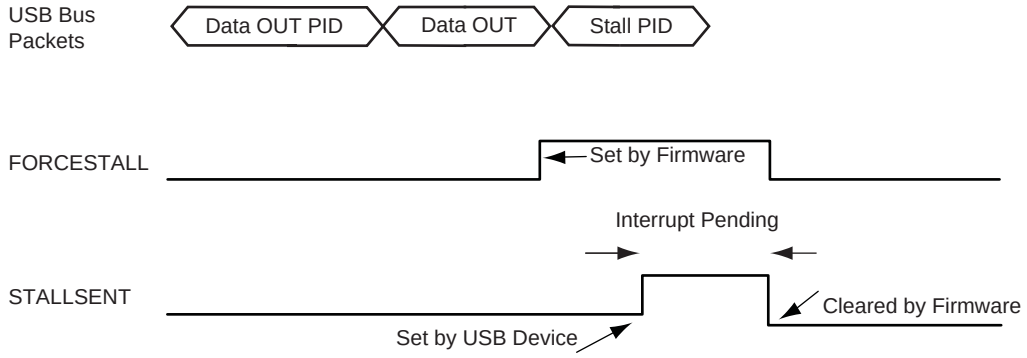


Figure 32-13: Stall Handshake (Data OUT Transfer)



32.6.2.5 Transmit Data Cancellation

Some endpoints have dual-banks whereas some endpoints have only one bank. The procedure to cancel transmission data held in these banks is described below.

To see the organization of dual-bank availability refer to Table 32-1 ”USB Endpoint Description”.

*Endpoints **Without** Dual-Banks*

The cancellation procedure depends on the TXPKTRDY flag value in the UDP_CSR:

- TXPKTRDY is not set:
  - Reset the endpoint to clear the FIFO (pointers). (See Section 32.7.9 ”UDP Reset Endpoint Register”.)
- TXPKTRDY has already been set:
  - Clear TXPKTRDY so that no packet is ready to be sent
  - Reset the endpoint to clear the FIFO (pointers). (See Section 32.7.9 ”UDP Reset Endpoint Register”.)

*Endpoints **With** Dual-Banks*

The cancellation procedure depends on the TXPKTRDY flag value in the UDP_CSR:

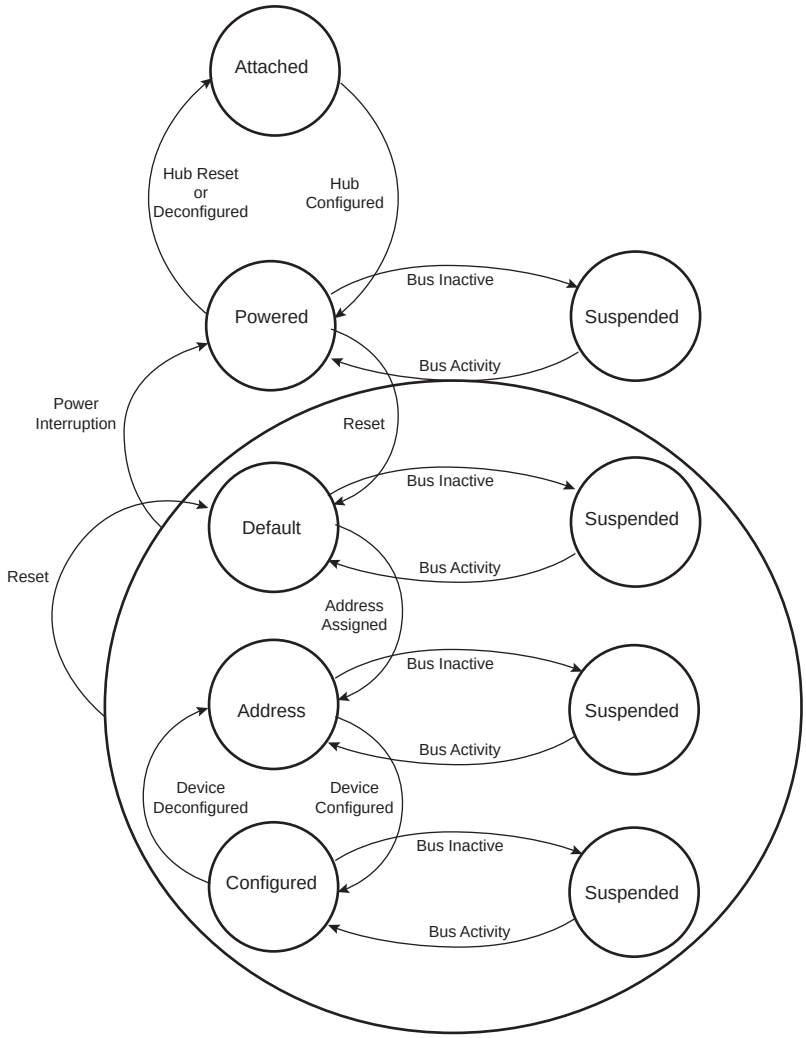
- TXPKTRDY is not set:
  - Reset the endpoint to clear the FIFO (pointers). (See Section 32.7.9 ”UDP Reset Endpoint Register”.)
- TXPKTRDY has already been set:
  - Clear TXPKTRDY and read it back until actually read at 0.
  - Set TXPKTRDY and read it back until actually read at 1.
  - Clear TXPKTRDY so that no packet is ready to be sent.
  - Reset the endpoint to clear the FIFO (pointers). (See Section 32.7.9 ”UDP Reset Endpoint Register”.)

© 2017 Microchip Technology Inc.

32.6.3 Controlling Device States

A USB device has several possible states. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0*.

Figure 32-14: USB Device State Diagram



Movement from one state to another depends on the USB bus state or on standard requests sent through control transactions via the default endpoint (endpoint 0).

After a period of bus inactivity, the USB device enters Suspend Mode. Accepting Suspend/Resume requests from the USB host is mandatory. Constraints in Suspend Mode are very strict for bus-powered applications; devices must not consume more than 2.5 mA on the USB bus.

While in Suspend Mode, the host may wake up a device by sending a resume signal (bus activity) or a USB device may send a wakeup request to the host, e.g., waking up a PC by moving a USB mouse.

The wakeup feature is not mandatory for all devices and must be negotiated with the host.

32.6.3.1 Not Powered State

Self powered devices can detect 5V VBUS using a PIO as described in the typical connection section. When the device is not connected to a host, device power consumption can be reduced by disabling MCK for the UDP, disabling UDPCK and disabling the transceiver. DDP and DDM lines are pulled down by 330 KΩ resistors.

32.6.3.2 Entering Attached State

When no device is connected, the USB DDP and DDM signals are tied to GND by 15 KΩ pull-down resistors integrated in the hub downstream ports. When a device is attached to a hub downstream port, the device connects a 1.5 KΩ pull-up resistor on DDP. The USB bus line goes into IDLE state, DDP is pulled up by the device 1.5 KΩ resistor to 3.3V and DDM is pulled down by the 15 KΩ resistor of the host.

To enable integrated pull-up, the PUON bit in the UDP_TXVC register must be set.

Warning: To write to the UDP_TXVC register, MCK clock must be enabled on the UDP. This is done in the PMC.

After pull-up connection, the device enters the powered state. In this state, the UDPCK and MCK must be enabled in the PMC. The transceiver can remain disabled.

32.6.3.3 From Powered State to Default State

After its connection to a USB host, the USB device waits for an end-of-bus reset. The unmaskable flag ENDBUSRES is set in the Interrupt Status Register (UDP_ISR) and an interrupt is triggered.

Once the ENDBUSRES interrupt has been triggered, the device enters Default State. In this state, the UDP software must:

- Enable the default endpoint, setting the EPEDS flag in the UDP_CSR0 and, optionally, enabling the interrupt for endpoint 0 by writing 1 to the Interrupt Enable Register (UDP_IER). The enumeration then begins by a control transfer.
- Configure the Interrupt Mask Register (UDP_IMR) which has been reset by the USB reset detection
- Enable the transceiver clearing the TXVDIS flag in the UDP_TXVC register.

In this state UDPCK and MCK must be enabled.

Warning: Each time an ENDBUSRES interrupt is triggered, the UDP_IMR and UDP_CSRs have been reset.

32.6.3.4 From Default State to Address State

After a set address standard device request, the USB host peripheral enters the address state.

Warning: Before the device enters in address state, it must achieve the Status IN transaction of the control transfer, i.e., the UDP device sets its new address once the TXCOMP flag in the UDP_CSR0 has been received and cleared.

To move to address state, the driver software sets the FADDEN flag in the Global State Register (UDP_GLB_STAT), sets its new address, and sets the FEN bit in the Function Address Register (UDP_FADDR).

32.6.3.5 From Address State to Configured State

Once a valid Set Configuration standard request has been received and acknowledged, the device enables endpoints corresponding to the current configuration. This is done by setting the EPEDS and EPTYPE fields in the UDP_CSRx and, optionally, enabling corresponding interrupts in the UDP_IER.

32.6.3.6 Entering in Suspend State

When a Suspend (no bus activity on the USB bus) is detected, the RXSUSP signal in the UDP_ISR is set. This triggers an interrupt if the corresponding bit is set in the UDP_IMR. This flag is cleared by writing to the Interrupt Clear Register (UDP_ICR) and the device then enters Suspend mode.

In this state bus powered devices must drain no more than 2.5 mA from the 5V VBUS. As an example, the microcontroller switches to slow clock, disables the PLL and main oscillator, and goes into Idle mode. It may also switch off other devices on the board.

The USB device peripheral clocks can be switched off. Resume event is asynchronously detected. MCK and UDPCK can be switched off in the PMC and the USB transceiver can be disabled by setting the TXVDIS bit in the UDP_TXVC register.

Warning: Read, write operations to the UDP registers are allowed only if MCK is enabled for the UDP peripheral. Switching off MCK for the UDP peripheral must be one of the last operations after writing to the UDP_TXVC register and acknowledging the RXSUSP.

32.6.3.7 Receiving a Host Resume

In suspend mode, a resume event on the USB bus line is detected asynchronously, transceiver and clocks are disabled (however the pull-up shall not be removed).

Once the resume is detected on the bus, the WAKEUP signal in the UDP_ISR is set. It may generate an interrupt if the corresponding bit in the UDP_IMR is set. This interrupt may be used to wake up the core, enable PLL and main oscillators and configure clocks.

Warning: Read, write operations to the UDP registers are allowed only if MCK is enabled for the UDP peripheral. MCK for the UDP must be enabled before clearing the WAKEUP bit in the UDP_ICR and clearing TXVDIS in the UDP_TXVC register.

© 2017 Microchip Technology Inc.

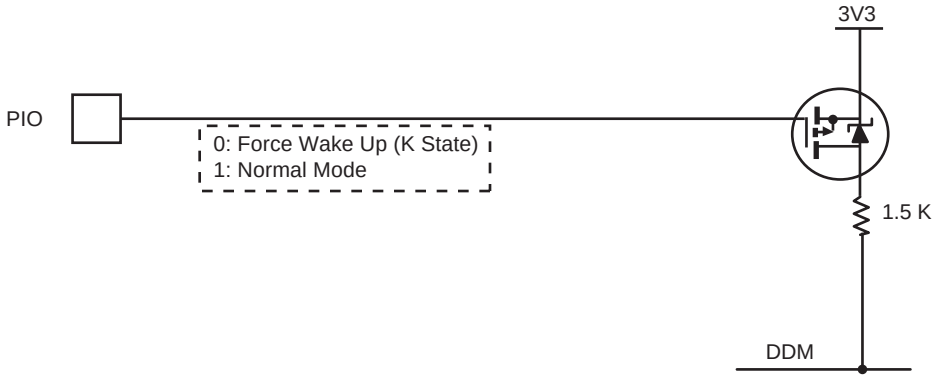
32.6.3.8 Sending a Device Remote Wakeup Request

In Suspend state it is possible to wake up the host sending an external resume.

- The device must wait at least 5 ms after being entered in suspend before sending an external resume.
- The device has 10 ms from the moment it starts to drain current and it forces a K state to resume the host.
- The device must force a K state from 1 to 15 ms to resume the host

To force a K state to the bus (DDM at 3.3V and DDP tied to GND), it is possible to use a transistor to connect a pull-up on DDM. The K state is obtained by disabling the pull-up on DDP and enabling the pull-up on DDM. This should be under the control of the application.

Figure 32-15: Board Schematic to Drive a K State



32.7 USB Device Port (UDP) User Interface

WARNING: The UDP peripheral clock in the PMC must be enabled before any read/write operations to the UDP registers, including the UDP_TXVC register.

Table 32-6: Register Mapping

Offset	Register	Name	Access	Reset
0x000	Frame Number Register	UDP_FRM_NUM	Read-only	0x0000_0000
0x004	Global State Register	UDP_GLB_STAT	Read/Write	0x0000_0010
0x008	Function Address Register	UDP_FADDR	Read/Write	0x0000_0100
0x00C	Reserved	–	–	–
0x010	Interrupt Enable Register	UDP_IER	Write-only	
0x014	Interrupt Disable Register	UDP_IDR	Write-only	
0x018	Interrupt Mask Register	UDP_IMR	Read-only	0x0000_1200
0x01C	Interrupt Status Register	UDP_ISR	Read-only	–(1)
0x020	Interrupt Clear Register	UDP_ICR	Write-only	
0x024	Reserved	–	–	–
0x028	Reset Endpoint Register	UDP_RST_EP	Read/Write	0x0000_0000
0x02C	Reserved	–	–	–
0x030	Endpoint Control and Status Register 0	UDP_CSR0	Read/Write	0x0000_0000
...	...	...	...	...
0x030 + 0x4 * 5	Endpoint Control and Status Register 5	UDP_CSR5	Read/Write	0x0000_0000
0x050	Endpoint FIFO Data Register 0	UDP_FDR0	Read/Write	–(1)
...	...	...	...	...
0x050 + 0x4 * 5	Endpoint FIFO Data Register 5	UDP_FDR5	Read/Write	–(1)
0x070	Reserved	–	–	–
0x074	Transceiver Control Register	UDP_TXVC(2)	Read/Write	0x0000_0100
0x078–0xFC	Reserved	–	–	–

Note 1: Reset values are not defined for UDP_ISR or UDP_FDRx. UDP_FDRs reflect Dual Port RAM memory locations which are not affected by any reset signals.

2: See Warning above Table 32-6.

© 2017 Microchip Technology Inc.

32.7.1 UDP Frame Number Register

Name:UDP_FRM_NUM

Address:0xF803C000

Access: Read-only

31	30	29	28	27	26	25	24
---	---	---	---	---	---	---	---
23	22	21	20	19	18	17	16
–	–	–	–	–	–	FRM_OK	FRM_ERR
15	14	13	12	11	10	9	8
–	–	–	–	–	FRM_NUM		
7	6	5	4	3	2	1	0
FRM_NUM							

FRM_NUM: Frame Number as Defined in the Packet Field Formats

This 11-bit value is incremented by the host on a per frame basis. This value is updated at each start of frame.

Value updated at the SOF_EOP (Start of Frame End of Packet).

FRM_ERR: Frame Error

This bit is set at SOF_EOP when the SOF packet is received containing an error.

This bit is reset upon receipt of SOF_PID.

FRM_OK: Frame OK

This bit is set at SOF_EOP when the SOF packet is received without any error.

This bit is reset upon receipt of SOF_PID (Packet Identification).

In the Interrupt Status Register, the SOF interrupt is updated upon receiving SOF_PID. This bit is set without waiting for EOP.

Note: In the 8-bit Register Interface, FRM_OK is bit 4 of FRM_NUM_H and FRM_ERR is bit 3 of FRM_NUM_L.

32.7.2 UDP Global State Register

Name:UDP_GLB_STAT

Address:0xF803C004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	RMWUPE	RSMINPR	ESR	CONFG	FADDEN

This register is used to get and set the device state as specified in Chapter 9 of the *USB Serial Bus Specification, Rev.2.0*.

FADDEN: Function Address Enable

Read:

0: Device is not in address state

1: Device is in address state

Write:

0: No effect, only a reset can bring back a device to the default state.

1: Sets device in address state. This occurs after a successful Set Address request. Beforehand, the UDP_FADDR must have been initialized with Set Address parameters. Set Address must complete the Status Stage before setting FADDEN. Refer to chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

CONFG: Configured

Read:

0: Device is not in configured state

1: Device is in configured state

Write:

0: Sets device in a non configured state

1: Sets device in configured state

The device is set in configured state when it is in address state and receives a successful Set Configuration request. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

© 2017 Microchip Technology Inc.

32.7.3 UDP Function Address Register

Name:UDP_FADDR

Address:0xF803C008

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	FEN
7	6	5	4	3	2	1	0
–	FADD						

FADD: Function Address Value

The Function Address Value must be programmed by firmware once the device receives a set address request from the host, and has achieved the status stage of the no-data control sequence. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information. After power up or reset, the function address value is set to 0.

FEN: Function Enable

Read:

0: Function endpoint disabled

1: Function endpoint enabled

Write:

0: Disables function endpoint

1: Default value

The Function Enable bit (FEN) allows the microcontroller to enable or disable the function endpoints. The microcontroller sets this bit after receipt of a reset from the host. Once this bit is set, the USB device is able to accept and transfer data packets from and to the host.

32.7.4 UDP Interrupt Enable Register

Name:UDP_IER

Address:0xF803C010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

EP0INT: Enable Endpoint 0 Interrupt

EP1INT: Enable Endpoint 1 Interrupt

EP2INT: Enable Endpoint 2Interrupt

EP3INT: Enable Endpoint 3 Interrupt

EP4INT: Enable Endpoint 4 Interrupt

EP5INT: Enable Endpoint 5 Interrupt

0: No effect

1: Enables corresponding Endpoint Interrupt

RXSUSP: Enable UDP Suspend Interrupt

0: No effect

1: Enables UDP Suspend Interrupt

RXRSM: Enable UDP Resume Interrupt

0: No effect

1: Enables UDP Resume Interrupt

SOFINT: Enable Start Of Frame Interrupt

0: No effect

1: Enables Start Of Frame Interrupt

WAKEUP: Enable UDP Bus Wakeup Interrupt

0: No effect

1: Enables UDP bus Interrupt

© 2017 Microchip Technology Inc.

32.7.5 UDP Interrupt Disable Register

Name:UDP_IDR

Address:0xF803C014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

EP0INT: Disable Endpoint 0 Interrupt

EP1INT: Disable Endpoint 1 Interrupt

EP2INT: Disable Endpoint 2 Interrupt

EP3INT: Disable Endpoint 3 Interrupt

EP4INT: Disable Endpoint 4 Interrupt

EP5INT: Disable Endpoint 5 Interrupt

0: No effect

1: Disables corresponding Endpoint Interrupt

RXSUSP: Disable UDP Suspend Interrupt

0: No effect

1: Disables UDP Suspend Interrupt

RXRSM: Disable UDP Resume Interrupt

0: No effect

1: Disables UDP Resume Interrupt

SOFINT: Disable Start Of Frame Interrupt

0: No effect

1: Disables Start Of Frame Interrupt

WAKEUP: Disable USB Bus Interrupt

0: No effect

1: Disables USB Bus Wakeup Interrupt

32.7.6 UDP Interrupt Mask Register

Name:UDP_IMR

Address:0xF803C018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	WAKEUP	BIT12	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

EP0INT: Mask Endpoint 0 Interrupt

EP1INT: Mask Endpoint 1 Interrupt

EP2INT: Mask Endpoint 2 Interrupt

EP3INT: Mask Endpoint 3 Interrupt

EP4INT: Mask Endpoint 4 Interrupt

EP5INT: Mask Endpoint 5 Interrupt

0: Corresponding Endpoint Interrupt is disabled

1: Corresponding Endpoint Interrupt is enabled

RXSUSP: Mask UDP Suspend Interrupt

0: UDP Suspend Interrupt is disabled

1: UDP Suspend Interrupt is enabled

RXRSM: Mask UDP Resume Interrupt.

0: UDP Resume Interrupt is disabled

1: UDP Resume Interrupt is enabled

SOFINT: Mask Start Of Frame Interrupt

0: Start of Frame Interrupt is disabled

1: Start of Frame Interrupt is enabled

BIT12: UDP_IMR Bit 12

Bit 12 of UDP_IMR cannot be masked and is always read at 1.

WAKEUP: USB Bus Wakeup Interrupt

0: USB Bus Wakeup Interrupt is disabled

1: USB Bus Wakeup Interrupt is enabled

Note: When the USB block is in suspend mode, the application may power down the USB logic. In this case, any USB HOST resume request that is made must be taken into account and, thus, the reset value of the RXRSM bit of the register UDP_IMR is enabled.

© 2017 Microchip Technology Inc.

32.7.7 UDP Interrupt Status Register

Name:UDP_ISR

Address:0xF803C01C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBUSRES	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

EP0INT: Endpoint 0 Interrupt Status

EP1INT: Endpoint 1 Interrupt Status

EP2INT: Endpoint 2 Interrupt Status

EP3INT: Endpoint 3 Interrupt Status

EP4INT: Endpoint 4 Interrupt Status

EP5INT: Endpoint 5 Interrupt Status

0: No Endpointx Interrupt pending

1: Endpointx Interrupt has been raised

Several signals can generate this interrupt. The reason can be found by reading UDP_CSR0:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP0INT is a sticky bit. Interrupt remains valid until EP0INT is cleared by writing in the corresponding UDP_CSR0 bit.

RXSUSP: UDP Suspend Interrupt Status

0: No UDP Suspend Interrupt pending

1: UDP Suspend Interrupt has been raised

The USB device sets this bit when it detects no activity for 3 ms. The USB device enters Suspend mode.

RXRSM: UDP Resume Interrupt Status

0: No UDP Resume Interrupt pending

1: UDP Resume Interrupt has been raised

The USB device sets this bit when a UDP resume signal is detected at its port.

After reset, the state of this bit is undefined, the application must clear this bit by setting the RXRSM flag in the UDP_ICR.

SOFINT: Start of Frame Interrupt Status

0: No Start of Frame Interrupt pending

1: Start of Frame Interrupt has been raised

This interrupt is raised each time a SOF token has been detected. It can be used as a synchronization signal by using isochronous endpoints.

ENDBUSRES: End of BUS Reset Interrupt Status

0: No End of Bus Reset Interrupt pending

1: End of Bus Reset Interrupt has been raised

This interrupt is raised at the end of a UDP reset sequence. The USB device must prepare to receive requests on the endpoint 0. The host starts the enumeration, then performs the configuration.

WAKEUP: UDP Resume Interrupt Status

0: No Wakeup Interrupt pending

1: A Wakeup Interrupt (USB Host Sent a RESUME or RESET) occurred since the last clear.

After reset the state of this bit is undefined; the application must clear this bit by setting the WAKEUP flag in the UDP_ICR.

© 2017 Microchip Technology Inc.

32.7.8 UDP Interrupt Clear Register

Name:UDP_ICR

Address:0xF803C020

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBUSRES	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

RXSUSP: Clear UDP Suspend Interrupt

0: No effect

1: Clears UDP Suspend Interrupt

RXRSM: Clear UDP Resume Interrupt

0: No effect

1: Clears UDP Resume Interrupt

SOFINT: Clear Start Of Frame Interrupt

0: No effect

1: Clears Start Of Frame Interrupt

ENDBUSRES: Clear End of Bus Reset Interrupt

0: No effect

1: Clears End of Bus Reset Interrupt

WAKEUP: Clear Wakeup Interrupt

0: No effect

1: Clears Wakeup Interrupt

32.7.9 UDP Reset Endpoint Register

Name:UDP_RST_EP

Address:0xF803C028

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
		EP5	EP4	EP3	EP2	EP1	EP0

EP0: Reset Endpoint 0

EP1: Reset Endpoint 1

EP2: Reset Endpoint 2

EP3: Reset Endpoint 3

EP4: Reset Endpoint 4

EP5: Reset Endpoint 5

This flag is used to reset the FIFO associated with the endpoint and the bit RXBYTECOUNT in the UDP_CSRx. It also resets the data toggle to DATA0. It is useful after removing a HALT condition on a BULK endpoint. Refer to Chapter 5.8.5 in the *USB Serial Bus Specification, Rev.2.0*.

Warning: This flag must be cleared at the end of the reset. It does not clear UDP_CSRx flags.

0: No reset

1: Forces the corresponding endpoint FIF0 pointers to 0, therefore RXBYTECNT field is read at 0 in UDP_CSRx

Resetting the endpoint is a two-step operation:

1. Set the corresponding EPx field.
2. Clear the corresponding EPx field.

© 2017 Microchip Technology Inc.

32.7.10 UDP Endpoint Control and Status Register (CONTROL_BULK)

Name:UDP_CSRx [x = 0..5] (CONTROL_BULK)

Address:0xF803C030

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	RXBYTECNT		

23	22	21	20	19	18	17	16
RXBYTECNT							

15	14	13	12	11	10	9	8
EPEDS	–	–	–	DTGLE	EPTYPE		

7	6	5	4	3	2	1	0
DIR	RX_DATA_BK1	FORCESTALL	TXPKTRDY	STALLSENT	RXSETUP	RX_DATA_ BK0	TXCOMP

WARNING: Due to synchronization between MCK and UDPCK, the software application must wait for the end of the write operation before executing another write by polling the bits which must be set/cleared.

As an example, to perform a control operation on the endpoint without modifying the status flags while accessing the control bits and fields of this register, the status flag bits must first be defined with the “No effect” value ‘1’. Once the overall UDP_CSR value is defined, the register can be written and then the synchronization wait procedure must be executed.

```
//! Bitmap for all status bits in CSR that are not affected by a value 1.
#define UDP_REG_NO_EFFECT_1_ALL (UDP_CSR_RX_DATA_BK0 |\
                                 UDP_CSR_RX_DATA_BK1 |\
                                 UDP_CSR_STALLSENT   |\
                                 UDP_CSR_RXSETUP     |\
                                 UDP_CSR_TXCOMP)



/*! Sets specified bit(s) in the UDP_CSR.
*  \param ep   Endpoint number.
*  \param bits Bitmap to set to 1.
*/

#define udp_set_csr(ep, bits)                           \
    do {                                        \
        volatile uint32_t reg;                  \
        volatile uint32_t nop_count;            \
        reg  = UDP->UDP_CSR[ep];                \
        reg |= UDP_REG_NO_EFFECT_1_ALL;         \
        reg |= (bits);                          \
        UDP->UDP_CSR[ep] = reg;                 \
        for (nop_count = 0; nop_count < 20; nop_count ++) {\
            __NOP();                            \
        }                                       \
    } while (0)
```

```
/*! Clears specified bit(s) in the UDP_CSR.
*  \param ep   Endpoint number.
*  \param bits Bitmap to set to 0.
*/
#define udp_clear_csr(ep, bits)                          \
    do {                                                 \
        volatile uint32_t reg;                           \
        volatile uint32_t nop_count;                     \
        reg  = UDP->UDP_CSR[ep];                         \
        reg |= UDP_REG_NO_EFFECT_1_ALL;                  \
        reg &= ~(bits);                                  \
        UDP->UDP_CSR[ep] = reg;                          \
        for (nop_count = 0; nop_count < 20; nop_count ++) {\
            __NOP();                                     \
        }                                                \
    } while (0)
```

In a preemptive environment, set or clear the flag and wait for a time of 1 UDPCK clock cycle and 1peripheral clock cycle. However, RX_DATA_BK0, TXPKTRDY, RX_DATA_BK1 require wait times of 3 UDPCK clock cycles and 5 peripheral clock cycles before accessing DPR.

TXCOMP: Generates an IN Packet with Data Previously Written in the DPR

This flag generates an interrupt while it is set to one.

Write (cleared by the firmware):

0: Clear the flag, clear the interrupt

1: No effect

Read (Set by the USB peripheral):

0: Data IN transaction has not been acknowledged by the Host

1: Data IN transaction is achieved, acknowledged by the Host

After having issued a Data IN transaction setting TXPKTRDY, the device firmware waits for TXCOMP to be sure that the host has acknowledged the transaction.

RX_DATA_BK0: Receive Data Bank 0

This flag generates an interrupt while it is set to one.

Write (cleared by the firmware):

0: Notify USB peripheral device that data have been read in the FIFO's Bank 0.

1: To leave the read value unchanged.

Read (Set by the USB peripheral):

0: No data packet has been received in the FIFO's Bank 0.

1: A data packet has been received, it has been stored in the FIFO's Bank 0.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to the microcontroller memory. The number of bytes received is available in RXBYTCENT field. Bank 0 FIFO values are read through the UDP_FDRx. Once a transfer is done, the device firmware must release Bank 0 to the USB peripheral device by clearing RX_DATA_BK0.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

© 2017 Microchip Technology Inc.

RXSETUP: Received Setup

This flag generates an interrupt while it is set to one.

Read:

0: No setup packet available.

1: A setup data packet has been sent by the host and is available in the FIFO.

Write:

0: Device firmware notifies the USB peripheral device that it has read the setup data in the FIFO.

1: No effect.

This flag is used to notify the USB device firmware that a valid Setup data packet has been sent by the host and successfully received by the USB device. The USB device firmware may transfer Setup data from the FIFO by reading the UDP_FDRx to the microcontroller memory. Once a transfer has been done, RXSETUP must be cleared by the device firmware.

Ensuing Data OUT transaction is not accepted while RXSETUP is set.

STALLSENT: Stall Sent

This flag generates an interrupt while it is set to one.

This ends a STALL handshake.

Read:

0: Host has not acknowledged a stall

1: Host has acknowledged the stall

Write:

0: Resets the STALLSENT flag, clears the interrupt

1: No effect

This is mandatory for the device firmware to clear this flag. Otherwise the interrupt remains.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

TXPKTRDY: Transmit Packet Ready

This flag is cleared by the USB device.

This flag is set by the USB device firmware.

Read:

0: There is no data to send.

1: The data is waiting to be sent upon reception of token IN.

Write:

0: Can be used in the procedure to cancel transmission data. (See Section 32.6.2.5 ”Transmit Data Cancellation”)

1: A new data payload has been written in the FIFO by the firmware and is ready to be sent.

This flag is used to generate a Data IN transaction (device to host). Device firmware checks that it can write a data payload in the FIFO, checking that TXPKTRDY is cleared. Transfer to the FIFO is done by writing in the UDP_FDRx. Once the data payload has been transferred to the FIFO, the firmware notifies the USB device setting TXPKTRDY to one. USB bus transactions can start. TXCOMP is set once the data payload has been received by the host.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

FORCESTALL: Force Stall (used by Control, Bulk and Isochronous Endpoints)

Read:

0: Normal state

1: Stall state

Write:

0: Return to normal state

1: Send STALL to the host

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

Control endpoints: During the data stage and status stage, this bit indicates that the microcontroller cannot complete the request.

Bulk and interrupt endpoints: This bit notifies the host that the endpoint is halted.

The host acknowledges the STALL, device firmware is notified by the STALLSENT flag.

RX_DATA_BK1: Receive Data Bank 1 (only used by endpoints with ping-pong attributes)

This flag generates an interrupt while it is set to one.

Write (cleared by the firmware):

0: Notifies USB device that data have been read in the FIFO's Bank 1.

1: To leave the read value unchanged.

Read (Set by the USB peripheral):

0: No data packet has been received in the FIFO's Bank 1.

1: A data packet has been received, it has been stored in FIFO's Bank 1.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to microcontroller memory. The number of bytes received is available in RXBYTECNT field. Bank 1 FIFO values are read through UDP_FDRx. Once a transfer is done, the device firmware must release Bank 1 to the USB device by clearing RX_DATA_BK1.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

DIR: Transfer Direction (only available for control endpoints) (Read/Write)

0: Allows Data OUT transactions in the control data stage.

1: Enables Data IN transactions in the control data stage.

Refer to Chapter 8.5.3 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the control data stage.

This bit must be set before UDP_CSRx/RXSETUP is cleared at the end of the setup stage. According to the request sent in the setup data packet, the data stage is either a device to host (DIR = 1) or host to device (DIR = 0) data transfer. It is not necessary to check this bit to reverse direction for the status stage.

EPTYPE: Endpoint Type (Read/Write)

Value	Name	Description
0	CTRL	Control
1	ISO_OUT	Isochronous OUT
2	BULK_OUT	Bulk OUT
3	INT_OUT	Interrupt OUT
4	–	Reserved
5	ISO_IN	Isochronous IN
6	BULK_IN	Bulk IN
7	INT_IN	Interrupt IN

DTGLE: Data Toggle (Read-only)

0: Identifies DATA0 packet

1: Identifies DATA1 packet

Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on DATA0, DATA1 packet definitions.

© 2017 Microchip Technology Inc.

EPEDS: Endpoint Enable Disable

Read:

0: Endpoint disabled

1: Endpoint enabled

Write:

0: Disables endpoint

1: Enables endpoint

Control endpoints are always enabled. Reading or writing this field has no effect on control endpoints.

Note: After reset, all endpoints are configured as control endpoints (zero).

RXBYTECNT: Number of Bytes Available in the FIFO (Read-only)

When the host sends a data packet to the device, the USB device stores the data in the FIFO and notifies the microcontroller. The microcontroller can load the data from the FIFO by reading RXBYTECENT bytes in the UDP_FDRx.

© 2017 Microchip Technology Inc.

32.7.11 UDP Endpoint Control and Status Register (ISOCHRONOUS)

Name:UDP_CSRx [x = 0..5] (ISOCHRONOUS)

Address:0xF803C030

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	RXBYTECNT		

23	22	21	20	19	18	17	16
RXBYTECNT							

15	14	13	12	11	10	9	8
EPEDS	–	–	–	DTGLE	EPTYPE		

7	6	5	4	3	2	1	0
DIR	RX_DATA_BK1	FORCESTALL	TXPKTRDY	ISOERROR	RXSETUP	RX_DATA_BK0	TXCOMP

TXCOMP: Generates an IN Packet with Data Previously Written in the DPR

This flag generates an interrupt while it is set to one.

Write (cleared by the firmware):

0: Clear the flag, clear the interrupt.

1: No effect.

Read (Set by the USB peripheral):

0: Data IN transaction has not been acknowledged by the Host.

1: Data IN transaction is achieved, acknowledged by the Host.

After having issued a Data IN transaction setting TXPKTRDY, the device firmware waits for TXCOMP to be sure that the host has acknowledged the transaction.

RX_DATA_BK0: Receive Data Bank 0

This flag generates an interrupt while it is set to one.

Write (cleared by the firmware):

0: Notify USB peripheral device that data have been read in the FIFO's Bank 0.

1: To leave the read value unchanged.

Read (Set by the USB peripheral):

0: No data packet has been received in the FIFO's Bank 0.

1: A data packet has been received, it has been stored in the FIFO's Bank 0.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to the microcontroller memory. The number of bytes received is available in RXBYTCENT field. Bank 0 FIFO values are read through the UDP_FDRx. Once a transfer is done, the device firmware must release Bank 0 to the USB peripheral device by clearing RX_DATA_BK0.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

RXSETUP: Received Setup

This flag generates an interrupt while it is set to one.

Read:

0: No setup packet available.

1: A setup data packet has been sent by the host and is available in the FIFO.

© 2017 Microchip Technology Inc.

Write:

0: Device firmware notifies the USB peripheral device that it has read the setup data in the FIFO.

1: No effect.

This flag is used to notify the USB device firmware that a valid Setup data packet has been sent by the host and successfully received by the USB device. The USB device firmware may transfer Setup data from the FIFO by reading the UDP_FDRx to the microcontroller memory. Once a transfer has been done, RXSETUP must be cleared by the device firmware.

Ensuing Data OUT transaction is not accepted while RXSETUP is set.

ISOERROR: A CRC error has been detected in an isochronous transfer

This flag generates an interrupt while it is set to one.

Read:

0: No error in the previous isochronous transfer.

1: CRC error has been detected, data available in the FIFO are corrupted.

Write:

0: Resets the ISOERROR flag, clears the interrupt.

1: No effect.

TXPKTRDY: Transmit Packet Ready

This flag is cleared by the USB device.

This flag is set by the USB device firmware.

Read:

0: There is no data to send.

1: The data is waiting to be sent upon reception of token IN.

Write:

0: Can be used in the procedure to cancel transmission data. (See Section 32.6.2.5 ”Transmit Data Cancellation”)

1: A new data payload has been written in the FIFO by the firmware and is ready to be sent.

This flag is used to generate a Data IN transaction (device to host). Device firmware checks that it can write a data payload in the FIFO, checking that TXPKTRDY is cleared. Transfer to the FIFO is done by writing in the UDP_FDRx. Once the data payload has been transferred to the FIFO, the firmware notifies the USB device setting TXPKTRDY to one. USB bus transactions can start. TXCOMP is set once the data payload has been received by the host.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

FORCESTALL: Force Stall (used by Control, Bulk and Isochronous Endpoints)

Read:

0: Normal state.

1: Stall state.

Write:

0: Return to normal state.

1: Send STALL to the host.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

Control endpoints: During the data stage and status stage, this bit indicates that the microcontroller cannot complete the request.

Bulk and interrupt endpoints: This bit notifies the host that the endpoint is halted.

The host acknowledges the STALL, device firmware is notified by the STALLSENT flag.

RX_DATA_BK1: Receive Data Bank 1 (only used by endpoints with ping-pong attributes)

This flag generates an interrupt while it is set to one.

Write (cleared by the firmware):

0: Notifies USB device that data have been read in the FIFO's Bank 1.

1: To leave the read value unchanged.

Read (set by the USB peripheral):

0: No data packet has been received in the FIFO's Bank 1.

1: A data packet has been received, it has been stored in FIFO's Bank 1.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to microcontroller memory. The number of bytes received is available in RXBYTECNT field. Bank 1 FIFO values are read through UDP_FDRx. Once a transfer is done, the device firmware must release Bank 1 to the USB device by clearing RX_DATA_BK1.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

DIR: Transfer Direction (only available for control endpoints) (Read/Write)

0: Allows Data OUT transactions in the control data stage.

1: Enables Data IN transactions in the control data stage.

Refer to Chapter 8.5.3 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the control data stage.

This bit must be set before UDP_CSRx/RXSETUP is cleared at the end of the setup stage. According to the request sent in the setup data packet, the data stage is either a device to host (DIR = 1) or host to device (DIR = 0) data transfer. It is not necessary to check this bit to reverse direction for the status stage.

EPTYPE: Endpoint Type (Read/Write)

Value	Name	Description
0	CTRL	Control
1	ISO_OUT	Isochronous OUT
2	BULK_OUT	Bulk OUT
3	INT_OUT	Interrupt OUT
4	–	Reserved
5	ISO_IN	Isochronous IN
6	BULK_IN	Bulk IN
7	INT_IN	Interrupt IN

DTGLE: Data Toggle (Read-only)

0: Identifies DATA0 packet

1: Identifies DATA1 packet

Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on DATA0, DATA1 packet definitions.

EPEDS: Endpoint Enable Disable

Read:

0: Endpoint disabled

1: Endpoint enabled

Write:

0: Disables endpoint

1: Enables endpoint

Control endpoints are always enabled. Reading or writing this field has no effect on control endpoints.

Note: After reset, all endpoints are configured as control endpoints (zero).

© 2017 Microchip Technology Inc.

RXBYTECNT: Number of Bytes Available in the FIFO (Read-only)

When the host sends a data packet to the device, the USB device stores the data in the FIFO and notifies the microcontroller. The microcontroller can load the data from the FIFO by reading RXBYTECENT bytes in the UDP_FDRx.

32.7.12 UDP FIFO Data Register

Name:UDP_FDRx [x = 0..5]

Address:0xF803C050

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
FIFO_DATA							

FIFO_DATA: FIFO Data Value

The microcontroller can push or pop values in the FIFO through this register.

RXBYTECNT in the corresponding UDP_CSRx is the number of bytes to be read from the FIFO (sent by the host).

The maximum number of bytes to write is fixed by the Max Packet Size in the Standard Endpoint Descriptor. It can not be more than the physical memory size associated to the endpoint. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information.

© 2017 Microchip Technology Inc.

32.7.13 UDP Transceiver Control Register

Name:UDP_TXVC

Address:0xF803C074

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PUON	TXVDIS
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

WARNING: The UDP peripheral clock in the PMC must be enabled before any read/write operations to the UDP registers including the UDP_TXVC register.

TXVDIS: Transceiver Disable

When UDP is disabled, power consumption can be reduced significantly by disabling the embedded transceiver. This can be done by setting TXVDIS bit.

To enable the transceiver, TXVDIS must be cleared.

PUON: Pull-up On

0: The 1.5KΩ integrated pull-up on DDP is disconnected.

1: The 1.5 KΩ integrated pull-up on DDP is connected.

NOTE: If the USB pull-up is not connected on DDP, the user should not write in any UDP register other than the UDP_TXVC register. This is because if DDP and DDM are floating at 0, or pulled down, then SE0 is received by the device with the consequence of a USB Reset.

33. USB Host Port (UHP)

33.1 Description

The USB Host Port (UHP) interfaces the USB with the host application. It handles Open HCI protocol (Open Host Controller Interface) as well as USB v2.0 Full-speed and Low-speed protocols.

The USB Host Port integrates a root hub and transceivers on downstream ports. It provides several high-speed half-duplex serial communication ports at a baud rate of 12 Mbit/s. Up to 127 USB devices (printer, camera, mouse, keyboard, disk, etc.) and the USB hub can be connected to the USB host in the USB “tiered star” topology.

The USB Host Port controller is fully compliant with the OpenHCI specification. The USB Host Port User Interface (registers description) can be found in the Open HCI Rev 1.0 Specification le on www.hp.com. The standard OHCI USB stack driver can be easily ported to Microchip’s architecture in the same way all existing class drivers run without hardware specialization.

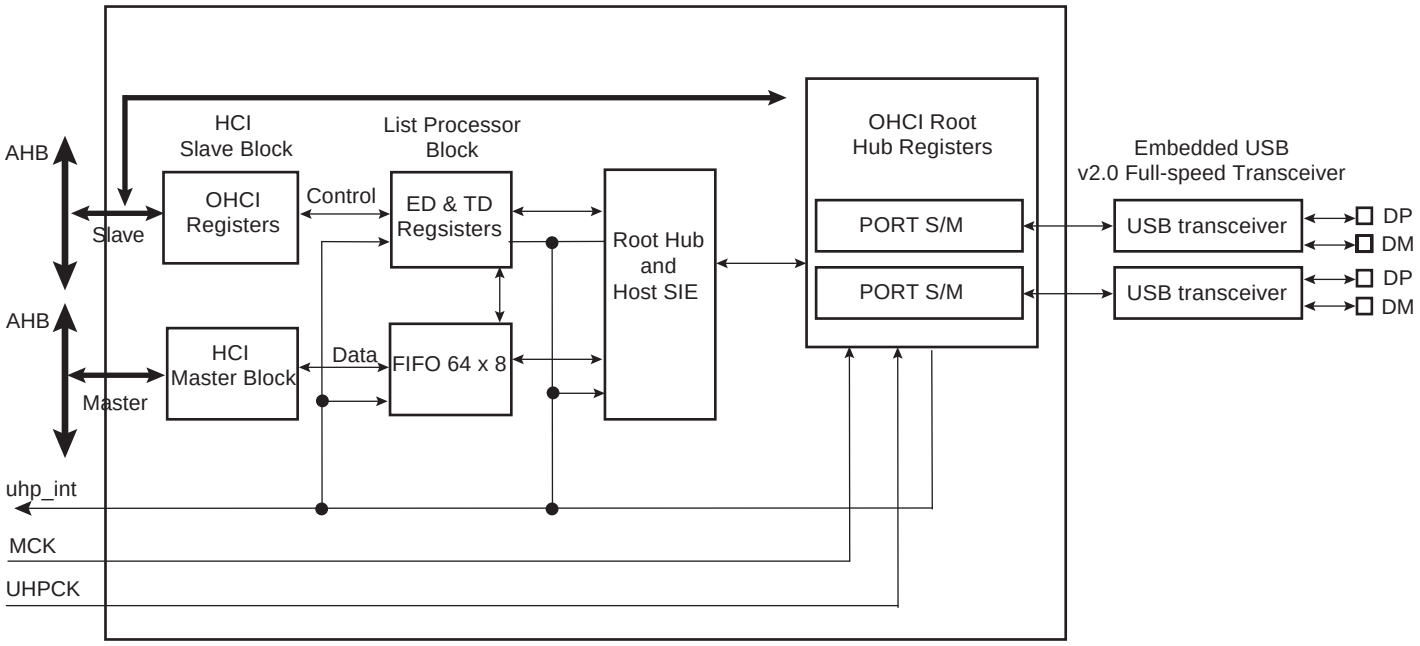
This means that all standard class devices are automatically detected and available to the user application. As an example, integrating an HID (Human Interface Device) class driver provides a plug & play feature for all USB keyboards and mouses.

33.2 Embedded Characteristics

- Compliant with OpenHCI Rev 1.0 Specification
- Compliant with USB V2.0 Full-speed and Low-speed Specification
- Supports Both Low-speed 1.5 Mbps and Full-speed 12 Mbps USB devices
- Root Hub Integrated with 1 Downstream USB Ports
- Embedded USB Transceivers
- Supports Power Management

33.3 Block Diagram

Figure 33-1: Block Diagram



Access to the USB host operational registers is achieved through the AHB bus slave interface. The OpenHCI host controller initializes master DMA transfers through the ASB bus master interface as follows:

- Fetches endpoint descriptors and transfer descriptors
- Access to endpoint data from system memory
- Access to the HC communication area
- Write status and retire transfer Descriptor

© 2017 Microchip Technology Inc.

Memory access errors (abort, misalignment) lead to an “UnrecoverableError” indicated by the corresponding flag in the host controller operational registers.

The USB root hub is integrated in the USB host. Several USB downstream ports are available. The number of downstream ports can be determined by the software driver reading the root hub’s operational registers. Device connection is automatically detected by the USB host port logic.

USB physical transceivers are integrated in the product and driven by the root hub’s ports.

Over current protection on ports can be activated by the USB host controller. Microchip’s standard product does not dedicate pads to external over current protection.

33.4 Product Dependencies

33.4.1 I/O Lines

DPs and DMs are not controlled by any PIO controllers. The embedded USB physical transceivers are controlled by the USB host controller.

33.4.2 Power Management

The USB host controller requires a 48 MHz clock. This clock must be generated by a PLL with a correct accuracy of ± 0.25%.

Thus the USB device peripheral receives two clocks from the Power Management Controller (PMC): the master clock MCK used to drive the peripheral user interface (MCK domain) and the UHPCLK 48 MHz clock used to interface with the bus USB signals (Recovered 12 MHz domain).

33.4.3 Interrupt

The USB host interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling USB host interrupts requires programming the AIC before configuring the UHP.

33.5 Functional Description

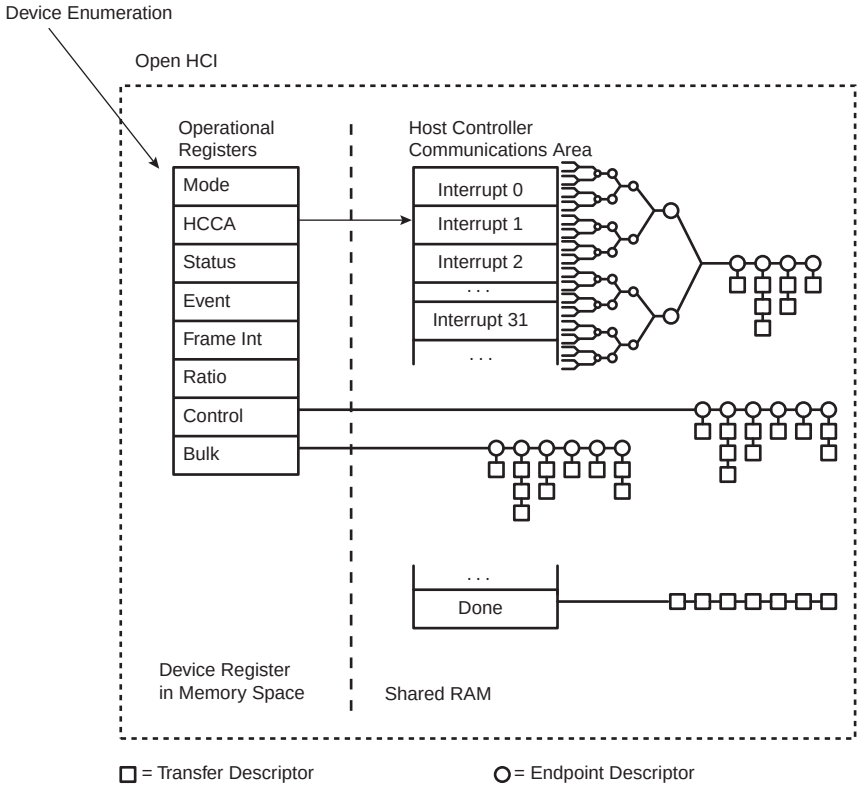
Refer to the Open Host Controller Interface Specification for USB Release 1.0.a.

33.5.1 Host Controller Interface

There are two communication channels between the Host Controller and the Host Controller Driver. The first channel uses a set of operational registers located on the USB Host Controller. The Host Controller is the target for all communications on this channel. The operational registers contain control, status and list pointer registers. They are mapped in the memory mapped area. Within the operational register set there is a pointer to a location in the processor address space named the Host Controller Communication Area (HCCA). The HCCA is the second communication channel. The host controller is the master for all communication on this channel. The HCCA contains the head pointers to the interrupt Endpoint Descriptor lists, the head pointer to the done queue and status information associated with start-of-frame processing.

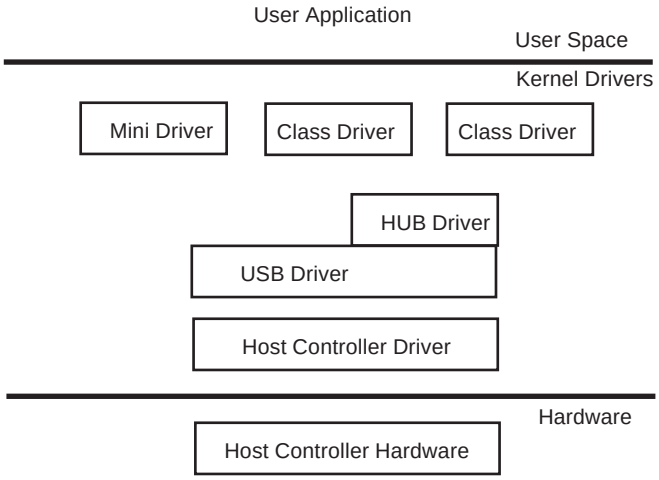
The basic building blocks for communication across the interface are Endpoint Descriptors (ED, 4 double words) and Transfer Descriptors (TD, 4 or 8 double words). The host controller assigns an Endpoint Descriptor to each endpoint in the system. A queue of Transfer Descriptors is linked to the Endpoint Descriptor for the specific endpoint.

Figure 33-2: USB Host Communication



33.5.2 Host Controller Driver

Figure 33-3: USB Host Drivers



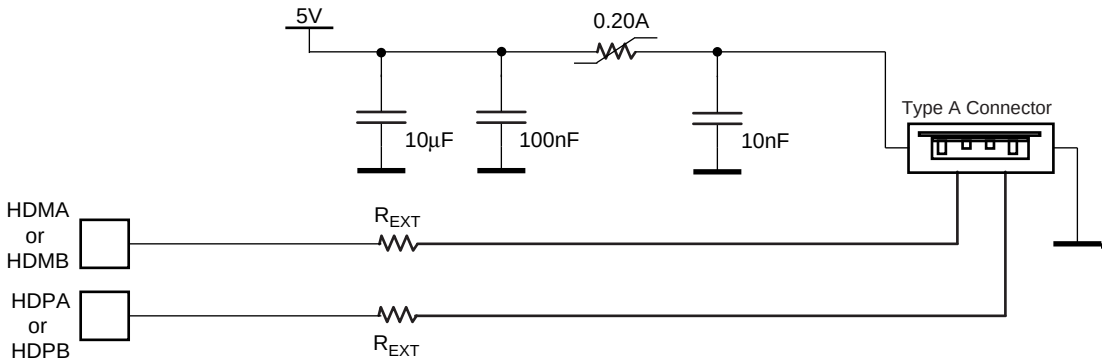
USB Handling is done through several layers as follows:

- Host controller hardware and serial engine: Transmits and receives USB data on the bus.
- Host controller driver: Drives the Host controller hardware and handles the USB protocol.
- USB Bus driver and hub driver: Handles USB commands and enumeration. Offers a hardware independent interface.
- Mini driver: Handles device specific commands.
- Class driver: Handles standard devices. This acts as a generic driver for a class of devices, for example the HID driver.

© 2017 Microchip Technology Inc.

33.6 Typical Connection

Figure 33-4: Board Schematic to Interface UHP Device Controller



A termination serial resistor must be connected to HDP and HDM. The resistor value is defined in the electrical specification of the product (R_{EXT}).

34. High Speed Multimedia Card Interface (HSMCI)

34.1 Description

The High Speed Multimedia Card Interface (HSMCI) supports the MultiMedia Card (MMC) Specification V4.3, the SD Memory Card Specification V2.0, the SDIO V2.0 specification and CE-ATA V1.1.

The HSMCI includes a command register, response registers, data registers, timeout counters and error detection logic that automatically handle the transmission of commands and, when required, the reception of the associated responses and data with a limited processor overhead.

The HSMCI supports stream, block and multi block data read and write, and is compatible with the DMA Controller (DMAC), minimizing processor intervention for large buffer transfers.

The HSMCI operates at a rate of up to Master Clock divided by 2 and supports the interfacing of 1 slot(s). Each slot may be used to interface with a High Speed MultiMedia Card bus (up to 30 Cards) or with an SD Memory Card. A bit field in the SD Card Register performs this selection.

The SD Memory Card communication is based on a 9-pin interface (clock, command, four data and three power lines) and the High Speed MultiMedia Card on a 7-pin interface (clock, command, one data, three power lines and one reserved for future use).

The SD Memory Card interface also supports High Speed MultiMedia Card operations. The main differences between SD and High Speed MultiMedia Cards are the initialization process and the bus topology.

HSMCI fully supports CE-ATA Revision 1.1, built on the MMC System Specification v4.0. The module includes dedicated hardware to issue the command completion signal and capture the host command completion signal disable.

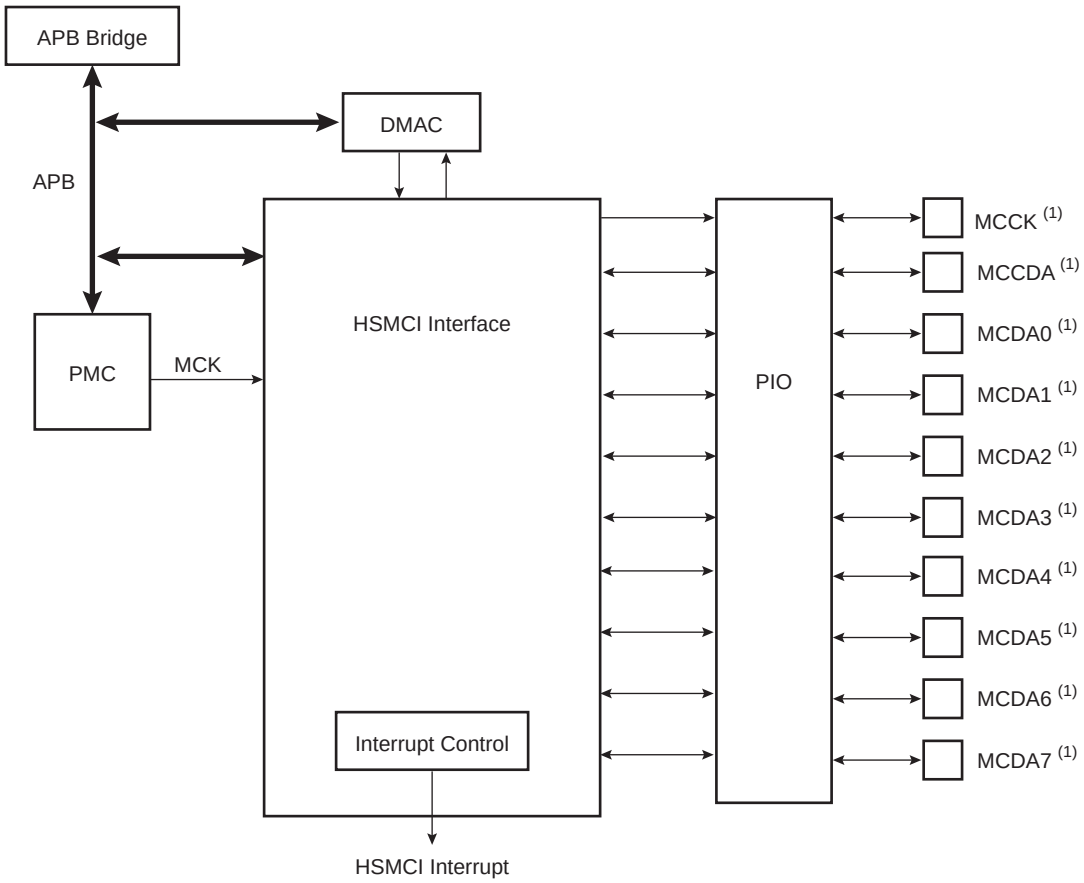
34.2 Embedded Characteristics

- Compatible with MultiMedia Card Specification Version 4.3
- Compatible with SD Memory Card Specification Version 2.0
- Compatible with SDIO Specification Version 2.0
- Compatible with CE-ATA Specification 1.1
- Cards Clock Rate Up to Master Clock Divided by 2
- Boot Operation Mode Support
- High Speed Mode Support
- Embedded Power Management to Slow Down Clock Rate When Not Used
- Supports 1 Multiplexed Slot(s)
  - Each Slot for either a High Speed MultiMedia Card Bus (Up to 30 Cards) or an SD Memory Card
- Support for Stream, Block and Multi-block Data Read and Write
- Supports Connection to DMA Controller (DMAC)
  - Minimizes Processor Intervention for Large Buffer Transfers
- Built in FIFO (from 16 to 256 bytes) with Large Memory Aperture Supporting Incremental Access
- Support for CE-ATA Completion Signal Disable Command
- Protection Against Unexpected Modification On-the-Fly of the Configuration Registers

© 2017 Microchip Technology Inc.

34.3 Block Diagram

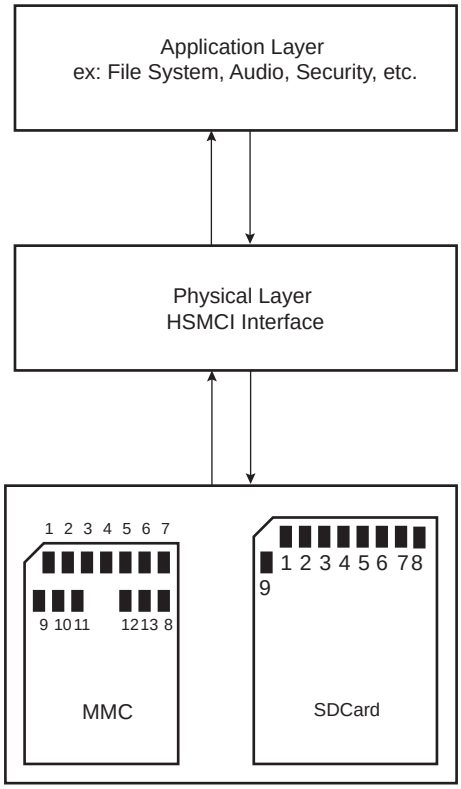
Figure 34-1: Block Diagram (8-bit configuration)



Note 1: When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

34.4 Application Block Diagram

Figure 34-2: Application Block Diagram



34.5 Pin Name List

Table 34-1: I/O Lines Description for 8-bit Configuration

Pin Name[1]	Pin Description	Type[2]	Comments
MCCDA	Command/response	I/O/PP/OD	CMD of an MMC or SDCard/SDIO
MCCK	Clock	I/O	CLK of an MMC or SD Card/SDIO
MCDA0–MCDA7	Data 0..7 of Slot A	I/O/PP	DAT[0..7] of an MMC DAT[0..3] of an SD Card/SDIO

Note 1: When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCCDB to HSMCIx_CDB, MCCDC to HSMCIx_CDC, MCCDD to HSMCIx_CDD, MCDAy to HSMCIx_DAy, MCDBy to HSMCIx_DBy, MCDCy to HSMCIx_DCy, MCDDy to HSMCIx_DDy.

2: I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

© 2017 Microchip Technology Inc.

34.6 Product Dependencies

34.6.1 I/O Lines

The pins used for interfacing the High Speed MultiMedia Cards or SD Cards are multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the peripheral functions to HSMCI pins.

Table 34-2: I/O Lines

Instance	Signal	I/O Line	Peripheral
HSMCI	MCCDA	PA16	A
HSMCI	MCCK	PA17	A
HSMCI	MCDA0	PA15	A
HSMCI	MCDA1	PA18	A
HSMCI	MCDA2	PA19	A
HSMCI	MCDA3	PA20	A
HSMCI	MCDA4	PA11	B
HSMCI	MCDA5	PA12	B
HSMCI	MCDA6	PA13	B
HSMCI	MCDA7	PA14	B

34.6.2 Power Management

The HSMCI is clocked through the Power Management Controller (PMC), so the programmer must first configure the PMC to enable the HSMCI clock.

34.6.3 Interrupt Sources

The HSMCI has an interrupt line connected to the interrupt controller.

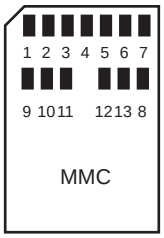
Handling the HSMCI interrupt requires programming the interrupt controller before configuring the HSMCI.

Table 34-3: Peripheral IDs

Instance	ID
HSMCI	12

34.7 Bus Topology

Figure 34-3: High Speed MultiMedia Memory Card Bus Topology



The High Speed MultiMedia Card communication is based on a 13-pin serial bus interface. It has three communication lines and four supply lines.

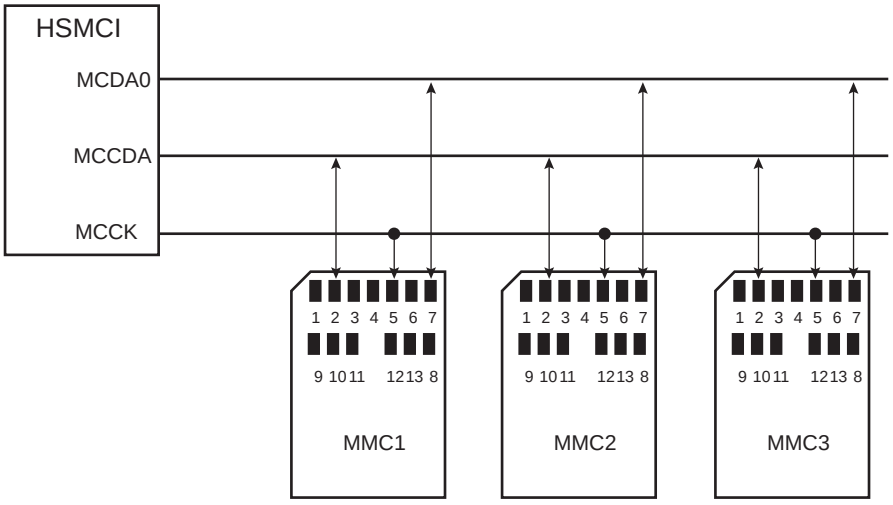
Table 34-4: Bus Topology

Pin Number	Name	Type[1]	Description	HSMCI Pin Name[2] (Slot z)
1	DAT[3]	I/O/PP	Data	MCDz3
2	CMD	I/O/PP/OD	Command/response	MCCDz
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	MCCK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data 0	MCDz0
8	DAT[1]	I/O/PP	Data 1	MCDz1
9	DAT[2]	I/O/PP	Data 2	MCDz2
10	DAT[4]	I/O/PP	Data 4	MCDz4
11	DAT[5]	I/O/PP	Data 5	MCDz5
12	DAT[6]	I/O/PP	Data 6	MCDz6
13	DAT[7]	I/O/PP	Data 7	MCDz7

Note 1: I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

2: When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

Figure 34-4: MMC Bus Connections (One Slot)



Note: When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA MCDAy to HSMCIx_DAy.

© 2017 Microchip Technology Inc.

Figure 34-5: SD Memory Card Bus Topology



The SD Memory Card bus includes the signals listed in Table 34-5.

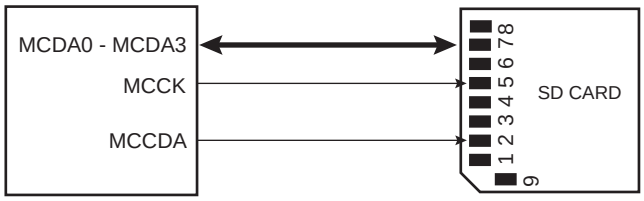
Table 34-5: SD Memory Card Bus Signals

Pin Number	Name	Type[1]	Description	HSMCI Pin Name[2] (Slot z)
1	CD/DAT[3]	I/O/PP	Card detect/ Data line Bit 3	MCDz3
2	CMD	PP	Command/response	MCCDz
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	MCCK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data line Bit 0	MCDz0
8	DAT[1]	I/O/PP	Data line Bit 1 or Interrupt	MCDz1
9	DAT[2]	I/O/PP	Data line Bit 2	MCDz2

Note 1: I: input, O: output, PP: Push Pull, OD: Open Drain.

2: When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

Figure 34-6: SD Card Bus Connections with One Slot



Note: When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA MCDAy to HSMCIx_DAy.

When the HSMCI is configured to operate with SD memory cards, the width of the data bus can be selected in the HSMCI_SDCR. Clearing the SDCBUS bit in this register means that the width is one bit; setting it means that the width is four bits. In the case of High Speed MultiMedia cards, only the data line 0 is used. The other data lines can be used as independent PIOs.

34.8 High Speed MultiMedia Card Operations

After a power-on reset, the cards are initialized by a special message-based High Speed MultiMedia Card bus protocol. Each message is represented by one of the following tokens:

- Command—A command is a token that starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line.
- Response—A response is a token which is sent from an addressed card or (synchronously) from all connected cards to the host as an answer to a previously received command. A response is transferred serially on the CMD line.

- Data—Data can be transferred from the card to the host or vice versa. Data is transferred via the data line.

Card addressing is implemented using a session address assigned during the initialization phase by the bus controller to all currently connected cards. Their unique CID number identifies individual cards.

The structure of commands, responses and data blocks is described in the High Speed MultiMedia Card System Specification. See also Table 34-6.

High Speed MultiMedia Card bus data transfers are composed of these tokens.

There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case, no data token is present in an operation. The bits on the DAT and the CMD lines are transferred synchronous to the clock HSMCI clock.

Two types of data transfer commands are defined:

- Sequential commands—These commands initiate a continuous data stream. They are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum.
- Block-oriented commands—These commands send a data block succeeded by CRC bits.

Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read or when a multiple block transmission has a predefined block count (see Section 34.8.2 “Data Transfer Operation”).

The HSMCI provides a set of registers to perform the entire range of High Speed MultiMedia Card operations.

34.8.1 Command - Response Operation

After reset, the HSMCI is disabled and becomes valid after setting the MCIEN bit in the HSMCI_CR.

The PWSEN bit saves power by dividing the HSMCI clock by $2^{PWSDIV} + 1$ when the bus is inactive.

The two bits, RDPROOF and WRPROOF in the HSMCI Mode Register (HSMCI_MR) allow stopping the HSMCI clock during read or write access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.

All the timings for High Speed MultiMedia Card are defined in the High Speed MultiMedia Card System Specification.

The two bus modes (open drain and push/pull) needed to process all the operations are defined in the HSMCI Command Register (HSMCI_CMDR). The HSMCI_CMDR allows a command to be carried out.

For example, to perform an ALL_SEND_CID command:

	Host Command					N_{ID} Cycles			Response			High Impedance State		
CMD	S	T	Content	CRC	E	Z	******	Z	S	T	CID Content	Z	Z	Z

The command ALL_SEND_CID and the fields and values for the HSMCI_CMDR are described in Table 34-6 and Table 34-7.

Table 34-6: ALL_SEND_CID Command Description

CMD Index	Type	Argument	Response	Abbreviation	Command Description
CMD2	bcr(1)	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line

Note 1: bcr means broadcast command with response.

© 2017 Microchip Technology Inc.

Table 34-7: Fields and Values for HSMCI_CMDR

Field	Value
CMDNB (command number)	2 (CMD2)
RSPTYP (response type)	2 (R2: 136 bits response)
SPCMD (special command)	0 (not a special command)
OPCMD (open drain command)	1
MAXLAT (max latency for command to response)	0 (NID cycles ==> 5 cycles)
TRCMD (transfer command)	0 (No transfer)
TRDIR (transfer direction)	X (available only in transfer command)
TRTYP (transfer type)	X (available only in transfer command)
IOSPCMD (SDIO special command)	0 (not a special command)

The HSMCI_ARGR contains the argument field of the command.

To send a command, the user must perform the following steps:

- Fill the argument register (HSMCI_ARGR) with the command argument.
- Set the command register (HSMCI_CMDR) (see Table 34-7).

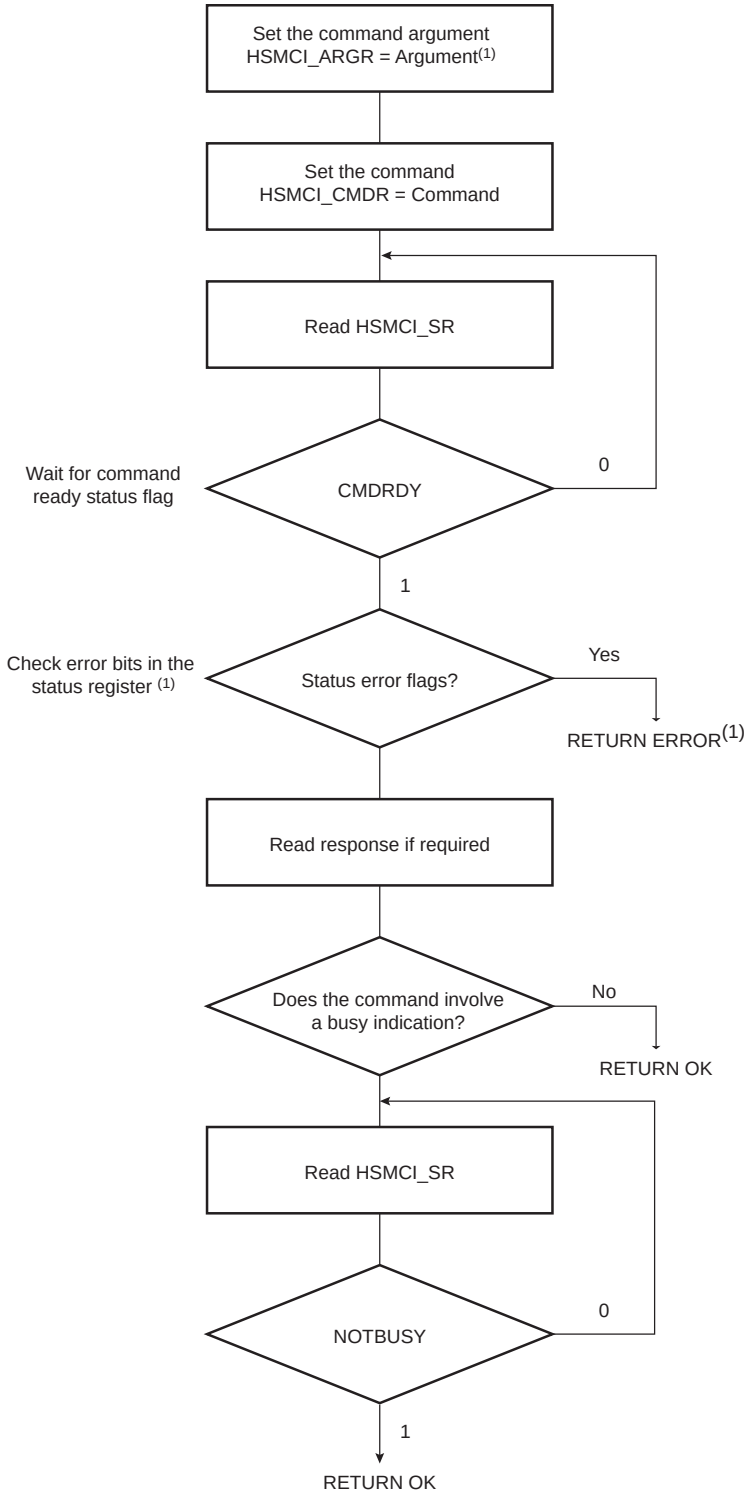
The command is sent immediately after writing the command register.

While the card maintains a busy indication (at the end of a STOP_TRANSMISSION command CMD12, for example), a new command shall not be sent. The NOTBUSY flag in the Status Register (HSMCI_SR) is asserted when the card releases the busy indication.

If the command requires a response, it can be read in the HSMCI Response Register (HSMCI_RSPR). The response size can be from 48 bits up to 136 bits depending on the command. The HSMCI embeds an error detection to prevent any corrupted data during the transfer.

The following flowchart shows how to send a command to the card and read the response if needed. In this example, the status register bits are polled but setting the appropriate bits in the HSMCI Interrupt Enable Register (HSMCI_IER) allows using an interrupt method.

Figure 34-7: Command/Response Functional Flow Diagram



Note: If the command is SEND_OP_COND, the CRC error flag is always present (refer to R3 response in the High Speed MultiMedia Card specification).

© 2017 Microchip Technology Inc.

34.8.2 Data Transfer Operation

The High Speed MultiMedia Card allows several read/write operations (single block, multiple blocks, stream, etc.). These kinds of transfer can be selected setting the Transfer Type (TRTYP) field in the HSMCI Command Register (HSMCI_CMDR).

These operations can be done using the features of the DMA Controller.

In all cases, the block length (BLKLEN field) must be defined either in the HSMCI Mode Register (HSMCI_MR) or in the HSMCI Block Register (HSMCI_BLKR). This field determines the size of the data block.

Consequent to MMC Specification 3.1, two types of multiple block read (or write) transactions are defined (the host can use either one at any time):

- Open-ended/Infinite Multiple block read (or write):

  The number of blocks for the read (or write) multiple block operation is not defined. The card will continuously transfer (or program) data blocks until a stop transmission command is received.

- Multiple block read (or write) with predefined block count (since version 3.1 and higher):

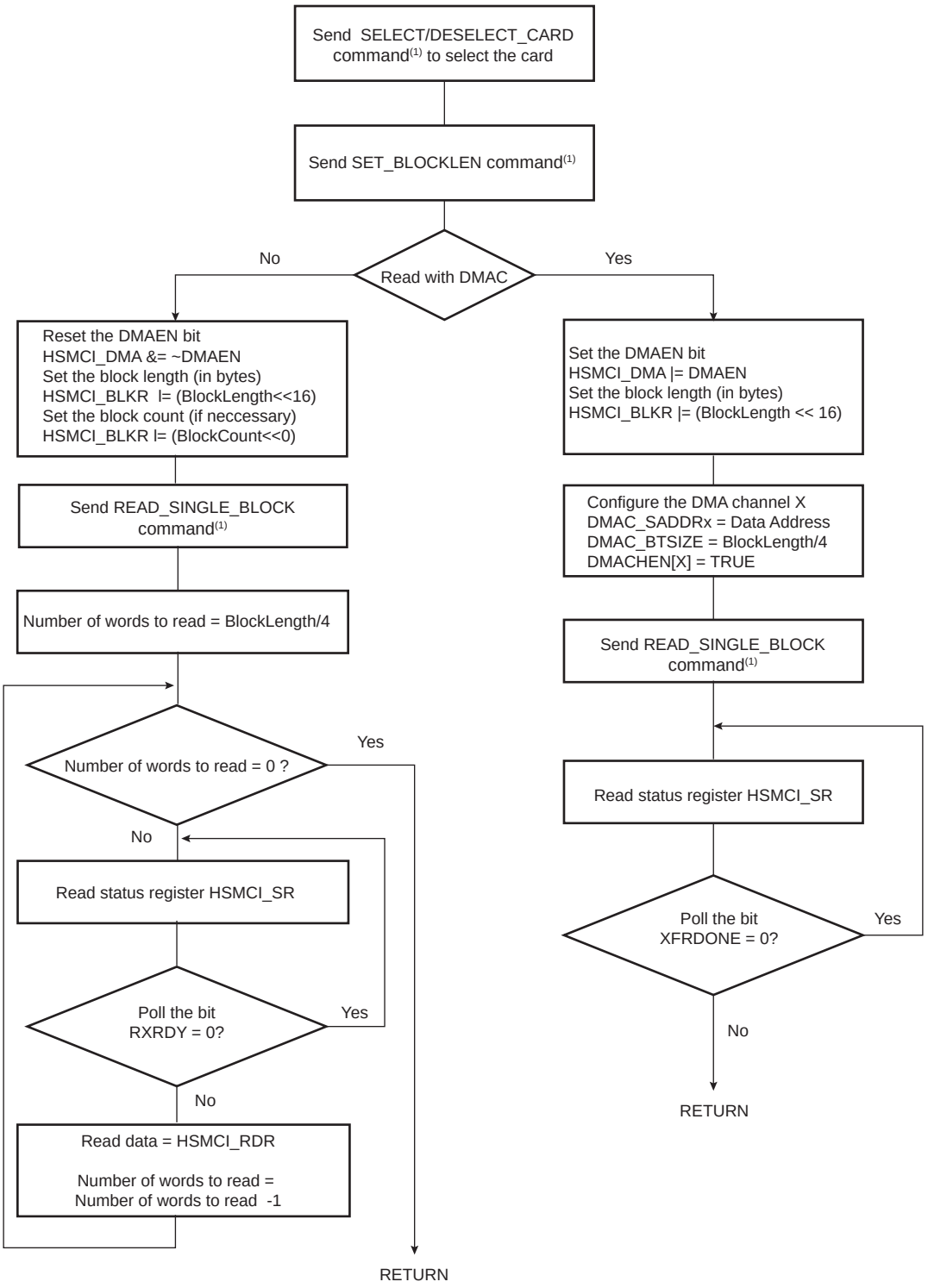
  The card will transfer (or program) the requested number of data blocks and terminate the transaction. The stop command is not required at the end of this type of multiple block read (or write), unless terminated with an error. In order to start a multiple block read (or write) with predefined block count, the host must correctly program the HSMCI Block Register (HSMCI_BLKR). Otherwise the card will start an open-ended multiple block read. The BCNT field of the HSMCI_BLKR defines the number of blocks to transfer (from 1 to 65535 blocks). Programming the value 0 in the BCNT field corresponds to an infinite block transfer.

34.8.3 Read Operation

The following flowchart (Figure 34-8) shows how to read a single block with or without use of DMAC facilities. In this example, a polling method is used to wait for the end of read. Similarly, the user can configure the HSMCI Interrupt Enable Register (HSMCI_IER) to trigger an interrupt at the end of read.

© 2017 Microchip Technology Inc.

Figure 34-8: Read Functional Flow Diagram



Note 1: It is assumed that this command has been correctly sent (see Figure 34-7).

© 2017 Microchip Technology Inc.

34.8.4 Write Operation

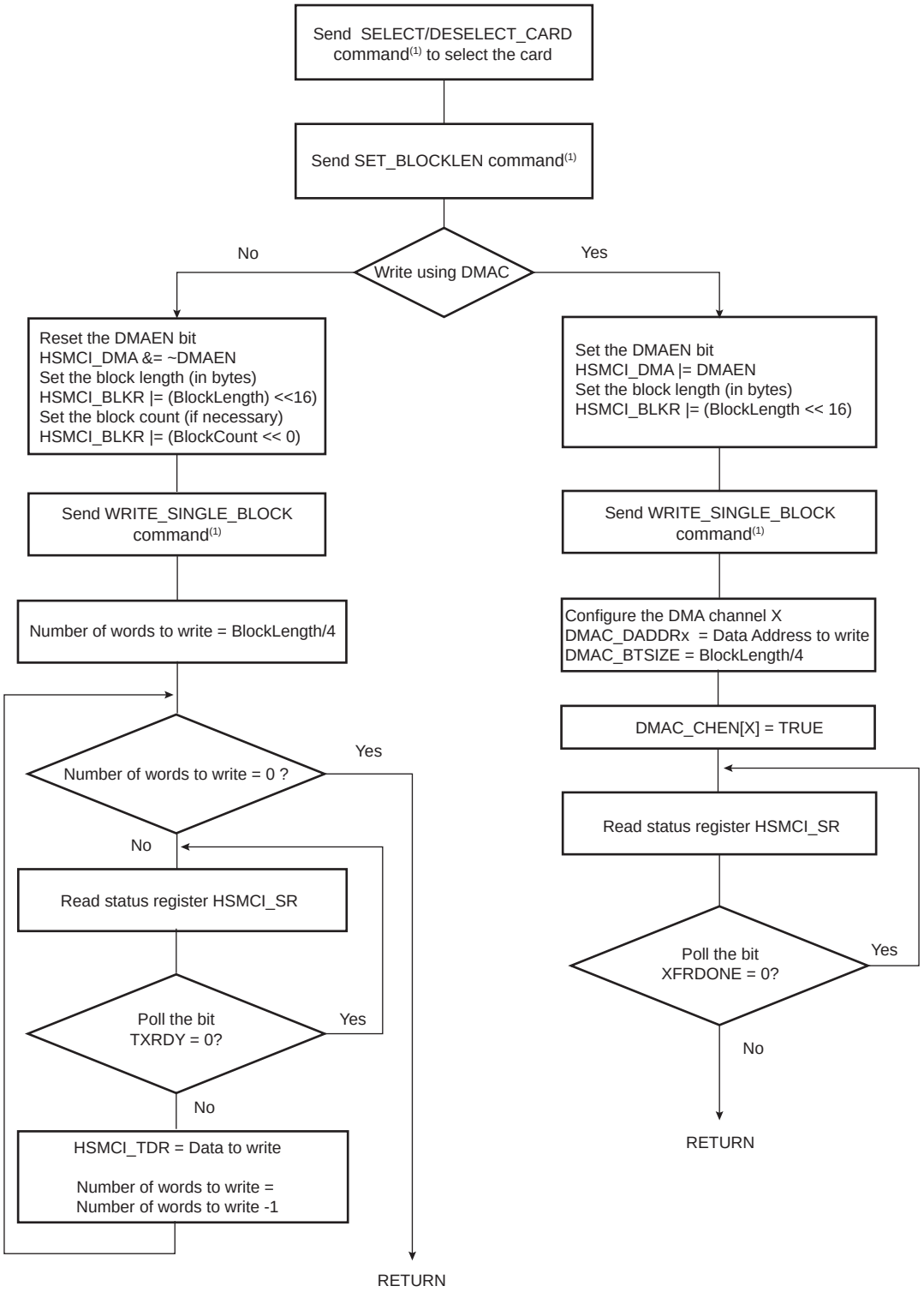
In write operation, the HSMCI Mode Register (HSMCI_MR) is used to define the padding value when writing non-multiple block size. If the bit PADV is 0, then 0x00 value is used when padding data, otherwise 0xFF is used.

If set, the bit DMAEN in the HSMCI DMA Condiguration Register (HSMCI_DMA) enables DMA transfer.

The flowchart in Figure 34-9 shows how to write a single block with or without use of DMA facilities. Polling or interrupt method can be used to wait for the end of write according to the contents of the HSMCI Interrupt Mask Register (HSMCI_IMR).

© 2017 Microchip Technology Inc.

Figure 34-9: Write Functional Flow Diagram

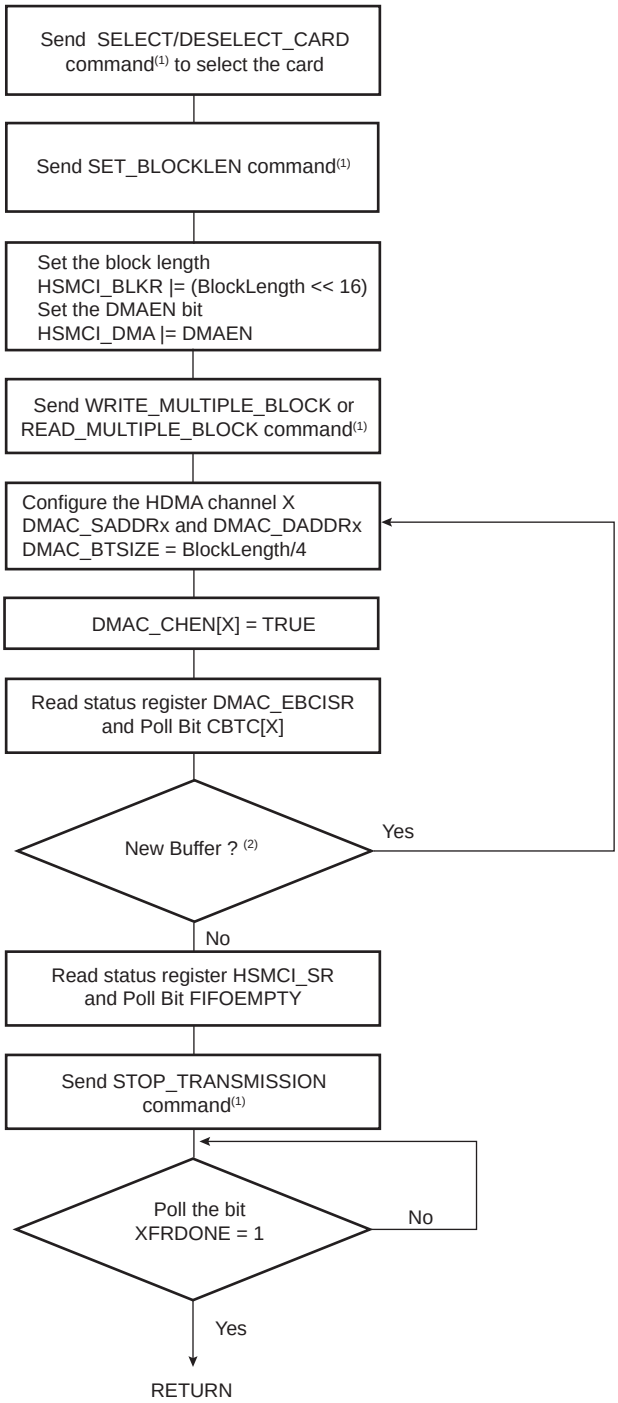


Note: It is assumed that this command has been correctly sent (see Figure 34-7).

© 2017 Microchip Technology Inc.

The flowchart in Figure 34-10 shows how to manage read multiple block and write multiple block transfers with the DMA Controller. Polling or interrupt method can be used to wait for the end of write according to the contents of the HSMCI_IMR.

Figure 34-10: Read Multiple Block and Write Multiple Block



Note 1: It is assumed that this command has been correctly sent (see Figure 34-7).

2: Handle errors reported in HSMCI_SR.

34.8.5 WRITE_SINGLE_BLOCK Operation using DMA Controller

1. Wait until the current command execution has successfully terminated.
   c) Check that CMDRDY and NOTBUSY fields are asserted in HSMCI_SR
2. Program the block length in the card. This value defines the value *block_length.*
3. Program the block length in the HSMCI Configuration Register with *block_length* value.
4. Configure the fields of the HSMCI_DMA register as follows:
   - OFFSET field with *dma_offset.*
   - CHKSIZE is user defined and set according to DMAC_CTRLAx.DCSIZE.
   - DMAEN is set to true to enable DMA hardware handshaking in the HSMCI. This bit was previously set to false.
5. Issue a WRITE_SINGLE_BLOCK command writing HSMCI_ARG then HSMCI_CMDR.
6. Program the DMA Controller.
   a) Read the channel register to choose an available (disabled) channel.
   b) Clear any pending interrupts on the channel from the previous DMAC transfer by reading the DMAC_EBCISR.
   c) Program the channel registers.
   d) The DMAC_SADDRx for Channel x must be set to the location of the source data. When the first data location is not word aligned, the two LSB bits define the temporary value called *dma_offset.* The two LSB bits of DMAC_SADDRx must be configured to 0.
   e) The DMAC_DADDRx for Channel x must be set with the starting address of the HSMCI_FIFO address.
   f) Configure the fields of DMAC_CTRLAx for Channel x as follows:
      –DST_WIDTH is set to WORD.
      –SRC_WIDTH is set to WORD.
      –DCSIZE must be set according to the value of HSMCI_DMA.CHKSIZE field.
      –BTSIZE is programmed with *CEILING((block_length + dma_offset) / 4),* where the ceiling function is the function that returns the smallest integer not less than x.
   g) Configure the fields of DMAC_CTRLBx for Channel x as follows:
      –DST_INCR is set to INCR, the *block_length* value must not be larger than the HSMCI_FIFO aperture.
      –SRC_INCR is set to INCR.
      –FC field is programmed with memory to peripheral flow control mode.
      –Both DST_DSCR and SRC_DSCR are set (descriptor fetch is disabled).
      –DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA controller is able to prefetch data and write HSMCI simultaneously.
   h) Configure the fields of DMAC_CFGx for Channel x as follows:
      –FIFOCFG defines the watermark of the DMAC channel FIFO.
      –DST_H2SEL is set to true to enable hardware handshaking on the destination.
      –DST_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.
   i) Enable Channel x, writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.
7. Wait for XFRDONE in the HSMCI_SR.

34.8.6 READ_SINGLE_BLOCK Operation using DMA Controller

34.8.6.1 Block Length is Multiple of 4

1. Wait until the current command execution has successfully completed.
   a) Check that CMDRDY and NOTBUSY are asserted in HSMCI_SR.
2. Program the block length in the card. This value defines the value *block_length.*
3. Program the block length in the HSMCI Configuration Register with *block_length* value.
4. Set RDPROOF bit in HSMCI_MR to avoid overflow.
5. Configure the fields of the HSMCI_DMA register as follows:
   a) ROPT bit is configured to 0.
   b) OFFSET field is configured to 0.

© 2017 Microchip Technology Inc.

c) CHKSIZE is user defined.
d) DMAEN is set to true to enable DMAC hardware handshaking in the HSMCI. This bit was previously set to false.

6. Issue a READ_SINGLE_BLOCK command.
7. Program the DMA controller.
   a) Read the channel register to choose an available (disabled) channel.
   b) Clear any pending interrupts on the channel from the previous DMA transfer by reading the DMAC_EBCISR.
   c) Program the channel registers.
   d) The DMAC_SADDRx for Channel x must be set with the starting address of the HSMCI_FIFO address.
   e) The DMAC_DADDRx for Channel x must be word aligned.
   f) Configure the fields of the DMAC_CTRLAx register for Channel x as follows:
      –DST_WIDTH is set to WORD.
      –SRC_WIDTH is set to WORD.
      –SCSIZE must be set according to the value of HSMCI_DMA.CHKSIZE.
      –BTSIZE is programmed with *block_length/4.*
   g) Configure the fields of the DMAC_CFGx register for Channel x as follows:
      –DST_INCR is set to INCR.
      –SRC_INCR is set to INCR.
      –FC field is programmed with peripheral to memory flow control mode.
      –Both DST_DSCR and SRC_DSCR are set (descriptor fetch is disabled).
      –DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA controller is able to prefetch data and write HSMCI simultaneously.
   h) Configure the fields of the DMAC_CFGx register for Channel x as follows:
      –FIFOCFG defines the watermark of the DMA channel FIFO.
      –SRC_H2SEL is set to true to enable hardware handshaking on the destination.
      –SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.
      –Enable Channel x, writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.
8. Wait for XFRDONE in the HSMCI_SR.

34.8.6.2 Block Length is Not Multiple of 4 and Padding Not Used (HSMCI_DMA.ROPT = 0)

In the previous DMA transfer flow (block length multiple of 4), the DMA controller is configured to use only WORD AHB access. When the block length is no longer a multiple of 4 this is no longer true. The DMA controller is programmed to copy exactly the block length number of bytes using two transfer descriptors.

1. Use the previous step until READ_SINGLE_BLOCK then
2. Program the DMA controller to use a two descriptors linked list.
   a) Read the channel register to choose an available (disabled) channel.
   b) Clear any pending interrupts on the channel from the previous DMA transfer by reading the DMAC_EBCISR.
   c) Program the channel registers in the Memory for the first descriptor. This descriptor will be word oriented. This descriptor is referred to as LLI_W, standing for LLI word oriented transfer.
   d) The LLI_W.DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.
   e) The LLI_W.DMAC_DADDRx field in the memory must be word aligned.
   f) Configure the fields of LLI_W.DMAC_CTRLAx as follows:
      –DST_WIDTH is set to WORD.
      –SRC_WIDTH is set to WORD.
      –SCSIZE must be set according to the value of HSMCI_DMA.CHKSIZE.
      –BTSIZE is programmed with *block_length/4.* If BTSIZE is zero, this descriptor is skipped later.
   g) Configure the fields of LLI_W.DMAC_CTRLBx as follows:
      –DST_INCR is set to INCR

–SRC_INCR is set to INCR

–FC field is programmed with peripheral to memory flow control mode.

–SRC_DSCR is set to zero. (descriptor fetch is enabled for the SRC)

–DST_DSCR is set to one. (descriptor fetch is disabled for the DST)

–DIF and SIF are set with their respective layer ID. If SIF is different from DIF, DMA controller is able to prefetch data and write HSMCI simultaneously.

h) Configure the fields of LLI_W.DMAC_CFGx for Channel x as follows:

–FIFOCFG defines the watermark of the DMA channel FIFO.

–DST_REP is set to zero meaning that address are contiguous.

–SRC_H2SEL is set to true to enable hardware handshaking on the destination.

–SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

i) Program LLI_W.DMAC_DSCRx with the address of LLI_B descriptor. And set DSCRx_IF to the AHB Layer ID. This operation actually links the Word oriented descriptor on the second byte oriented descriptor. When *block_length[1:0]* is equal to 0 (multiple of 4) LLI_W.DMAC_DSCRx points to 0, only LLI_W is relevant.

j) Program the channel registers in the Memory for the second descriptor. This descriptor will be byte oriented. This descriptor is referred to as LLI_B, standing for LLI Byte oriented.

k) The LLI_B.DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.

l) The LLI_B.DMAC_DADDRx is not relevant if previous word aligned descriptor was enabled. If 1, 2 or 3 bytes are transferred that address is user defined and not word aligned.

m) Configure the fields of LLI_B.DMAC_CTRLAx as follows:

–DST_WIDTH is set to BYTE.

–SRC_WIDTH is set to BYTE.

–SCSIZE must be set according to the value of HSMCI_DMA.CHKSIZE.

–BTSIZE is programmed with *block_length[1:0].* (last 1, 2, or 3 bytes of the buffer).

n) Configure the fields of LLI_B.DMAC_CTRLBx as follows:

–DST_INCR is set to INCR

–SRC_INCR is set to INCR

–FC field is programmed with peripheral to memory flow control mode.

–Both SRC_DSCR and DST_DSCR are set (descriptor fetch is disabled) or Next descriptor location points to 0.

–DIF and SIF are set with their respective layer ID. If SIF is different from DIF, DMA Controller is able to prefetch data and write HSMCI simultaneously.

o) Configure the LLI_B.DMAC_CFGx memory location for Channel x as follows:

–FIFOCFG defines the watermark of the DMA channel FIFO.

–SRC_H2SEL is set to true to enable hardware handshaking on the destination.

–SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

p) Program LLI_B.DMAC_DSCR with 0.

q) Program the DMAC_CTRLBx register for Channel x with 0. Its content is updated with the LLI fetch operation.

r) Program DMAC_DSCRx with the address of LLI_W if *block_length* greater than 4 else with address of LLI_B.

s) Enable Channel x writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.

3. Wait for XFRDONE in the HSMCI_SR.

34.8.6.3 Block Length is Not Multiple of 4, with Padding Value (HSMCI_DMA.ROPT = 1)

When the ROPT bit is set, the DMA Controller performs only WORD access on the bus to transfer a non-multiple of 4 block length, unlike the previous flow, in which the transfer size is rounded to the nearest multiple of 4.

1. Program the HSMCI, see previous flow.

© 2017 Microchip Technology Inc.

- ROPT bit is set.

2. Program the DMA Controller
   a) Read the channel register to choose an available (disabled) channel.
   b) Clear any pending interrupts on the channel from the previous DMA transfer by reading the DMAC_EBCISR.
   c) Program the channel registers.
   d) The DMAC_SADDRx for Channel x must be set with the starting address of the HSMCI_FIFO address.
   e) The DMAC_DADDRx for Channel x must be word aligned.
   f) Configure the fields of DMAC_CTRLAx for Channel x as follows:
      –DST_WIDTH is set to WORD
      –SRC_WIDTH is set to WORD
      –SCSIZE must be set according to the value of HSMCI_DMA.CHKSIZE.
      –BTSIZE is programmed with *CEILING(block_length/4).*
   g) Configure the fields of DMAC_CTRLBx for Channel x as follows:
      –DST_INCR is set to INCR
      –SRC_INCR is set to INCR
      –FC field is programmed with peripheral to memory flow control mode.
      –Both DST_DSCR and SRC_DSCR are set (descriptor fetch is disabled).
      –DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA Controller is able to prefetch data and write HSMCI simultaneously.
   h) Configure the fields of DMAC_CFGx for Channel x as follows:
      –FIFOCFG defines the watermark of the DMA channel FIFO.
      –SRC_H2SEL is set to true to enable hardware handshaking on the destination.
      –SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.
      –Enable Channel x writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.
3. Wait for XFRDONE in the HSMCI_SR.

34.8.7 WRITE_MULTIPLE_BLOCK

34.8.7.1 One Block per Descriptor

1. Wait until the current command execution has successfully terminated.
   a) Check that CMDRDY and NOTBUSY are asserted in HSMCI_SR.
2. Program the block length in the card. This value defines the value *block_length.*
3. Program the block length in the HSMCI Configuration Register with *block_length* value.
4. Configure the fields of the HSMCI_DMA register as follows:

- OFFSET field with *dma_offset.*
- CHKSIZE is user defined.
- DMAEN is set to true to enable DMAC hardware handshaking in the HSMCI. This bit was previously set to false.

5. Issue a WRITE_MULTIPLE_BLOCK command.
6. Program the DMA Controller to use a list of descriptors. Each descriptor transfers one block of data. Block *n* of data is transferred with descriptor LLI(n).
   a) Read the channel register to choose an available (disabled) channel.
   b) Clear any pending interrupts on the channel from the previous DMAC transfer by reading the DMAC_EBCISR.
   c) Program a List of descriptors.
   d) The LLI(n).DMAC_SADDRx memory location for Channel x must be set to the location of the source data. When the first data location is not word aligned, the two LSB bits define the temporary value called *dma_offset.* The two LSB bits of LLI(n).DMAC_SADDRx must be configured to 0.
   e) The LLI(n).DMAC_DADDRx for Channel x must be set with the starting address of the HSMCI_FIFO address.
   f) Configure the fields of LLI(n).DMAC_CTRLAx for Channel x as follows:
      –DST_WIDTH is set to WORD.

–SRC_WIDTH is set to WORD.

–DCSIZE must be set according to the value of HSMCI_DMA.CHKSIZE.

–BTSIZE is programmed with *CEILING((block_length + dma_offset)/4).*

g) Configure the fields of LLI(n).DMAC_CTRLBx for Channel x as follows:

–DST_INCR is set to INCR.

–SRC_INCR is set to INCR.

–DST_DSCR is configured to 0 (fetch operation is enabled for the destination).

–SRC_DSCR is set (source address is contiguous).

–FC field is programmed with memory to peripheral flow control mode.

–Both DST_DSCR and SRC_DSCR are set (descriptor fetch is disabled).

–DIF and SIF are set with their respective layer ID. If SIF is different from DIF, DMA Controller is able to prefetch data and write HSMCI simultaneously.

h) Configure the fields of LLI(n).DMAC_CFGx for Channel x as follows:

–FIFOCFG defines the watermark of the DMA channel FIFO.

–DST_H2SEL is set to true to enable hardware handshaking on the destination.

–SRC_REP is configured to 0. (contiguous memory access at block boundary)

–DST_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

i) If LLI(n) is the last descriptor, then LLI(n).DSCR points to 0 else LLI(n) points to the start address of LLI(n+1).

j) Program DMAC_CTRLBx for the Channel Register x with 0. Its content is updated with the LLI fetch operation.

k) Program DMAC_DSCRx for the Channel Register x with the address of the first descriptor LLI(0).

l) Enable Channel x writing one to DMAC_CHER[x]. The DMA is ready and waiting for request.

7. Poll CBTC[x] bit in the DMAC_EBCISR.
8. If a new list of buffers shall be transferred, repeat step 6. Check and handle HSMCI errors.
9. Poll FIFOEMPTY field in the HSMCI_SR.
10. Send The STOP_TRANSMISSION command writing HSMCI_ARG then HSMCI_CMDR.
11. Wait for XFRDONE in the HSMCI_SR.

34.8.8 READ_MULTIPLE_BLOCK

34.8.8.1 Block Length is a Multiple of 4

1. Wait until the current command execution has successfully terminated.
   a) Check that CMDRDY and NOTBUSY are asserted in HSMCI_SR.
2. Program the block length in the card. This value defines the value *block_length.*
3. Program the block length in the HSMCI Configuration Register with *block_length* value.
4. Set RDPROOF bit in HSMCI_MR to avoid overflow.
5. Configure the fields of the HSMCI_DMA register as follows:
   - ROPT bit is configured to 0.
   - OFFSET field is configured to 0.
   - CHKSIZE is user defined.
   - DMAEN is set to true to enable DMAC hardware handshaking in the HSMCI. This bit was previously set to false.
6. Issue a READ_MULTIPLE_BLOCK command.
7. Program the DMA Controller to use a list of descriptors:
   a) Read the channel register to choose an available (disabled) channel.
   b) Clear any pending interrupts on the channel from the previous DMA transfer by reading the DMAC_EBCISR.
   c) Program the channel registers in the Memory with the first descriptor. This descriptor will be word oriented. This descriptor is referred to as LLI_W(n), standing for LLI word oriented transfer for block *n*.
   d) The LLI_W(n).DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.
   e) The LLI_W(n).DMAC_DADDRx field in the memory must be word aligned.

© 2017 Microchip Technology Inc.

f) Configure the fields of LLI_W(n).DMAC_CTRLAx as follows:
   – DST_WIDTH is set to WORD
   – SRC_WIDTH is set to WORD
   – SCSIZE must be set according to the value of HSMCI_DMA.CHKSIZE.
   – BTSIZE is programmed with *block_length/4.*

g) Configure the fields of LLI_W(n).DMAC_CTRLBx as follows:
   – DST_INCR is set to INCR.
   – SRC_INCR is set to INCR.
   – FC field is programmed with peripheral to memory flow control mode.
   – SRC_DSCR is configured to 0 (descriptor fetch is enabled for the SRC).
   – DST_DSCR is set to TRUE (descriptor fetch is disabled for the DST).
   – DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA Controller is able to prefetch data and write HSMCI simultaneously.

h) Configure the fields of the LLI_W(n).DMAC_CFGx register for Channel x as follows:
   – FIFOCFG defines the watermark of the DMA channel FIFO.
   – DST_REP is set to zero. Addresses are contiguous.
   – SRC_H2SEL is set to true to enable hardware handshaking on the destination.
   – SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

i) Program LLI_W(n).DMAC_DSCRx with the address of LLI_W(n+1) descriptor. And set the DSCRx_IF to the AHB Layer ID. This operation actually links descriptors together. If LLI_W(n) is the last descriptor then LLI_W(n).DMAC_DSCRx points to 0.

j) Program the DMAC_CTRLBx register for Channel x with 0. Its content is updated with the LLI Fetch operation.

k) Program DMAC_DSCRx for Channel x with the address of LLI_W(0).

l) Enable Channel x writing one to DMAC_CHER[x]. The DMA is ready and waiting for request.

8. Poll CBTC[x] bit in the DMAC_EBCISR.
9. If a new list of buffer shall be transferred repeat step 6. Check and handle HSMCI errors.
10. Poll FIFOEMPTY field in the HSMCI_SR.
11. Send The STOP_TRANSMISSION command writing the HSMCI_ARG then the HSMCI_CMDR.
12. Wait for XFRDONE in the HSMCI_SR.

34.8.8.2 Block Length is Not Multiple of 4 (HSMCI_DMA.ROPT = 0)

Two DMA Transfer descriptors are used to perform the HSMCI block transfer.

1. Use the previous step to configure the HSMCI to perform a READ_MULTIPLE_BLOCK command.
2. Issue a READ_MULTIPLE_BLOCK command.
3. Program the DMA Controller to use a list of descriptors.
   a) Read the channel register to choose an available (disabled) channel.
   b) Clear any pending interrupts on the channel from the previous DMAC transfer by reading the DMAC_EBCISR.
   c) For every block of data repeat the following procedure:
   d) Program the channel registers in the Memory for the first descriptor. This descriptor will be word oriented. This descriptor is referred to as LLI_W(n) standing for LLI word oriented transfer for block *n*.
   e) The LLI_W(n).DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.
   f) The LLI_W(n).DMAC_DADDRx field in the memory must be word aligned.
   g) Configure the fields of LLI_W(n).DMAC_CTRLAx as follows:
      – DST_WIDTH is set to WORD.
      – SRC_WIDTH is set to WORD.
      – SCSIZE must be set according to the value of HSMCI_DMA.CHKSIZE.
      – BTSIZE is programmed with *block_length/4.* If BTSIZE is zero, this descriptor is skipped later.

h) Configure the fields of LLI_W(n).DMAC_CTRLBx as follows:
   – DST_INCR is set to INCR.
   – SRC_INCR is set to INCR.
   – FC field is programmed with peripheral to memory flow control mode.
   – SRC_DSCR is configured to 0 (descriptor fetch is enabled for the SRC).
   – DST_DSCR is set to TRUE (descriptor fetch is disabled for the DST).
   – DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA Controller is able to prefetch data and write HSMCI simultaneously.

i) Configure the fields of LLI_W(n).DMAC_CFGx for Channel x as follows:
   – FIFOCFG defines the watermark of the DMA channel FIFO.
   – DST_REP is set to zero. Address are contiguous.
   – SRC_H2SEL is set to true to enable hardware handshaking on the destination.
   – SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

j) Program LLI_W(n).DMAC_DSCRx with the address of LLI_B(n) descriptor. And set the DSCRx_IF to the AHB Layer ID. This operation actually links the Word oriented descriptor on the second byte oriented descriptor. When *block_length[1:0]* is equal to 0 (multiple of 4) LLI_W(n).DMAC_DSCRx points to 0, only LLI_W(n) is relevant.

k) Program the channel registers in the Memory for the second descriptor. This descriptor will be byte oriented. This descriptor is referred to as LLI_B(n), standing for LLI Byte oriented.

l) The LLI_B(n).DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.

m) The LLI_B(n).DMAC_DADDRx is not relevant if previous word aligned descriptor was enabled. If 1, 2 or 3 bytes are transferred, that address is user defined and not word aligned.

n) Configure the fields of LLI_B(n).DMAC_CTRLAx as follows:
   – DST_WIDTH is set to BYTE.
   – SRC_WIDTH is set to BYTE.
   – SCSIZE must be set according to the value of HSMCI_DMA.CHKSIZE.
   – BTSIZE is programmed with *block_length[1:0].* (last 1, 2, or 3 bytes of the buffer).

o) Configure the fields of LLI_B(n).DMAC_CTRLBx as follows:
   – DST_INCR is set to INCR.
   – SRC_INCR is set to INCR.
   – FC field is programmed with peripheral to memory flow control mode.
   – Both SRC_DSCR and DST_DSCR are set (descriptor fetch is disabled) or Next descriptor location points to 0.
   – DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA Controller is able to prefetch data and write HSMCI simultaneously.

p) Configure the LLI_B(n).DMAC_CFGx memory location for Channel x as follows:
   – FIFOCFG defines the watermark of the DMAC channel FIFO.
   – SRC_H2SEL is set to true to enable hardware handshaking on the destination.
   – SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller

q) Program LLI_B(n).DMAC_DSCR with address of descriptor LLI_W(n+1). If LLI_B(n) is the last descriptor, then program LLI_B(n).DMAC_DSCR with 0.

r) Program the DMAC_CTRLBx register for Channel x with 0. Its content is updated with the LLI Fetch operation.

s) Program DMAC_DSCRx with the address of LLI_W(0) if *block_length* is greater than 4 else with address of LLI_B(0).

t) Enable Channel x writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.

4. Enable DMADONE interrupt in the HSMCI_IER.
5. Poll CBTC[x] bit in the DMAC_EBCISR.
6. If a new list of buffers shall be transferred, repeat step 7. Check and handle HSMCI errors.

© 2017 Microchip Technology Inc.

7. Poll FIFOEMPTY field in the HSMCI_SR.
8. Send The STOP_TRANSMISSION command writing HSMCI_ARG then HSMCI_CMDR.
9. Wait for XFRDONE in the HSMCI_SR.

34.8.8.3 Block Length is Not a Multiple of 4 (HSMCI_DMA.ROPT = 1)

One DMA Transfer descriptor is used to perform the HSMCI block transfer, the DMA writes a value rounded up to the nearest multiple of 4.

1. Use the previous step to configure the HSMCI to perform a READ_MULTIPLE_BLOCK.
2. Set the ROPT bit in the HSMCI_DMA register.
3. Issue a READ_MULTIPLE_BLOCK command.
4. Program the DMA controller to use a list of descriptors:
   a) Read the channel register to choose an available (disabled) channel.
   b) Clear any pending interrupts on the channel from the previous DMAC transfer by reading the DMAC_EBCISR.
   c) Program the channel registers in the Memory with the first descriptor. This descriptor will be word oriented. This descriptor is referred to as LLI_W(n), standing for LLI word oriented transfer for block *n*.
   d) The LLI_W(n).DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.
   e) The LLI_W(n).DMAC_DADDRx field in the memory must be word aligned.
   f) Configure the fields of LLI_W(n).DMAC_CTRLAx as follows:
      –DST_WIDTH is set to WORD.
      –SRC_WIDTH is set to WORD.
      –SCSIZE must be set according to the value of HSMCI_DMA.CHKSIZE.
      –BTSIZE is programmed with *CEILING(block_length/4).*
   g) Configure the fields of LLI_W(n).DMAC_CTRLBx as follows:
      –DST_INCR is set to INCR
      –SRC_INCR is set to INCR
      –FC field is programmed with peripheral to memory flow control mode.
      –SRC_DSCR is configured to 0. (descriptor fetch is enabled for the SRC)
      –DST_DSCR is set to TRUE. (descriptor fetch is disabled for the DST)
      –DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA Controller is able to prefetch data and write HSMCI simultaneously.
   h) Configure the fields of LLI_W(n).DMAC_CFGx for Channel x as follows:
      –FIFOCFG defines the watermark of the DMA channel FIFO.
      –DST_REP is set to zero. Address are contiguous.
      –SRC_H2SEL is set to true to enable hardware handshaking on the destination.
      –SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.
   i) Program LLI_W(n).DMAC_DSCRx with the address of LLI_W(n+1) descriptor. And set the DSCRx_IF to the AHB Layer ID. This operation actually links descriptors together. If LLI_W(n) is the last descriptor then LLI_W(n).DMAC_DSCRx points to 0.
   j) Program the DMAC_CTRLBx register for Channel x with 0. Its content is updated with the LLI Fetch operation.
   k) Program the DMAC_DSCRx for Channel x with the address of LLI_W(0).
   l) Enable Channel x writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.
5. Poll CBTC[x] bit in the DMAC_EBCISR.
6. If a new list of buffers shall be transferred repeat step 7. Check and handle HSMCI errors.
7. Poll FIFOEMPTY field in the HSMCI_SR.
8. Send The STOP_TRANSMISSION command writing the HSMCI_ARG then the HSMCI_CMDR.
9. Wait for XFRDONE in the HSMCI_SR.

34.9 SD/SDIO Card Operation

The High Speed MultiMedia Card Interface allows processing of SD Memory (Secure Digital Memory Card) and SDIO (SD Input Output) Card commands.

SD/SDIO cards are based on the MultiMedia Card (MMC) format, but are physically slightly thicker and feature higher data transfer rates, a lock switch on the side to prevent accidental overwriting and security features. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the High Speed MultiMedia Card with some additions. SD slots can actually be used for more than flash memory cards. Devices that support SDIO can use small devices designed for the SD form factor, such as GPS receivers, Wi-Fi or Bluetooth adapters, modems, barcode readers, IrDA adapters, FM radio tuners, RFID readers, digital cameras and more.

SD/SDIO is covered by numerous patents and trademarks, and licensing is only available through the Secure Digital Card Association.

The SD/SDIO Card communication is based on a 9-pin interface (Clock, Command, 4 x Data and 3 x Power lines). The communication protocol is defined as a part of this specification. The main difference between the SD/SDIO Card and the High Speed MultiMedia Card is the initialization process.

The SD/SDIO Card Register (HSMCI_SDCR) allows selection of the Card Slot and the data bus width.

The SD/SDIO Card bus allows dynamic configuration of the number of data lines. After power up, by default, the SD/SDIO Card uses only DAT0 for data transfer. After initialization, the host can change the bus width (number of active data lines).

34.9.1 SDIO Data Transfer Type

SDIO cards may transfer data in either a multi-byte (1 to 512 bytes) or an optional block format (1 to 511 blocks), while the SD memory cards are fixed in the block transfer mode. The TRTYP field in the HSMCI Command Register (HSMCI_CMDR) allows to choose between SDIO Byte or SDIO Block transfer.

The number of bytes/blocks to transfer is set through the BCNT field in the HSMCI Block Register (HSMCI_BLKR). In SDIO Block mode, the field BLKLEN must be set to the data block size while this field is not used in SDIO Byte mode.

An SDIO Card can have multiple I/O or combined I/O and memory (called Combo Card). Within a multi-function SDIO or a Combo card, there are multiple devices (I/O and memory) that share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of suspend/resume (Refer to the SDIO Specification for more details). To send a suspend or a resume command, the host must set the SDIO Special Command field (IOSPCMD) in the HSMCI Command Register.

34.9.2 SDIO Interrupts

Each function within an SDIO or Combo card may implement interrupts (Refer to the SDIO Specification for more details). In order to allow the SDIO card to interrupt the host, an interrupt function is added to a pin on the DAT[1] line to signal the card's interrupt to the host. An SDIO interrupt on each slot can be enabled through the HSMCI Interrupt Enable Register. The SDIO interrupt is sampled regardless of the currently selected slot.

34.10 CE-ATA Operation

CE-ATA maps the streamlined ATA command set onto the MMC interface. The ATA task file is mapped onto MMC register space.

CE-ATA utilizes five MMC commands:

- GO_IDLE_STATE (CMD0): used for hard reset.
- STOP_TRANSMISSION (CMD12): causes the ATA command currently executing to be aborted.
- FAST_IO (CMD39): Used for single register access to the ATA taskfile registers, 8-bit access only.
- RW_MULTIPLE_REGISTERS (CMD60): used to issue an ATA command or to access the control/status registers.
- RW_MULTIPLE_BLOCK (CMD61): used to transfer data for an ATA command.

CE-ATA utilizes the same MMC command sequences for initialization as traditional MMC devices.

34.10.1 Executing an ATA Polling Command

1. Issue READ_DMA_EXT with RW_MULTIPLE_REGISTER (CMD60) for 8 KB of DATA.
2. Read the ATA status register until DRQ is set.
3. Issue RW_MULTIPLE_BLOCK (CMD61) to transfer DATA.
4. Read the ATA status register until DRQ && BSY are configured to 0.

© 2017 Microchip Technology Inc.

34.10.2 Executing an ATA Interrupt Command

1. Issue READ_DMA_EXT with RW_MULTIPLE_REGISTER (CMD60) for 8 KB of DATA with nIEN field set to zero to enable the command completion signal in the device.
2. Issue RW_MULTIPLE_BLOCK (CMD61) to transfer DATA.
3. Wait for Completion Signal Received Interrupt.

34.10.3 Aborting an ATA Command

If the host needs to abort an ATA command prior to the completion signal it must send a special command to avoid potential collision on the command line. The SPCMD field of the HSMCI_CMDR must be set to 3 to issue the CE-ATA completion Signal Disable Command.

34.10.4 CE-ATA Error Recovery

Several methods of ATA command failure may occur, including:

- No response to an MMC command, such as RW_MULTIPLE_REGISTER (CMD60).
- CRC is invalid for an MMC command or response.
- CRC16 is invalid for an MMC data packet.
- ATA Status register reflects an error by setting the ERR bit to one.
- The command completion signal does not arrive within a host specified time out period.

Error conditions are expected to happen infrequently. Thus, a robust error recovery mechanism may be used for each error event. The recommended error recovery procedure after a timeout is:

- Issue the command completion signal disable if nIEN was cleared to zero and the RW_MULTIPLE_BLOCK (CMD61) response has been received.
- Issue STOP_TRANSMISSION (CMD12) and successfully receive the R1 response.
- Issue a software reset to the CE-ATA device using FAST_IO (CMD39).

If STOP_TRANMISSION (CMD12) is successful, then the device is again ready for ATA commands. However, if the error recovery procedure does not work as expected or there is another timeout, the next step is to issue GO_IDLE_STATE (CMD0) to the device. GO_IDLE_STATE (CMD0) is a hard reset to the device and completely resets all device states.

Note that after issuing GO_IDLE_STATE (CMD0), all device initialization needs to be completed again. If the CE-ATA device completes all MMC commands correctly but fails the ATA command with the ERR bit set in the ATA Status register, no error recovery action is required. The ATA command itself failed implying that the device could not complete the action requested, however, there was no communication or protocol failure. After the device signals an error by setting the ERR bit to one in the ATA Status register, the host may attempt to retry the command.

34.11 HSMCI Boot Operation Mode

In boot operation mode, the processor can read boot data from the slave (MMC device) by keeping the CMD line low after power-on before issuing CMD1. The data can be read from either the boot area or user area, depending on register setting.

34.11.1 Boot Procedure, Processor Mode

1. Configure the HSMCI data bus width programming SDCBUS Field in the HSMCI_SDCR. The BOOT_BUS_WIDTH field located in the device Extended CSD register must be set accordingly.
2. Set the byte count to 512 bytes and the block count to the desired number of blocks, writing BLKLEN and BCNT fields of the HSMCI_BLKR.
3. Issue the Boot Operation Request command by writing to the HSMCI_CMDR with SPCMD field set to BOOTREQ, TRDIR set to READ and TRCMD set to “start data transfer”.
4. The BOOT_ACK field located in the HSMCI_CMDR must be set to one, if the BOOT_ACK field of the MMC device located in the Extended CSD register is set to one.
5. Host processor can copy boot data sequentially as soon as the RXRDY flag is asserted.
6. When Data transfer is completed, host processor shall terminate the boot stream by writing the HSMCI_CMDR with SPCMD field set to BOOTEND.

34.11.2 Boot Procedure DMA Mode

1. Configure the HSMCI data bus width by programming SDCBUS Field in the HSMCI_SDCR. The BOOT_BUS_WIDTH field in the device Extended CSD register must be set accordingly.
2. Set the byte count to 512 bytes and the block count to the desired number of blocks by writing BLKLEN and BCNT fields of the HSMCI_BLKR.
3. Enable DMA transfer in the HSMCI_DMA register.
4. Configure DMA controller, program the total amount of data to be transferred and enable the relevant channel.
5. Issue the Boot Operation Request command by writing to the HSMCI_CMDR with SPCND set to BOOTREQ, TRDIR set to READ and TRCMD set to “start data transfer”.
6. DMA controller copies the boot partition to the memory.
7. When DMA transfer is completed, host processor shall terminate the boot stream by writing the HSMCI_CMDR with SPCMD field set to BOOTEND.

34.12 HSMCI Transfer Done Timings

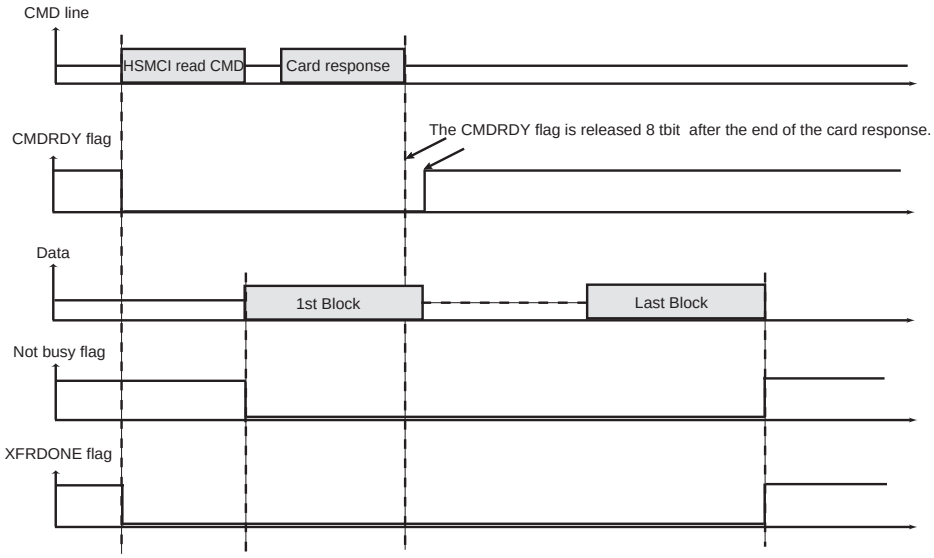
34.12.1 Definition

The XFRDONE flag in the HSMCI_SR indicates exactly when the read or write sequence is finished.

34.12.2 Read Access

During a read access, the XFRDONE flag behaves as shown in Figure 34-11.

Figure 34-11: XFRDONE During a Read Access

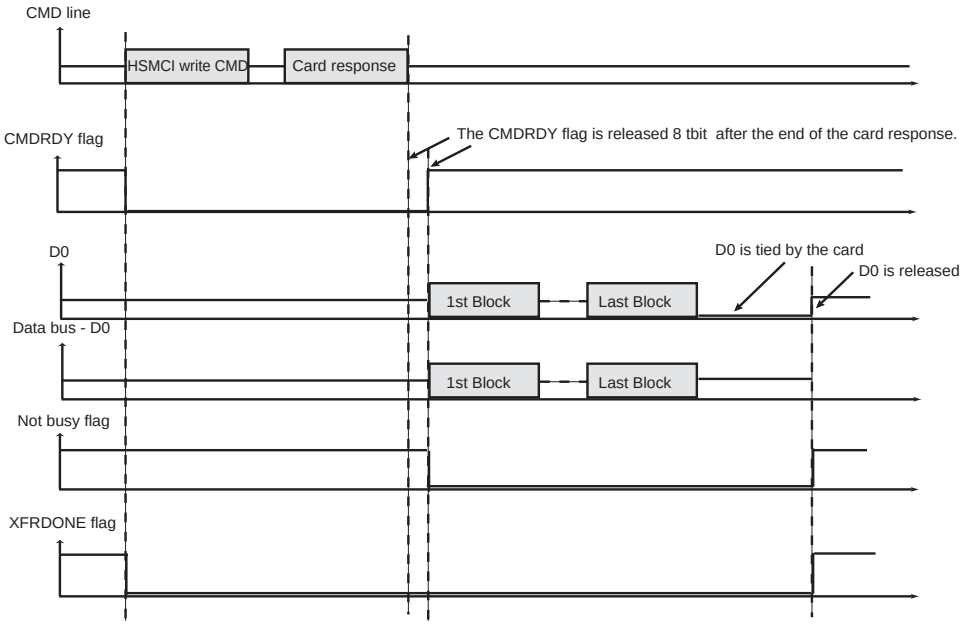


34.12.3 Write Access

During a write access, the XFRDONE flag behaves as shown in Figure 34-12.

© 2017 Microchip Technology Inc.

Figure 34-12: XFRDONE During a Write Access



34.13 Register Write Protection

To prevent any single software error from corrupting HSMCI behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the HSMCI Write Protection Mode Register (HSMCI_WPMR).

If a write access to a write-protected register is detected, the WPVS bit in the HSMCI Write Protection Status Register (HSMCI_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the HSMCI_WPSR.

The following registers can be protected:

- HSMCI Mode Register
- HSMCI Data Timeout Register
- HSMCI SDCard/SDIO Register
- HSMCI Completion Signal Timeout Register
- HSMCI DMA Configuration Register
- HSMCI Configuration Register

© 2017 Microchip Technology Inc.

34.14 High Speed MultiMedia Card Interface (HSMCI) User Interface

Table 34-8: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	HSMCI_CR	Write-only	–
0x04	Mode Register	HSMCI_MR	Read/Write	0x0
0x08	Data Timeout Register	HSMCI_DTOR	Read/Write	0x0
0x0C	SD/SDIO Card Register	HSMCI_SDCR	Read/Write	0x0
0x10	Argument Register	HSMCI_ARGR	Read/Write	0x0
0x14	Command Register	HSMCI_CMDR	Write-only	–
0x18	Block Register	HSMCI_BLKR	Read/Write	0x0
0x1C	Completion Signal Timeout Register	HSMCI_CSTOR	Read/Write	0x0
0x20	Response Register(1)	HSMCI_RSPR	Read-only	0x0
0x24	Response Register(1)	HSMCI_RSPR	Read-only	0x0
0x28	Response Register(1)	HSMCI_RSPR	Read-only	0x0
0x2C	Response Register(1)	HSMCI_RSPR	Read-only	0x0
0x30	Receive Data Register	HSMCI_RDR	Read-only	0x0
0x34	Transmit Data Register	HSMCI_TDR	Write-only	–
0x38–0x3C	Reserved	–	–	–
0x40	Status Register	HSMCI_SR	Read-only	0xC0E5
0x44	Interrupt Enable Register	HSMCI_IER	Write-only	–
0x48	Interrupt Disable Register	HSMCI_IDR	Write-only	–
0x4C	Interrupt Mask Register	HSMCI_IMR	Read-only	0x0
0x50	DMA Configuration Register	HSMCI_DMA	Read/Write	0x00
0x54	Configuration Register	HSMCI_CFG	Read/Write	0x00
0x58–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	HSMCI_WPMR	Read/Write	–
0xE8	Write Protection Status Register	HSMCI_WPSR	Read-only	–
0xEC–0xFC	Reserved	–	–	–
0x100–0x1FC	Reserved	–	–	–
0x200	FIFO Memory Aperture0	HSMCI_FIFO0	Read/Write	0x0
...	...	...	...	...
0x5FC	FIFO Memory Aperture255	HSMCI_FIFO255	Read/Write	0x0

Note 1: The Response Register can be read by N accesses at the same HSMCI_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

34.14.1 HSMCI Control Register

Name: HSMCI_CR

Address:0xF0008000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	–	–	PWSDIS	PWSEN	MCIDIS	MCIEN

MCIEN: Multi-Media Interface Enable

0: No effect.

1: Enables the Multi-Media Interface if MCDIS is 0.

MCIDIS: Multi-Media Interface Disable

0: No effect.

1: Disables the Multi-Media Interface.

PWSEN: Power Save Mode Enable

0: No effect.

1: Enables the Power Saving Mode if PWSDIS is 0.

Warning: Before enabling this mode, the user must set a value different from 0 in the PWSDIV field of the HSMCI_MR.

PWSDIS: Power Save Mode Disable

0: No effect.

1: Disables the Power Saving Mode.

SWRST: Software Reset

0: No effect.

1: Resets the HSMCI. A software triggered hardware reset of the HSMCI is performed.

© 2017 Microchip Technology Inc.

34.14.2 HSMCI Mode Register

Name: HSMCI_MR

Address:0xF0008004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	CLKODD
15	14	13	12	11	10	9	8
–	PADV	FBYTE	WRPROOF	RDPROOF	PWSDIV		
7	6	5	4	3	2	1	0
CLKDIV							

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

CLKDIV: Clock Divider

High Speed MultiMedia Card Interface clock (MCCK or HSMCI_CK) is Master Clock (MCK) divided by ({CLKDIV,CLKODD}+2).

PWSDIV: Power Saving Divider

High Speed MultiMedia Card Interface clock is divided by $2^{(PWSDIV)}$ + 1 when entering Power Saving Mode.

Warning: This value must be different from 0 before enabling the Power Save Mode in the HSMCI_CR (HSMCI_PWSEN bit).

RDPROOF: Read Proof Enable

Enabling Read Proof allows to stop the HSMCI Clock during read access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.

0: Disables Read Proof.

1: Enables Read Proof.

WRPROOF: Write Proof Enable

Enabling Write Proof allows to stop the HSMCI Clock during write access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.

0: Disables Write Proof.

1: Enables Write Proof.

FBYTE: Force Byte Transfer

Enabling Force Byte Transfer allow byte transfers, so that transfer of blocks with a size different from modulo 4 can be supported.

Warning: BLKLEN value depends on FBYTE.

0: Disables Force Byte Transfer.

1: Enables Force Byte Transfer.

PADV: Padding Value

0: 0x00 value is used when padding data in write transfer.

1: 0xFF value is used when padding data in write transfer.

PADV may be only in manual transfer.

CLKODD: Clock divider is odd

This bit is the least significant bit of the clock divider and indicates the clock divider parity.

34.14.3 HSMCI Data Timeout Register

Name: HSMCI_DTOR

Address:0xF0008008

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	DTOMUL			DTOCYC			

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

DTOCYC: Data Timeout Cycle Number

This field determines the maximum number of Master Clock cycles that the HSMCI waits between two data block transfers. It equals (DTOCYC x Multiplier).

DTOMUL: Data Timeout Multiplier

Value	Name	Description
0	1	DTOCYC
1	16	DTOCYC x 16
2	128	DTOCYC x 128
3	256	DTOCYC x 256
4	1024	DTOCYC x 1024
5	4096	DTOCYC x 4096
6	65536	DTOCYC x 65536
7	1048576	DTOCYC x 1048576

If the data time-out set by DTOCYC and DTOMUL has been exceeded, the Data Time-out Error flag (DTOE) in the HSMCI Status Register (HSMCI_SR) rises.

© 2017 Microchip Technology Inc.

34.14.4 HSMCI SDCard/SDIO Register

Name: HSMCI_SDCR

Address:0xF000800C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SDCBUS		–	–	–	–	SDCSEL	

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

SDCSEL: SDCard/SDIO Slot

Value	Name	Description
0	SLOTA	Slot A is selected.
1	SLOTB	–
2	SLOTC	–
3	SLOTD	–

SDCBUS: SDCard/SDIO Bus Width

Value	Name	Description
0	1	1 bit
1	–	Reserved
2	4	4 bits
3	8	8 bits

34.14.5 HSMCI Argument Register

Name: HSMCI_ARGR

Address:0xF0008010

Access: Read/Write

31	30	29	28	27	26	25	24
ARG							
23	22	21	20	19	18	17	16
ARG							
15	14	13	12	11	10	9	8
ARG							
7	6	5	4	3	2	1	0
ARG							

ARG: Command Argument

© 2017 Microchip Technology Inc.

34.14.6 HSMCI Command Register

Name: HSMCI_CMDR

Address:0xF0008014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	BOOT_ACK	ATACS	IOSPCMD	
23	22	21	20	19	18	17	16
–	–	TRTYP			TRDIR	TRCMD	
15	14	13	12	11	10	9	8
–	–	–	MAXLAT	OPDCMD	SPCMD		
7	6	5	4	3	2	1	0
RSPTYP		CMDNB					

This register is write-protected while CMDRDY is 0 in HSMCI_SR. If an Interrupt command is sent, this register is only writable by an interrupt response (field SPCMD). This means that the current command execution cannot be interrupted or modified.

CMDNB: Command Number

This is the command index.

RSPTYP: Response Type

Value	Name	Description
0	NORESP	No response
1	48_BIT	48-bit response
2	136_BIT	136-bit response
3	R1B	R1b response type

SPCMD: Special Command

Value	Name	Description
0	STD	Not a special CMD.
1	INIT	Initialization CMD: 74 clock cycles for initialization sequence.
2	SYNC	Synchronized CMD: Wait for the end of the current data block transfer before sending the pending command.
3	CE_ATA	CE-ATA Completion Signal disable Command. The host cancels the ability for the device to return a command completion signal on the command line.
4	IT_CMD	Interrupt command: Corresponds to the Interrupt Mode (CMD40).
5	IT_RESP	Interrupt response: Corresponds to the Interrupt Mode (CMD40).
6	BOR	Boot Operation Request. Start a boot operation mode, the host processor can read boot data from the MMC device directly.
7	EBO	End Boot Operation. This command allows the host processor to terminate the boot operation mode.

OPDCMD: Open Drain Command

0 (PUSHPULL): Push pull command.

1 (OPENDRAIN): Open drain command.

MAXLAT: Max Latency for Command to Response

0 (5): 5-cycle max latency.

1 (64): 64-cycle max latency.

TRCMD: Transfer Command

Value	Name	Description
0	NO_DATA	No data transfer
1	START_DATA	Start data transfer
2	STOP_DATA	Stop data transfer
3	–	Reserved

TRDIR: Transfer Direction

0 (WRITE): Write.

1 (READ): Read.

TRTYP: Transfer Type

Value	Name	Description
0	SINGLE	MMC/SD Card Single Block
1	MULTIPLE	MMC/SD Card Multiple Block
2	STREAM	MMC Stream
4	BYTE	SDIO Byte
5	BLOCK	SDIO Block

IOSPCMD: SDIO Special Command

Value	Name	Description
0	STD	Not an SDIO Special Command
1	SUSPEND	SDIO Suspend Command
2	RESUME	SDIO Resume Command

ATACS: ATA with Command Completion Signal

0 (NORMAL): Normal operation mode.

1 (COMPLETION): This bit indicates that a completion signal is expected within a programmed amount of time (HSMCI_CSTOR).

BOOT_ACK: Boot Operation Acknowledge

The master can choose to receive the boot acknowledge from the slave when a Boot Request command is issued. When set to one this field indicates that a Boot acknowledge is expected within a programmable amount of time defined with DTOMUL and DTOCYC fields located in the HSMCI_DTOR. If the acknowledge pattern is not received then an acknowledge timeout error is raised. If the acknowledge pattern is corrupted then an acknowledge pattern error is set.

© 2017 Microchip Technology Inc.

34.14.7 HSMCI Block Register

Name: HSMCI_BLKR

Address:0xF0008018

Access: Read/Write

31	30	29	28	27	26	25	24
BLKLEN							
23	22	21	20	19	18	17	16
BLKLEN							
15	14	13	12	11	10	9	8
BCNT							
7	6	5	4	3	2	1	0
BCNT							

BCNT: MMC/SDIO Block Count - SDIO Byte Count

This field determines the number of data byte(s) or block(s) to transfer.

The transfer data type and the authorized values for BCNT field are determined by the TRTYP field in the HSMCI Command Register (HSMCI_CMDR).

When TRTYP = 1 (MMC/SDCARD Multiple Block), BCNT can be programmed from 1 to 65535, 0 corresponds to an infinite block transfer.

When TRTYP = 4 (SDIO Byte), BCNT can be programmed from 1 to 511, 0 corresponds to 512-byte transfer. Values in range 512 to 65536 are forbidden.

When TRTYP = 5 (SDIO Block), BCNT can be programmed from 1 to 511, 0 corresponds to an infinite block transfer. Values in range 512 to 65536 are forbidden.

Warning: In SDIO Byte and Block modes (TRTYP = 4 or 5), writing the 7 last bits of BCNT field with a value which differs from 0 is forbidden and may lead to unpredictable results.

BLKLEN: Data Block Length

This field determines the size of the data block.

Bits 16 and 17 must be configured to 0 if FBYTE is disabled.

Note: In SDIO Byte mode, BLKLEN field is not used.

34.14.8 HSMCI Completion Signal Timeout Register

Name: HSMCI_CSTOR

Address:0xF000801C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	CSTOMUL			CSTOCYC			

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

CSTOCYC: Completion Signal Timeout Cycle Number

This field determines the maximum number of Master Clock cycles that the HSMCI waits between two data block transfers. Its value is calculated by (CSTOCYC x Multiplier).

CSTOMUL: Completion Signal Timeout Multiplier

This field determines the maximum number of Master Clock cycles that the HSMCI waits between two data block transfers. Its value is calculated by (CSTOCYC x Multiplier).

These fields determine the maximum number of Master Clock cycles that the HSMCI waits between the end of the data transfer and the assertion of the completion signal. The data transfer comprises data phase and the optional busy phase. If a non-DATA ATA command is issued, the HSMCI starts waiting immediately after the end of the response until the completion signal.

Multiplier is defined by CSTOMUL as shown in the following table:

Value	Name	Description
0	1	CSTOCYC x 1
1	16	CSTOCYC x 16
2	128	CSTOCYC x 128
3	256	CSTOCYC x 256
4	1024	CSTOCYC x 1024
5	4096	CSTOCYC x 4096
6	65536	CSTOCYC x 65536
7	1048576	CSTOCYC x 1048576

If the data time-out set by CSTOCYC and CSTOMUL has been exceeded, the Completion Signal Time-out Error flag (CSTOE) in the HSMCI Status Register (HSMCI_SR) rises.

© 2017 Microchip Technology Inc.

34.14.9 HSMCI Response Register

Name: HSMCI_RSPR

Address:0xF0008020

Access: Read-only

31	30	29	28	27	26	25	24
RSP							
23	22	21	20	19	18	17	16
RSP							
15	14	13	12	11	10	9	8
RSP							
7	6	5	4	3	2	1	0
RSP							

RSP: Response

Note 1: The response register can be read by N accesses at the same HSMCI_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

34.14.10 HSMCI Receive Data Register

Name: HSMCI_RDR

Address:0xF0008030

Access: Read-only

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

DATA: Data to Read

© 2017 Microchip Technology Inc.

34.14.11 HSMCI Transmit Data Register

Name: HSMCI_TDR

Address:0xF0008034

Access: Write-only

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

DATA: Data to Write

34.14.12 HSMCI Status Register

Name: HSMCI_SR

Address:0xF0008040

Access: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

CMDRDY: Command Ready (cleared by writing in HSMCI_CMDR)

0: A command is in progress.

1: The last command has been sent.

RXRDY: Receiver Ready (cleared by reading HSMCI_RDR)

0: Data has not yet been received since the last read of HSMCI_RDR.

1: Data has been received since the last read of HSMCI_RDR.

TXRDY: Transmit Ready (cleared by writing in HSMCI_TDR)

0: The last data written in HSMCI_TDR has not yet been transferred in the Shift Register.

1: The last data written in HSMCI_TDR has been transferred in the Shift Register.

BLKE: Data Block Ended (cleared on read)

This flag must be used only for Write Operations.

0: A data block transfer is not yet finished.

1: A data block transfer has ended, including the CRC16 Status transmission. The flag is set for each transmitted CRC Status.

Refer to the MMC or SD Specification for more details concerning the CRC Status.

DTIP: Data Transfer in Progress (cleared at the end of CRC16 calculation)

0: No data transfer in progress.

1: The current data transfer is still in progress, including CRC16 calculation.

NOTBUSY: HSMCI Not Busy

A block write operation uses a simple busy signalling of the write operation duration on the data (DAT0) line: during a data transfer block, if the card does not have a free data receive buffer, the card indicates this condition by pulling down the data line (DAT0) to LOW. The card stops pulling down the data line as soon as at least one receive buffer for the defined data transfer block length becomes free.

Refer to the MMC or SD Specification for more details concerning the busy behavior.

For all the read operations, the NOTBUSY flag is cleared at the end of the host command.
For the Infinite Read Multiple Blocks, the NOTBUSY flag is set at the end of the STOP_TRANSMISSION host command (CMD12).

For the Single Block Reads, the NOTBUSY flag is set at the end of the data read block.

For the Multiple Block Reads with predefined block count, the NOTBUSY flag is set at the end of the last received data block.

The NOTBUSY flag allows to deal with these different states.

0: The HSMCI is not ready for new data transfer. Cleared at the end of the card response.

1: The HSMCI is ready for new data transfer. Set when the busy state on the data line has ended. This corresponds to a free internal data receive buffer of the card.

© 2017 Microchip Technology Inc.

SDIOIRQA: SDIO Interrupt for Slot A (cleared on read)

0: No interrupt detected on SDIO Slot A.

1: An SDIO Interrupt on Slot A occurred.

SDIOWAIT: SDIO Read Wait Operation Status

0: Normal Bus operation.

1: The data bus has entered IO wait state.

CSRCV: CE-ATA Completion Signal Received (cleared on read)

0: No completion signal received since last status read operation.

1: The device has issued a command completion signal on the command line.

RINDE: Response Index Error (cleared by writing in HSMCI_CMDR)

0: No error.

1: A mismatch is detected between the command index sent and the response index received.

RDIRE: Response Direction Error (cleared by writing in HSMCI_CMDR)

0: No error.

1: The direction bit from card to host in the response has not been detected.

RCRCE: Response CRC Error (cleared by writing in HSMCI_CMDR)

0: No error.

1: A CRC7 error has been detected in the response.

RENDE: Response End Bit Error (cleared by writing in HSMCI_CMDR)

0: No error.

1: The end bit of the response has not been detected.

RTOE: Response Time-out Error (cleared by writing in HSMCI_CMDR)

0: No error.

1: The response time-out set by MAXLAT in the HSMCI_CMDR has been exceeded.

DCRCE: Data CRC Error (cleared on read)

0: No error.

1: A CRC16 error has been detected in the last data block.

DTOE: Data Time-out Error (cleared on read)

0: No error.

1: The data time-out set by DTOCYC and DTOMUL in HSMCI_DTOR has been exceeded.

CSTOE: Completion Signal Time-out Error (cleared on read)

0: No error.

1: The completion signal time-out set by CSTOCYC and CSTOMUL in HSMCI_CSTOR has been exceeded.

BLKOVRE: DMA Block Overrun Error (cleared on read)

0: No error.

1: A new block of data is received and the DMA controller has not started to move the current pending block, a block overrun is raised.

DMADONE: DMA Transfer Done (cleared on read)

0: DMA buffer transfer has not completed since the last read of the HSMCI_SR.

1: DMA buffer transfer has completed since the last read of the HSMCI_SR.

FIFOEMPTY: FIFO empty flag

0: FIFO contains at least one byte.

1: FIFO is empty.

XFRDONE: Transfer Done flag

0: A transfer is in progress.

1: Command Register is ready to operate and the data bus is in the idle state.

ACKRCV: Boot Operation Acknowledge Received (cleared on read)

0: No Boot acknowledge received since the last read of the HSMCI_SR.

1: A Boot acknowledge signal has been received since the last read of HSMCI_SR.

ACKRCVE: Boot Operation Acknowledge Error (cleared on read)

0: No boot operation error since the last read of HSMCI_SR

1: Corrupted Boot Acknowledge signal received since the last read of HSMCI_SR.

OVRE: Overrun (if FERRCTRL = 1, cleared by writing in HSMCI_CMDR or cleared on read if FERRCTRL = 0)

0: No error.

1: At least one 8-bit received data has been lost (not read).

If FERRCTRL = 1 in HSMCI_CFG, OVRE is cleared on read.

If FERRCTRL = 0 in HSMCI_CFG, OVRE is cleared by writing HSMCI_CMDR.

UNRE: Underrun (if FERRCTRL = 1, cleared by writing in HSMCI_CMDR or cleared on read if FERRCTRL = 0)

0: No error.

1: At least one 8-bit data has been sent without valid information (not written).

If FERRCTRL = 1 in HSMCI_CFG, OVRE is cleared on read.

If FERRCTRL = 0 in HSMCI_CFG, OVRE is cleared by writing HSMCI_CMDR.

© 2017 Microchip Technology Inc.

34.14.13 HSMCI Interrupt Enable Register

Name: HSMCI_IER

Address:0xF0008044

Access: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

CMDRDY: Command Ready Interrupt Enable

RXRDY: Receiver Ready Interrupt Enable

TXRDY: Transmit Ready Interrupt Enable

BLKE: Data Block Ended Interrupt Enable

DTIP: Data Transfer in Progress Interrupt Enable

NOTBUSY: Data Not Busy Interrupt Enable

SDIOIRQA: SDIO Interrupt for Slot A Interrupt Enable

SDIOWAIT: SDIO Read Wait Operation Status Interrupt Enable

CSRCV: Completion Signal Received Interrupt Enable

RINDE: Response Index Error Interrupt Enable

RDIRE: Response Direction Error Interrupt Enable

RCRCE: Response CRC Error Interrupt Enable

RENDE: Response End Bit Error Interrupt Enable

RTOE: Response Time-out Error Interrupt Enable

DCRCE: Data CRC Error Interrupt Enable

DTOE: Data Time-out Error Interrupt Enable

CSTOE: Completion Signal Timeout Error Interrupt Enable

BLKOVRE: DMA Block Overrun Error Interrupt Enable

DMADONE: DMA Transfer completed Interrupt Enable

FIFOEMPTY: FIFO empty Interrupt enable

XFRDONE: Transfer Done Interrupt enable

ACKRCV: Boot Acknowledge Interrupt Enable

© 2017 Microchip Technology Inc.

ACKRCVE: Boot Acknowledge Error Interrupt Enable

OVRE: Overrun Interrupt Enable

UNRE: Underrun Interrupt Enable

© 2017 Microchip Technology Inc.

34.14.14 HSMCI Interrupt Disable Register

Name: HSMCI_IDR

Address:0xF0008048

Access: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

CMDRDY: Command Ready Interrupt Disable

RXRDY: Receiver Ready Interrupt Disable

TXRDY: Transmit Ready Interrupt Disable

BLKE: Data Block Ended Interrupt Disable

DTIP: Data Transfer in Progress Interrupt Disable

NOTBUSY: Data Not Busy Interrupt Disable

SDIOIRQA: SDIO Interrupt for Slot A Interrupt Disable

SDIOWAIT: SDIO Read Wait Operation Status Interrupt Disable

CSRCV: Completion Signal received interrupt Disable

RINDE: Response Index Error Interrupt Disable

RDIRE: Response Direction Error Interrupt Disable

RCRCE: Response CRC Error Interrupt Disable

RENDE: Response End Bit Error Interrupt Disable

RTOE: Response Time-out Error Interrupt Disable

DCRCE: Data CRC Error Interrupt Disable

DTOE: Data Time-out Error Interrupt Disable

CSTOE: Completion Signal Time out Error Interrupt Disable

BLKOVRE: DMA Block Overrun Error Interrupt Disable

DMADONE: DMA Transfer completed Interrupt Disable

FIFOEMPTY: FIFO empty Interrupt Disable

XFRDONE: Transfer Done Interrupt Disable

ACKRCV: Boot Acknowledge Interrupt Disable

ACKRCVE: Boot Acknowledge Error Interrupt Disable

OVRE: Overrun Interrupt Disable

UNRE: Underrun Interrupt Disable

© 2017 Microchip Technology Inc.

34.14.15 HSMCI Interrupt Mask Register

Name: HSMCI_IMR

Address:0xF000804C

Access: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

CMDRDY: Command Ready Interrupt Mask

RXRDY: Receiver Ready Interrupt Mask

TXRDY: Transmit Ready Interrupt Mask

BLKE: Data Block Ended Interrupt Mask

DTIP: Data Transfer in Progress Interrupt Mask

NOTBUSY: Data Not Busy Interrupt Mask

SDIOIRQA: SDIO Interrupt for Slot A Interrupt Mask

SDIOWAIT: SDIO Read Wait Operation Status Interrupt Mask

CSRCV: Completion Signal Received Interrupt Mask

RINDE: Response Index Error Interrupt Mask

RDIRE: Response Direction Error Interrupt Mask

RCRCE: Response CRC Error Interrupt Mask

RENDE: Response End Bit Error Interrupt Mask

RTOE: Response Time-out Error Interrupt Mask

DCRCE: Data CRC Error Interrupt Mask

DTOE: Data Time-out Error Interrupt Mask

CSTOE: Completion Signal Time-out Error Interrupt Mask

BLKOVRE: DMA Block Overrun Error Interrupt Mask

DMADONE: DMA Transfer Completed Interrupt Mask

FIFOEMPTY: FIFO Empty Interrupt Mask

XFRDONE: Transfer Done Interrupt Mask

ACKRCV: Boot Operation Acknowledge Received Interrupt Mask

© 2017 Microchip Technology Inc.

ACKRCVE: Boot Operation Acknowledge Error Interrupt Mask

OVRE: Overrun Interrupt Mask

UNRE: Underrun Interrupt Mask

© 2017 Microchip Technology Inc.

34.14.16 HSMCI DMA Configuration Register

Name: HSMCI_DMA

Address:0xF0008050

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	ROPT	–	–	–	DMAEN
7	6	5	4	3	2	1	0
–	CHKSIZE			–	–	OFFSET	

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

OFFSET: DMA Write Buffer Offset

This field indicates the number of discarded bytes when the DMA writes the first word of the transfer.

CHKSIZE: DMA Channel Read and Write Chunk Size

The CHKSIZE field indicates the number of data available when the DMA chunk transfer request is asserted.

Value	Name	Description
0	1	1 data available
1	4	4 data available
2	8	8 data available
3	16	16 data available
4	32	32 data available

DMAEN: DMA Hardware Handshaking Enable

0: DMA interface is disabled.

1: DMA Interface is enabled.

Note: To avoid unpredictable behavior, DMA hardware handshaking must be disabled when CPU transfers are performed.

ROPT: Read Optimization with padding

0: BLKLEN bytes are moved from the Memory Card to the system memory; two DMA descriptors are used when the transfer size is not a multiple of 4.

1: CEILING(BLKLEN/4) * 4 bytes are moved from the Memory Card to the system memory; only one DMA descriptor is used.

© 2017 Microchip Technology Inc.

34.14.17 HSMCI Configuration Register

Name: HSMCI_CFG

Address:0xF0008054

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	LSYNC	–	–	–	HSMODE
7	6	5	4	3	2	1	0
–	–	–	FERRCTRL	–	–	–	FIFOMODE

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

FIFOMODE: HSMCI Internal FIFO control mode

0: A write transfer starts when a sufficient amount of data is written into the FIFO.

When the block length is greater than or equal to 3/4 of the HSMCI internal FIFO size, then the write transfer starts as soon as half the FIFO is filled. When the block length is greater than or equal to half the internal FIFO size, then the write transfer starts as soon as one quarter of the FIFO is filled. In other cases, the transfer starts as soon as the total amount of data is written in the internal FIFO.

1: A write transfer starts as soon as one data is written into the FIFO.

FERRCTRL: Flow Error flag reset control mode

0: When an underflow/overflow condition flag is set, a new Write/Read command is needed to reset the flag.

1: When an underflow/overflow condition flag is set, a read status resets the flag.

HSMODE: High Speed Mode

0: Default bus timing mode.

1: If set to one, the host controller outputs command line and data lines on the rising edge of the card clock. The Host driver shall check the high speed support in the card registers.

LSYNC: Synchronize on the last block

0: The pending command is sent at the end of the current data block.

1: The pending command is sent at the end of the block transfer when the transfer length is not infinite (block count shall be different from zero).

© 2017 Microchip Technology Inc.

34.14.18 HSMCI Write Protection Mode Register

Name: HSMCI_WPMR

Address:0xF00080E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

WPEN: Write Protect Enable

0: Disables the Write Protection if WPKEY corresponds to 0x4D4349 (“MCI” in ASCII).

1: Enables the Write Protection if WPKEY corresponds to 0x4D4349 (“MCI” in ASCII).

See Section 34.13 “Register Write Protection” for the list of registers that can be write-protected.

WPKEY: Write Protect Key

Value	Name	Description
0x4D4349	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

34.14.19 HSMCI Write Protection Status Register

Name: HSMCI_WPSR

Address:0xF00080E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
WPVSRC							
15	14	13	12	11	10	9	8
WPVSRC							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

WPVS: Write Protection Violation Status

0: No write protection violation has occurred since the last read of the HSMCI_WPSR.

1: A write protection violation has occurred since the last read of the HSMCI_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

WPVSRC: Write Protection Violation Source

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

© 2017 Microchip Technology Inc.

34.14.20 HSMCI FIFOx Memory Aperture

Name: HSMCI_FIFOx [x=0..255]

Address:0xF0008200

Access: Read/Write

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

DATA: Data to Read or Data to Write

35. Serial Peripheral Interface (SPI)

35.1 Description

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a Shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turn being masters (multiple master protocol, contrary to single master protocol where one CPU is always the master while all of the others are always slaves). One master can simultaneously shift data into multiple slaves. However, only one slave can drive its output to write data back to the master at any given time.

A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI)—This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO)—This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK)—This control line is driven by the master and regulates the flow of the data bits. The master can transmit data at a variety of baud rates; there is one SPCK pulse for each bit that is transmitted.
- Slave Select (NSS)—This control line allows slaves to be turned on and off by hardware.

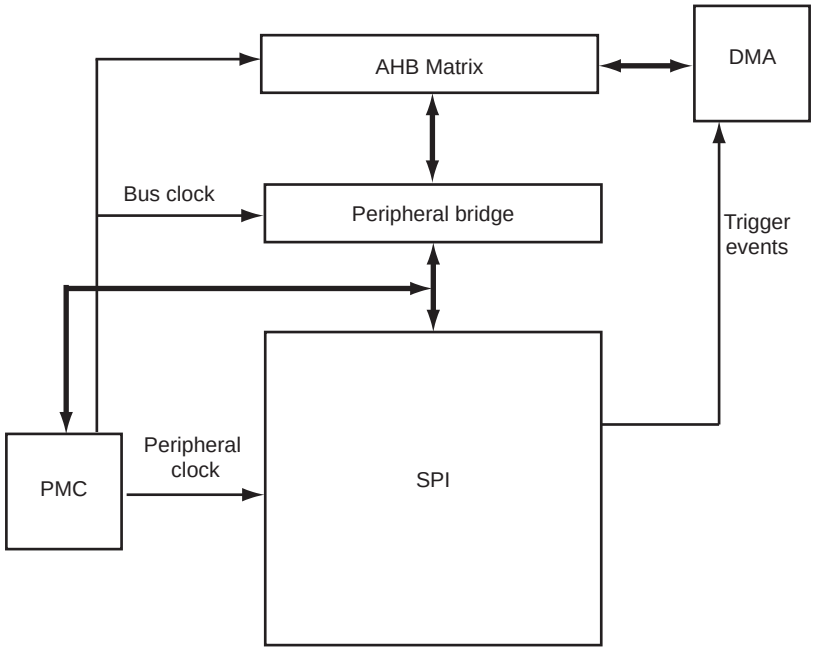
35.2 Embedded Characteristics

- Master or Slave Serial Peripheral Bus Interface
  - 8-bit to 16-bit programmable data length per chip select
  - Programmable phase and polarity per chip select
  - Programmable transfer delay between consecutive transfers and delay before SPI clock per chip select
  - Programmable delay between chip selects
  - Selectable mode fault detection
- Master Mode can drive SPCK up to Peripheral Clock
- Master Mode Bit Rate can be Independent of the Processor/Peripheral Clock
- Slave mode operates on SPCK, asynchronously with core and bus clock
- Four chip selects with external decoder support allow communication with up to 15 peripherals
- Communication with Serial External Devices Supported
  - Serial memories, such as DataFlash and 3-wire EEPROMs
  - Serial peripherals, such as ADCs, DACs, LCD controllers, CAN controllers and sensors
  - External coprocessors
- Connection to DMA Channel Capabilities, Optimizing Data Transfers
  - One channel for the receiver
  - One channel for the transmitter
- Register Write Protection

© 2017 Microchip Technology Inc.

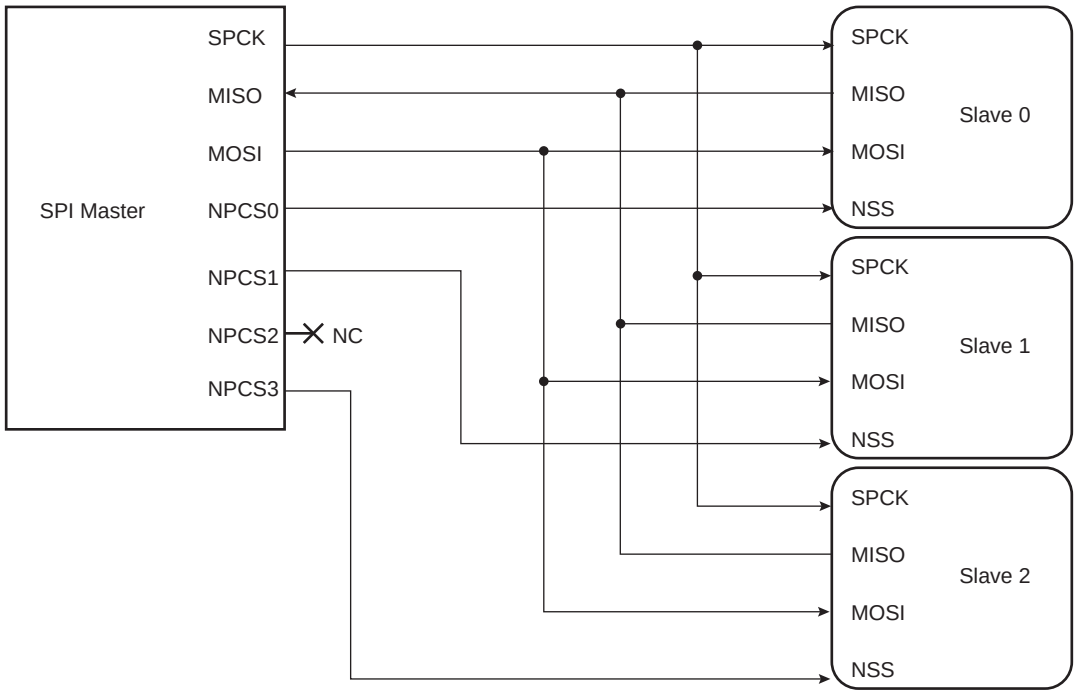
35.3 Block Diagram

Figure 35-1: Block Diagram



35.4 Application Block Diagram

Figure 35-2: Application Block Diagram: Single Master/Multiple Slave Implementation



35.5 Signal Description

Table 35-1: Signal Description

Pin Name	Pin Description	Type	
		Master	Slave
MISO	Master In Slave Out	Input	Output
MOSI	Master Out Slave In	Output	Input
SPCK	Serial Clock	Output	Input
NPCS1–NPCS3	Peripheral Chip Selects	Output	Unused
NPCS0/NSS	Peripheral Chip Select/Slave Select	Output	Input

35.6 Product Dependencies

35.6.1 I/O Lines

The pins used for interfacing the compliant external devices can be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions.

Table 35-2: I/O Lines

Instance	Signal	I/O Line	Peripheral
SPI0	SPI0_MISO	PA11	A
SPI0	SPI0_MOSI	PA12	A
SPI0	SPI0_NPCS0	PA14	A
SPI0	SPI0_NPCS1	PA7	B
SPI0	SPI0_NPCS2	PA1	B
SPI0	SPI0_NPCS3	PB3	B
SPI0	SPI0_SPCK	PA13	A
SPI1	SPI1_MISO	PA21	B
SPI1	SPI1_MOSI	PA22	B
SPI1	SPI1_NPCS0	PA8	B
SPI1	SPI1_NPCS1	PA0	B
SPI1	SPI1_NPCS2	PA31	B
SPI1	SPI1_NPCS3	PA30	B
SPI1	SPI1_SPCK	PA23	B

35.6.2 Power Management

The SPI can be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SPI clock.

© 2017 Microchip Technology Inc.

35.6.3 Interrupt

The SPI interface has an interrupt line connected to the interrupt controller. Handling the SPI interrupt requires programming the interrupt controller before configuring the SPI.

Table 35-3: Peripheral IDs

Instance	ID
SPI0	13
SPI1	14

35.6.4 Direct Memory Access Controller (DMAC)

The SPI interface can be used in conjunction with the DMAC in order to reduce processor overhead. For a full description of the DMAC, refer to the relevant section.

35.7 Functional Description

35.7.1 Modes of Operation

The SPI operates in Master mode or in Slave mode.

- The SPI operates in Master mode by setting the MSTR bit in the SPI Mode Register (SPI_MR):
  - Pins NPCS0 to NPCS3 are all configured as outputs
  - The SPCK pin is driven
  - The MISO line is wired on the receiver input
  - The MOSI line is driven as an output by the transmitter.
- The SPI operates in Slave mode if the MSTR bit in the SPI_MR is written to 0:
  - The MISO line is driven by the transmitter output
  - The MOSI line is wired on the receiver input
  - The SPCK pin is driven by the transmitter to synchronize the receiver.
  - The NPCS0 pin becomes an input, and is used as a slave select signal (NSS)
  - NPCS1 to NPCS3 are not driven and can be used for other purposes.

The data transfers are identically programmable for both modes of operation. The baud rate generator is activated only in Master mode.

35.7.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the SPI chip select registers (SPI_CSRx). The clock phase is programmed with the NCPHA bit. These two parameters determine the edges of the clock signal on which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Consequently, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are connected and require different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

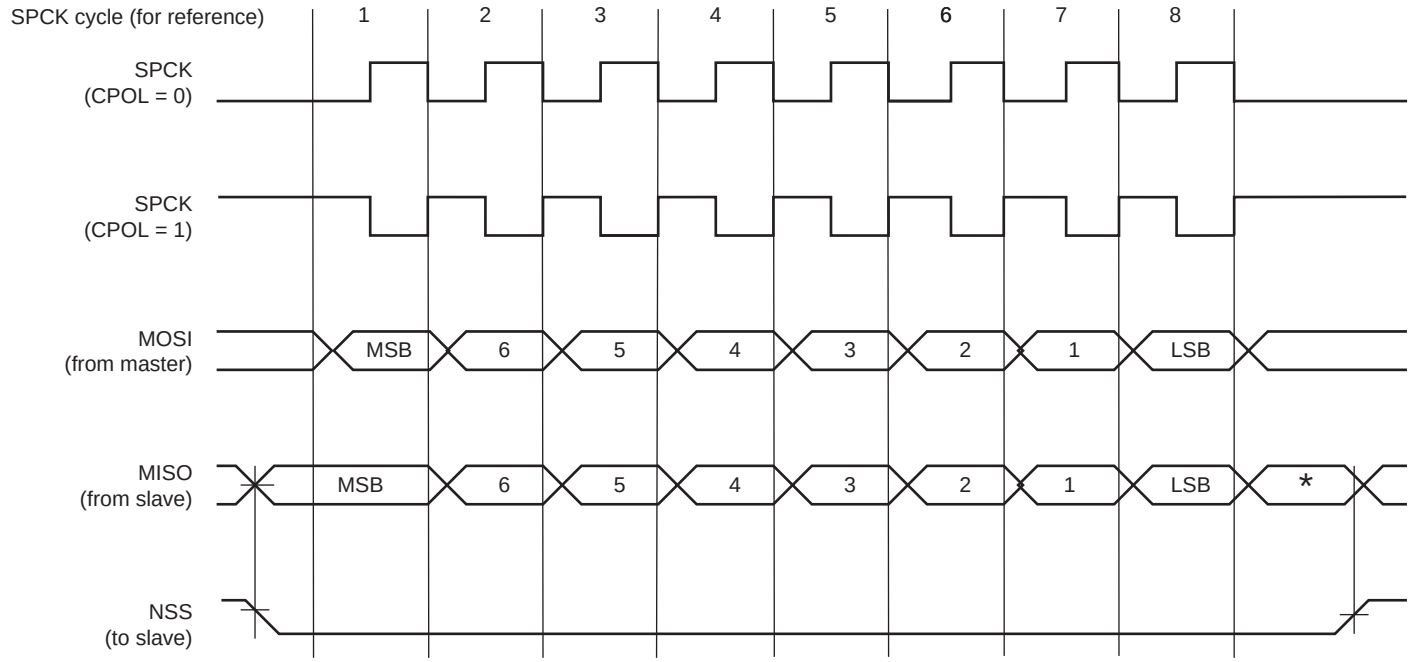
Table 35-4 shows the four modes and corresponding parameter settings.

Table 35-4: SPI Bus Protocol Modes

SPI Mode	CPOL	NCPHA	Shift SPCK Edge	Capture SPCK Edge	SPCK Inactive Level
0	0	1	Falling	Rising	Low
1	0	0	Rising	Falling	Low
2	1	1	Rising	Falling	High
3	1	0	Falling	Rising	High

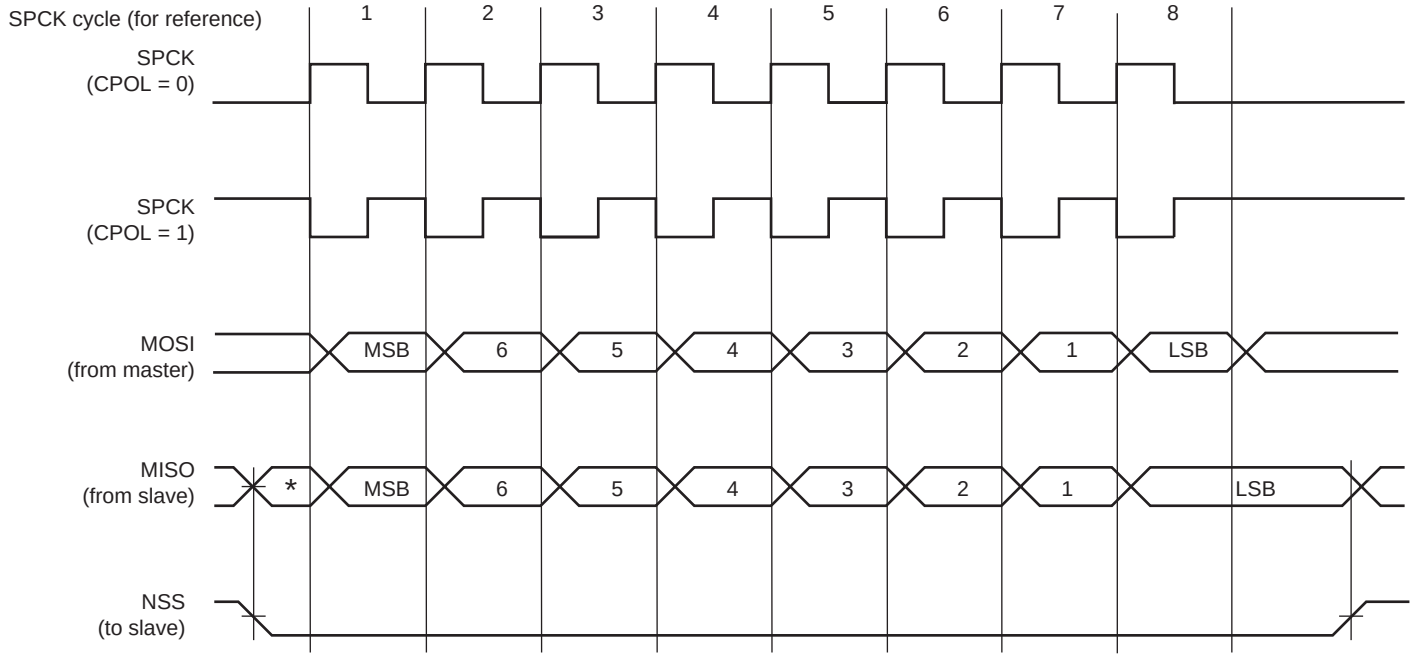
Figure 35-3 and Figure 35-4 show examples of data transfers.

Figure 35-3: SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



* Not defined.

Figure 35-4: SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



* Not defined.

© 2017 Microchip Technology Inc.

35.7.3 Master Mode Operations

When configured in Master mode, the SPI operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register (SPI_TDR) and the Receive Data Register (SPI_RDR), and a single shift register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer starts when the processor writes to the SPI_TDR. The written data is immediately transferred in the Shift register and the transfer on the SPI bus starts. While the data in the Shift register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift register. Data cannot be loaded in the SPI_RDR without transmitting data. If there is no data to transmit, dummy data can be used (SPI_TDR filled with ones). When the SPI_MR.WDRBT bit is set, new data cannot be transmitted if the SPI_RDR has not been read. If Receiving mode is not required, for example when communicating with a slave receiver only (such as an LCD), the receive status flags in the SPI Status register (SPI_SR) can be discarded.

Before writing the SPI_TDR, the PCS field in the SPI_MR must be set in order to select a slave.

If new data is written in the SPI_TDR during the transfer, it is kept in the SPI_TDR until the current transfer is completed. Then, the received data is transferred from the Shift register to the SPI_RDR, the data in the SPI_TDR is loaded in the Shift register and a new transfer starts.

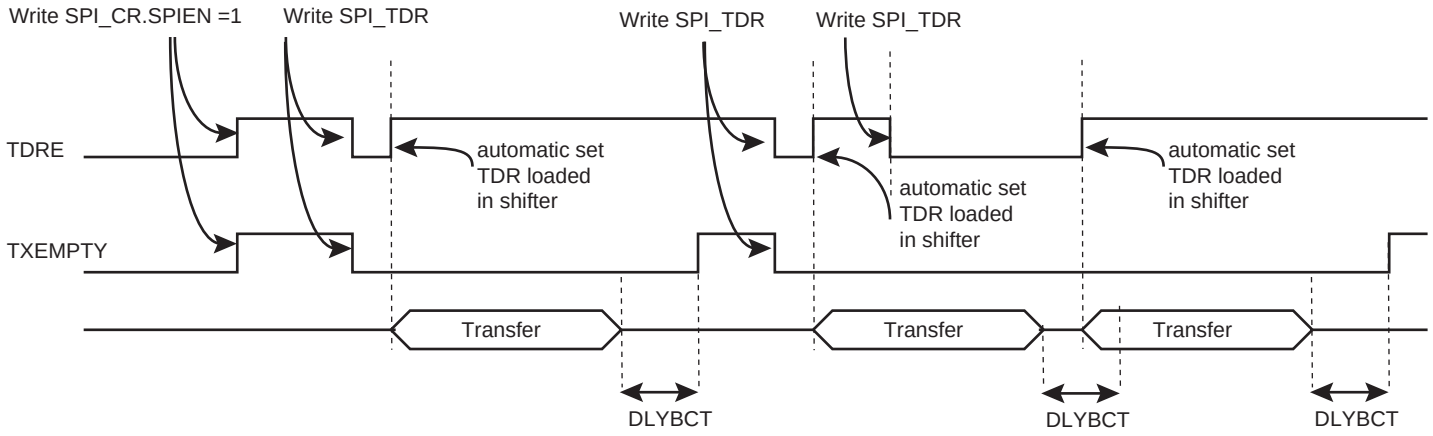
As soon as the SPI_TDR is written, the Transmit Data Register Empty (TDRE) flag in the SPI_SR is cleared. When the data written in the SPI_TDR is loaded into the Shift register, the TDRE flag in the SPI_SR is set. The TDRE bit is used to trigger the Transmit DMA channel.

See Figure 35-5.

The end of transfer is indicated by the TXEMPTY flag in the SPI_SR. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of this delay. The peripheral clock can be switched off at this time.

Note: When the SPI is enabled, the TDRE and TXEMPTY flags are set.

Figure 35-5: TDRE and TXEMPTY flag behavior



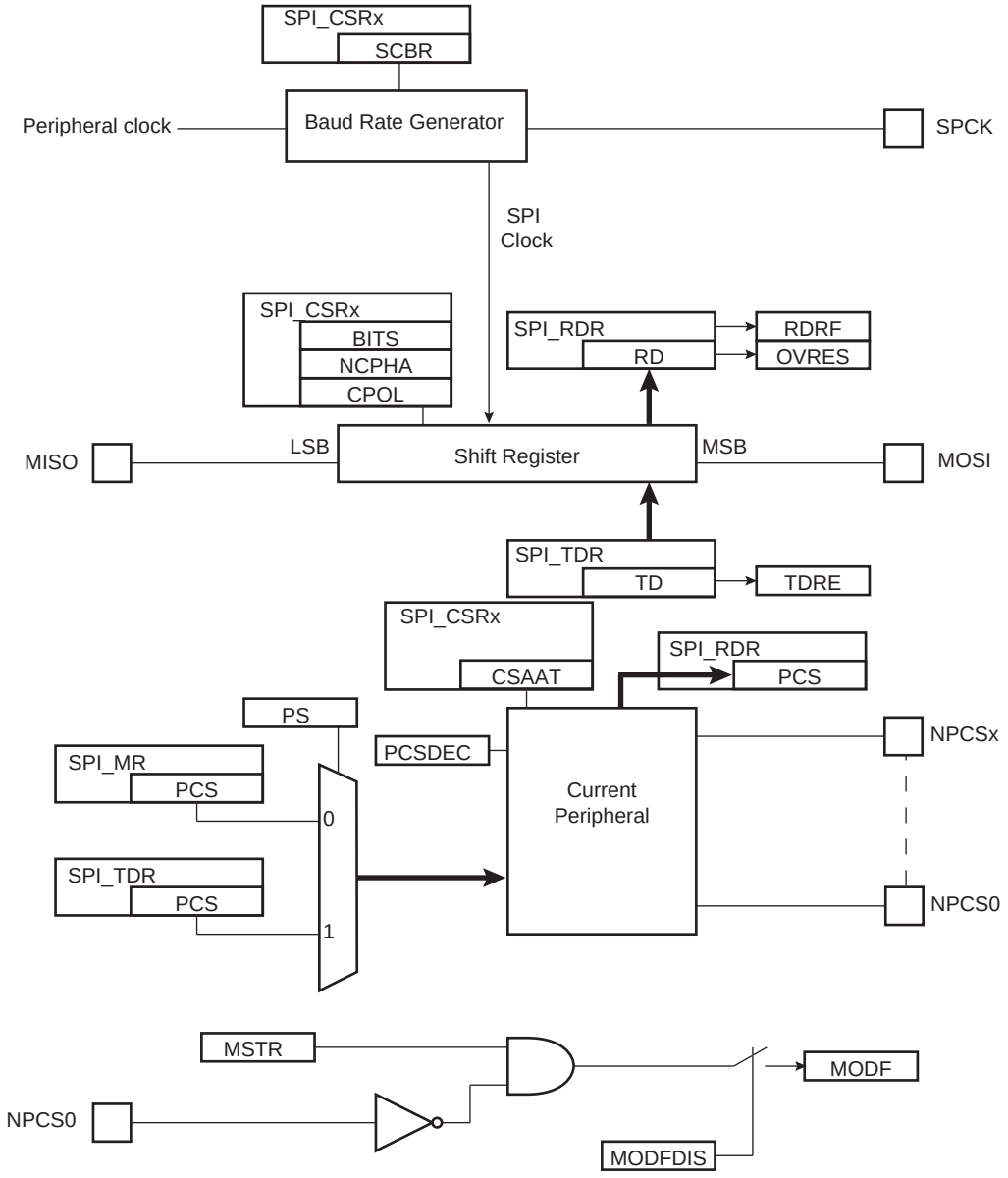
The transfer of received data from the Shift register to the SPI_RDR is indicated by the Receive Data Register Full (RDRF) bit in the SPI_SR. When the received data is read, the RDRF bit is cleared.

If the SPI_RDR has not been read before new data is received, the Overrun Error (OVRES) bit in the SPI_SR is set. As long as this flag is set, data is loaded in the SPI_RDR. The user has to read the SPI_SR to clear the OVRES bit.

Figure 35-6 shows a block diagram of the SPI when operating in Master mode. Figure 35-7 shows a flow chart describing how transfers are handled.

35.7.3.1 Master Mode Block Diagram

Figure 35-6: Master Mode Block Diagram



© 2017 Microchip Technology Inc.

35.7.3.2 Master Mode Flow Diagram

Figure 35-7: Master Mode Flow Diagram

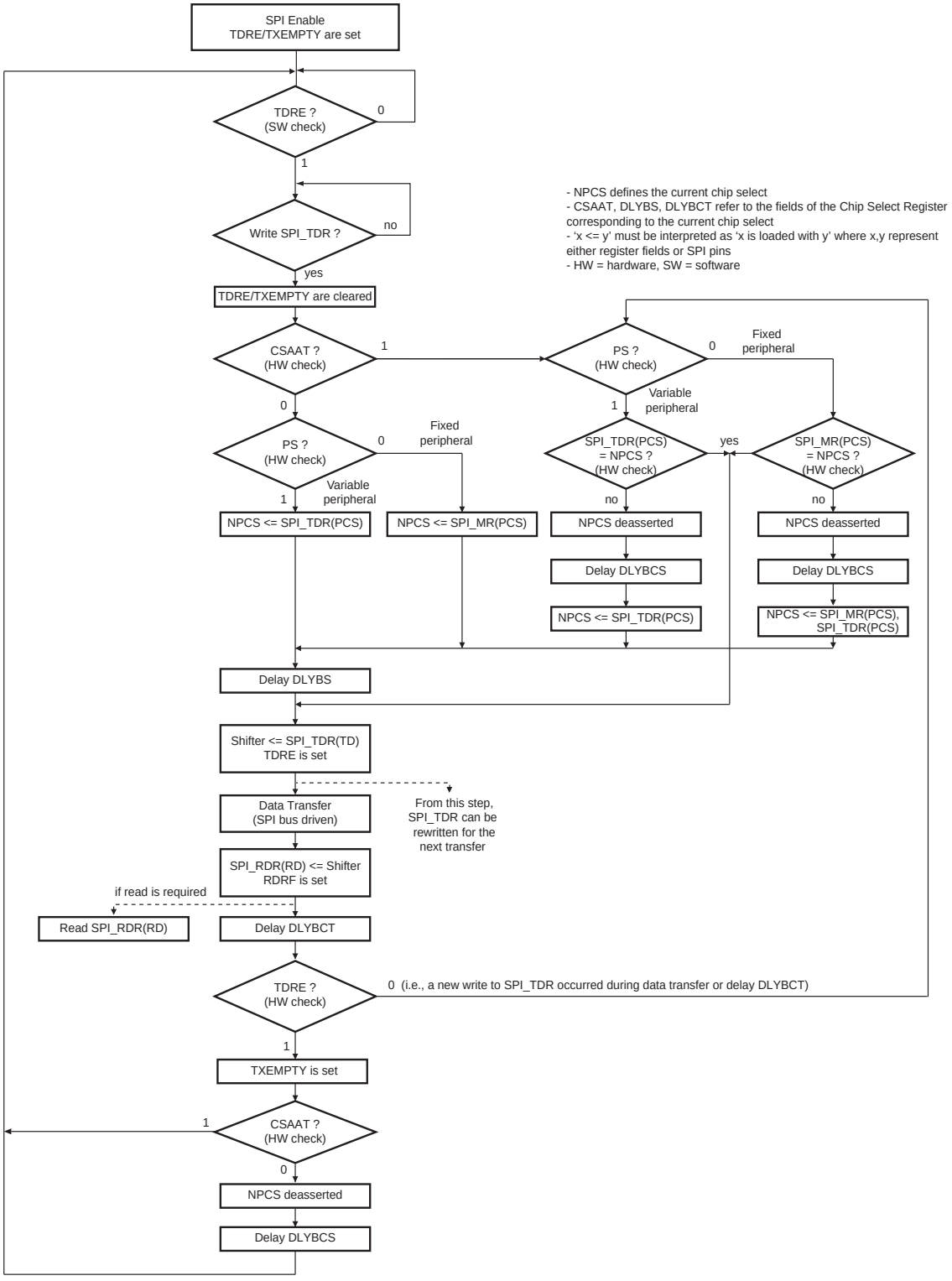
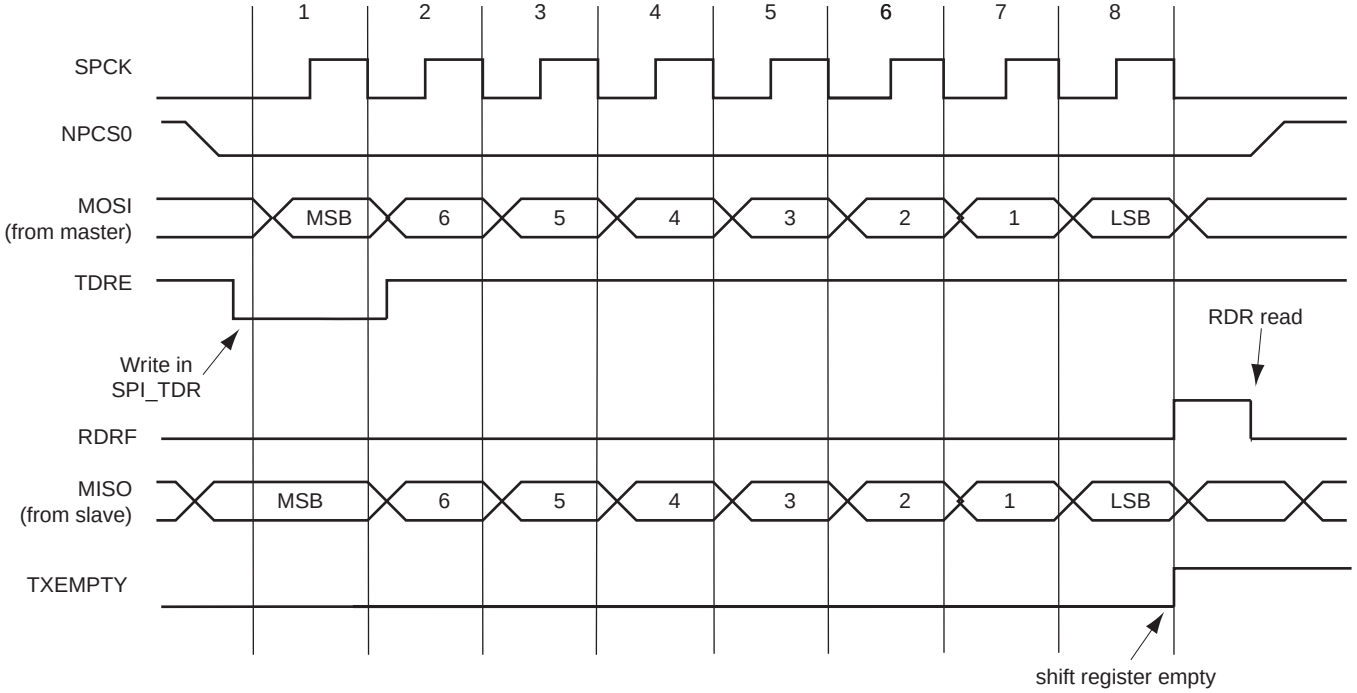


Figure 35-8 shows the behavior of Transmit Data Register Empty (TDRE), Receive Data Register (RDRF) and Transmission Register Empty (TXEMPTY) status flags within the SPI_SR during an 8-bit data transfer in Fixed mode without the DMA involved.

Figure 35-8: Status Register Flags Behavior



35.7.3.3 Clock Generation

The SPI Baud rate clock is generated by dividing the peripheral clock by a value between 1 and 255.

If the SCBR field in the SPI_CSR is programmed to 1, the operating baud rate is peripheral clock (see the electrical characteristics section for the SPCK maximum frequency). Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it to a valid value before performing the first transfer.

The divisor can be defined independently for each chip select, as it has to be programmed in the SCBR field. This allows the SPI to automatically adapt the baud rate for each interfaced peripheral without reprogramming.

35.7.3.4 Transfer Delays

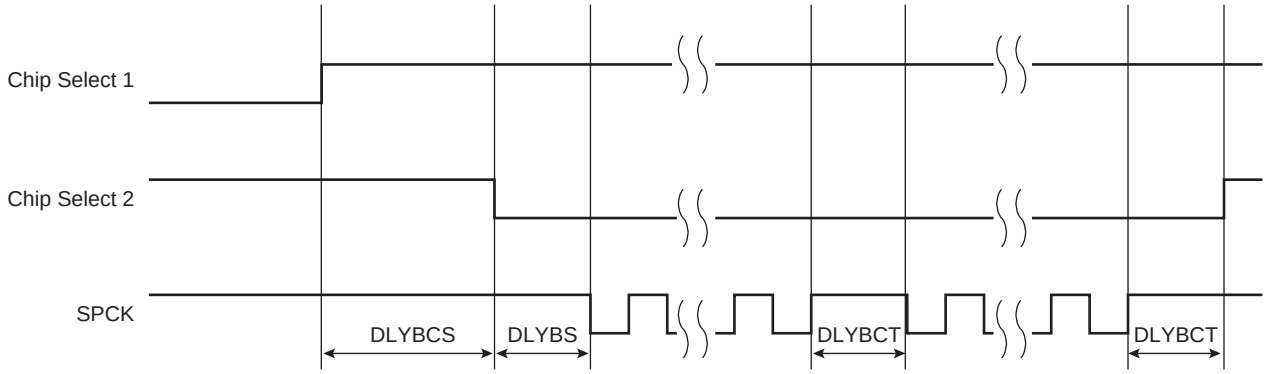
Figure 35-9 shows a chip select transfer change and consecutive transfers on the same chip select. Three delays can be programmed to modify the transfer waveforms:

- Delay between the chip selects—programmable only once for all chip selects by writing the DLYBCS field in the SPI_MR. The SPI slave device deactivation delay is managed through DLYBCS. If there is only one SPI slave device connected to the master, the DLYBCS field does not need to be configured. If several slave devices are connected to a master, DLYBCS must be configured depending on the highest deactivation delay. Refer to the SPI slave device electrical characteristics.
- Delay before SPCK—independently programmable for each chip select by writing the DLYBS field. The SPI slave device activation delay is managed through DLYBS. Refer to the SPI slave device electrical characteristics to define DLYBS.
- Delay between consecutive transfers—independently programmable for each chip select by writing the DLYBCT field. The time required by the SPI slave device to process received data is managed through DLYBCT. This time depends on the SPI slave system activity.

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

© 2017 Microchip Technology Inc.

Figure 35-9: Programmable Delays



35.7.3.5 Peripheral Selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all NPCS signals are high before and after each transfer.

- **Fixed Peripheral Select Mode**: SPI exchanges data with only one peripheral.
  Fixed Peripheral Select mode is enabled by clearing the PS bit in the SPI_MR. In this case, the current peripheral is defined by the PCS field in the SPI_MR and the PCS field in the SPI_TDR has no effect.
- **Variable Peripheral Select Mode:** Data can be exchanged with more than one peripheral without having to reprogram the NPCS field in the SPI_MR.
  Variable Peripheral Select mode is enabled by setting the PS bit in the SPI_MR. The PCS field in the SPI_TDR is used to select the current peripheral. This means that the peripheral selection can be defined for each new data. The value to write in the SPI_TDR has the following format:

  [xxxxxxx(7-bit) + LASTXFER(1-bit)(1)+ xxxx(4-bit) + PCS (4-bit) + DATA (8 to 16-bit)] with PCS equals the chip select to assert, as defined in Section 35.8.4 “SPI Transmit Data Register” and LASTXFER bit at 0 or 1 depending on the CSAAT bit.

  Note 1: Optional

  CSAAT, LASTXFER and CSNAAT bits are discussed in Section 35.7.3.9 “Peripheral Deselection with DMA”.

  If LASTXFER is used, the command must be issued after writing the last character. Instead of LASTXFER, the user can use the SPIDIS command. After the end of the DMA transfer, it is necessary to wait for the TXEMPTY flag and then write SPIDIS into the SPI Control Register (SPI_CR). This does not change the configuration register values). The NPCS is disabled after the last character transfer. Then, another DMA transfer can be started if the SPIEN has previously been written in the SPI_CR.

35.7.3.6 SPI Direct Access Memory Controller (DMAC)

In both Fixed and Variable modes, the Direct Memory Access Controller (DMAC) can be used to reduce processor overhead.

The fixed peripheral selection allows buffer transfers with a single peripheral. Using the DMAC is an optimal means, as the size of the data transfer between the memory and the SPI is either 8 bits or 16 bits. However, if the peripheral selection is modified, the SPI_MR must be reprogrammed.

The variable peripheral selection allows buffer transfers with multiple peripherals without reprogramming the SPI_MR. Data written in the SPI_TDR is 32 bits wide and defines the real data to be transmitted and the destination peripheral. Using the DMAC in this mode requires 32-bit wide buffers, with the data in the LSBs and the PCS and LASTXFER fields in the MSBs. However, the SPI still controls the number of bits (8 to 16) to be transferred through MISO and MOSI lines with the chip select configuration registers. This is not the optimal means in terms of memory size for the buffers, but it provides a very effective means to exchange data with several peripherals without any intervention of the processor.

35.7.3.7 Peripheral Chip Select Decoding

The user can program the SPI to operate with up to 15 slave peripherals by decoding the four chip select lines, NPCS0 to NPCS3 with an external decoder/demultiplexer (refer to Figure 35-10). This can be enabled by setting the PCSDEC bit in the SPI_MR.

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e., one NPCS line driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

When operating with decoding, the SPI directly outputs the value defined by the PCS field on the NPCS lines of either SPI_MR or SPI_TDR (depending on PS).

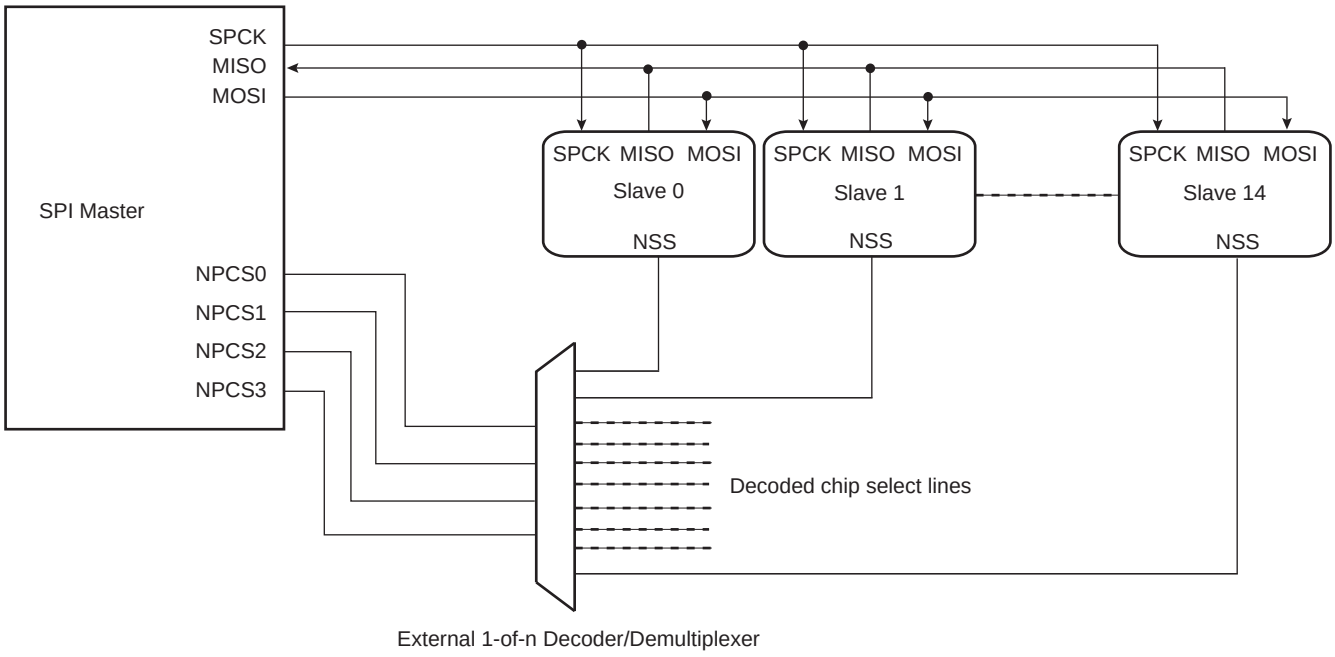
© 2017 Microchip Technology Inc.

As the SPI sets a default value of 0xF on the chip select lines (i.e., all chip select lines at 1) when not processing any transfer, only 15 peripherals can be decoded.

The SPI has four chip select registers (SPI_CSR0...SPI_CSR3). As a result, when external decoding is activated, each NPCS chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Consequently, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14. Figure 35-10 shows this type of implementation.

If the CSAAT bit is used, with or without the DMAC, the Mode Fault detection for NPCS0 line must be disabled. This is not needed for all other chip select lines since Mode Fault detection is only on NPCS0.

Figure 35-10: Chip Select Decoding Application Block Diagram: Single Master/Multiple Slave Implementation



35.7.3.8 Peripheral Deselection without DMA

During a transfer of more than one unit of data on a chip select without the DMA, the SPI_TDR is loaded by the processor, the TDRE flag rises as soon as the content of the SPI_TDR is transferred into the internal Shift register. When this flag is detected high, the SPI_TDR can be reloaded. If this reload by the processor occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the chip select is not de-asserted between the two transfers. But depending on the application software handling the SPI status register flags (by interrupt or polling method) or servicing other interrupts or other tasks, the processor may not reload the SPI_TDR in time to keep the chip select active (low). A null DLYBCT value (delay between consecutive transfers) in the SPI_CSR, gives even less time for the processor to reload the SPI_TDR. With some SPI slave peripherals, if the chip select line must remain active (low) during a full set of transfers, communication errors can occur.

To facilitate interfacing with such devices, the chip select registers [SPI_CSR0...SPI_CSR3] can be programmed with the Chip Select Active After Transfer (CSAAT) bit at 1. This allows the chip select lines to remain in their current state (low = active) until a transfer to another chip select is required. Even if the SPI_TDR is not reloaded, the chip select remains active. To de-assert the chip select line at the end of the transfer, the Last Transfer (LASTXFER) bit in SPI_CR must be set after writing the last data to transmit into SPI_TDR.

35.7.3.9 Peripheral Deselection with DMA

DMA provides faster reloads of the SPI_TDR compared to software. However, depending on the system activity, it is not guaranteed that the SPI_TDR is written with the next data before the end of the current transfer. Consequently, data can be lost by the de-assertion of the NPCS line for SPI slave peripherals requiring the chip select line to remain active between two transfers. The only way to guarantee a safe transfer in this case is the use of the CSAAT and LASTXFER bits.

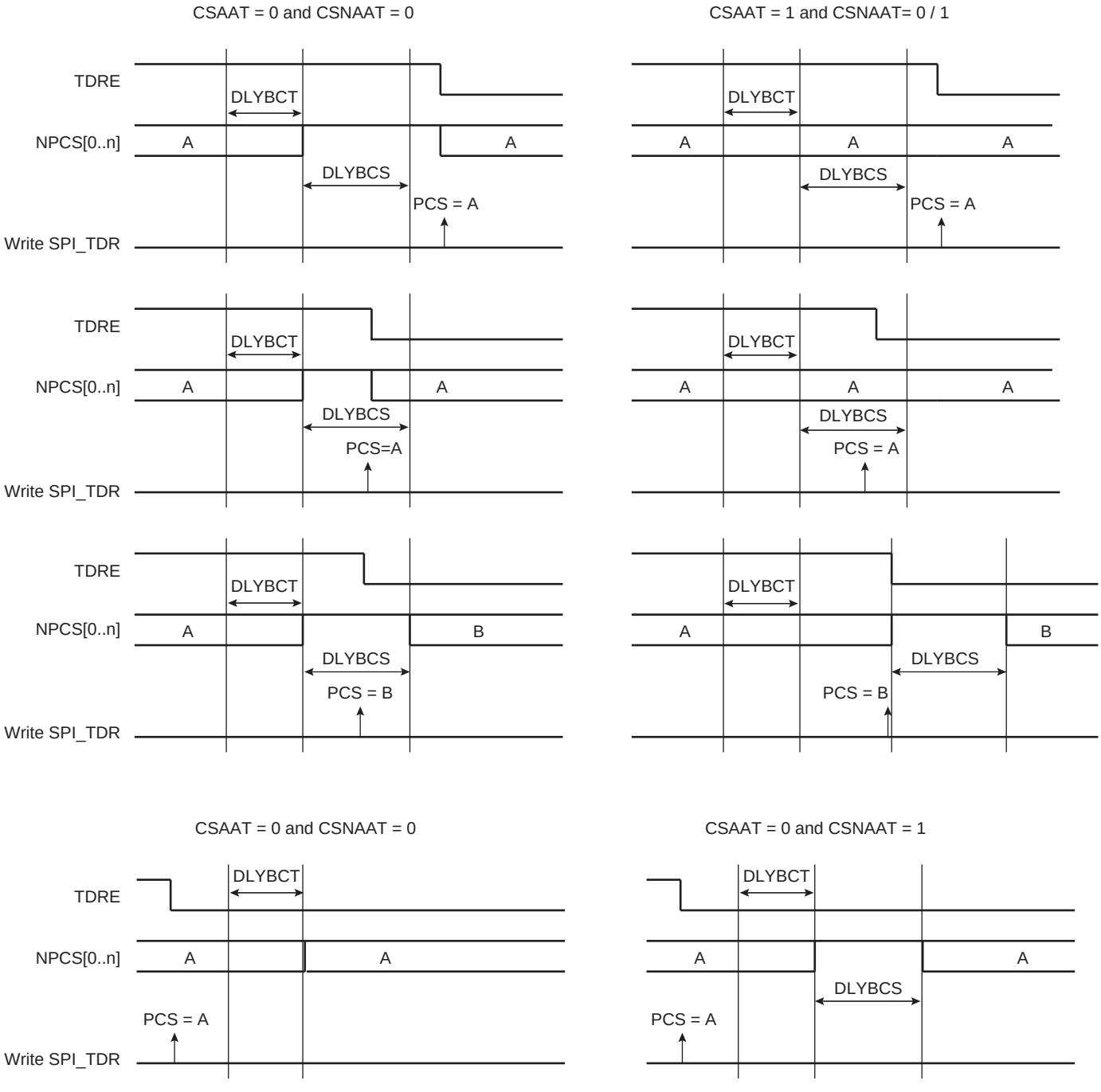
When the CSAAT bit is configured to 0, the NPCS does not rise in all cases between two transfers on the same peripheral. During a transfer on a chip select, the TDRE flag rises as soon as the content of the SPI_TDR is transferred into the internal shift register. When this flag is detected, the SPI_TDR can be reloaded. If this reload occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the chip select is not de-asserted between the two transfers. This can lead to difficulties

© 2017 Microchip Technology Inc.

to interface with some serial peripherals requiring the chip select to be de-asserted after each transfer. To facilitate interfacing with such devices, the SPI_CSR can be programmed with the Chip Select Not Active After Transfer (CSNAAT) bit at 1. This allows the chip select lines to be de-asserted systematically during a time “DLYBCS” (the value of the CSNAAT bit is processed only if the CSAAT bit is configured to 0 for the same chip select).

Figure 35-11 shows different peripheral deselection cases and the effect of the CSAAT and CSNAAT bits.

Figure 35-11: Peripheral Deselection



35.7.3.10 Mode Fault Detection

The SPI has the capability to operate in multi-master environment. Consequently, the NPCS0/NSS line must be monitored. If one of the masters on the SPI bus is currently transmitting, the NPCS0/NSS line is low and the SPI must not transmit any data. A mode fault is detected when the SPI is programmed in Master mode and a low level is driven by an external master on the NPCS0/NSS signal. In multi-master environment, NPCS0, MOSI, MISO and SPCK pins must be configured in open drain (through the PIO controller). When a mode fault is detected, the SPI_SR.MODF bit is set until SPI_SR is read and the SPI is automatically disabled until it is re-enabled by setting the SPI_CR.SPIEN bit.

By default, the mode fault detection is enabled. The user can disable it by setting the SPI_MR.MODFDIS bit.

35.7.4 SPI Slave Mode

When operating in Slave mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits until NSS goes active before receiving the serial clock from an external master. When NSS falls, the clock is validated and the data is loaded in the SPI_RDR depending on the BITS field configured in SPI_CSR0. These bits are processed following a phase and a polarity defined respectively by the NCPHA and CPOL bits in SPI_CSR0. Note that the fileds BITS, CPOL and NCPHA of the other chip select registers (SPI_CSR1...SPI_CSR3) have no effect when the SPI is programmed in Slave mode.

The bits are shifted out on the MISO line and sampled on the MOSI line.

Note: For more information on the BITS field, see also the note below the SPI_CSRx register bitmap (Section 35.8.9 “SPI Chip Select Register”).

When all bits are processed, the received data is transferred in the SPI_RDR and the RDRF bit rises. If the SPI_RDR has not been read before new data is received, the Overrun Error Status (OVRES) bit in the SPI_SR is set. As long as this flag is set, data is loaded in the SPI_RDR. The user must read SPI_SR to clear the OVRES bit.

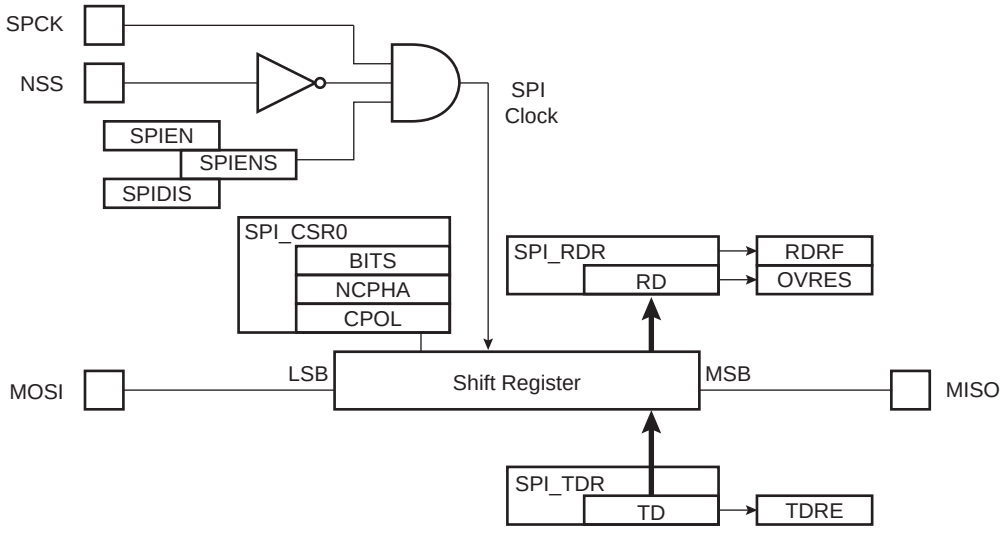
When a transfer starts, the data shifted out is the data present in the Shift register. If no data has been written in the SPI_TDR, the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift register resets to 0.

When a first data is written in the SPI_TDR, it is transferred immediately in the Shift register and the TDRE flag rises. If new data is written, it remains in the SPI_TDR until a transfer occurs, i.e., NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in the SPI_TDR is transferred in the Shift register and the TDRE flag rises. This enables frequent updates of critical variables with single transfers.

Then, new data is loaded in the Shift register from the SPI_TDR. If no character is ready to be transmitted, i.e., no character has been written in the SPI_TDR since the last load from the SPI_TDR to the Shift register, the SPI_TDR is retransmitted. In this case the Underrun Error Status Flag (UNDES) is set in the SPI_SR.

Figure 35-12 shows a block diagram of the SPI when operating in Slave mode.

Figure 35-12: Slave Mode Functional Block Diagram



© 2017 Microchip Technology Inc.

35.7.5 Register Write Protection

To prevent any single software error from corrupting SPI behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the SPI Write Protection Mode Register (SPI_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the SPI Write Protection Status Register (SPI_WPSR) is set and the WPVSRC field indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading SPI_WPSR.

The following registers can be write-protected:

- SPI Mode Register
- SPI Chip Select Register

35.8 Serial Peripheral Interface (SPI) User Interface

In the “Offset” column of Table 35-5, ‘CS_number’ denotes the chip select number.

Table 35-5: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	SPI_CR	Write-only	–
0x04	Mode Register	SPI_MR	Read/Write	0x0
0x08	Receive Data Register	SPI_RDR	Read-only	0x0
0x0C	Transmit Data Register	SPI_TDR	Write-only	–
0x10	Status Register	SPI_SR	Read-only	0x0
0x14	Interrupt Enable Register	SPI_IER	Write-only	–
0x18	Interrupt Disable Register	SPI_IDR	Write-only	–
0x1C	Interrupt Mask Register	SPI_IMR	Read-only	0x0
0x20–0x2C	Reserved	–	–	–
0x30 + (CS_number * 0x04)	Chip Select Register	SPI_CSR	Read/Write	0x0
0x40–0x48	Reserved	–	–	–
0x4C–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	SPI_WPMR	Read/Write	0x0
0xE8	Write Protection Status Register	SPI_WPSR	Read-only	0x0
0xEC–0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

35.8.1 SPI Control Register

Name: SPI_CR

Address:0xF0000000 (0), 0xF0004000 (1)

Access: Write-only

31	30	29	28	27	26	25	24
FIFODIS	FIFOEN	–	–	–	–	–	LASTXFER
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXFCLR	TXFCLR
15	14	13	12	11	10	9	8
–	–	–	REQCLR	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	–	–	–	–	SPIDIS	SPIEN

SPIEN: SPI Enable

0: No effect.

1: Enables the SPI to transfer and receive data.

SPIDIS: SPI Disable

0: No effect.

1: Disables the SPI.

All pins are set in Input mode after completion of the transmission in progress, if any.

If a transfer is in progress when SPIDIS is set, the SPI completes the transmission of the shifter register and does not start any new transfer, even if the SPI_THR is loaded.

Note: If both SPIEN and SPIDIS are equal to one when the SPI_CR is written, the SPI is disabled.

SWRST: SPI Software Reset

0: No effect.

1: Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

The SPI is in Slave mode after software reset.

REQCLR: Request to Clear the Comparison Trigger

0: No effect.

1: Restarts the comparison trigger to enable SPI_RDR loading.

TXFCLR: Transmit FIFO Clear

0: No effect.

1: Clears the Transmit FIFO, Transmit FIFO will become empty.

RXFCLR: Receive FIFO Clear

0: No effect.

1: Clears the Receive FIFO, Receive FIFO will become empty.

LASTXFER: Last Transfer

0: No effect.

1: The current NPCS is de-asserted after the character written in TD has been transferred. When SPI_CSRx.CSAAT is set, the communication with the current serial peripheral can be closed by raising the corresponding NPCS line as soon as TD transfer is completed.

Refer to Section 35.7.3.5 “Peripheral Selection” for more details.

FIFOEN: FIFO Enable

0: No effect.

1: Enables the Transmit and Receive FIFOs

FIFODIS: FIFO Disable

0: No effect.

1: Disables the Transmit and Receive FIFOs

© 2017 Microchip Technology Inc.

35.8.2 SPI Mode Register

Name: SPI_MR

Address:0xF0000004 (0), 0xF0004004 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
DLYBCS							
23	22	21	20	19	18	17	16
–	–	–	–	PCS			
15	14	13	12	11	10	9	8
–	–	–	-	–	–	–	–
7	6	5	4	3	2	1	0
LLB	–	WDRBT	MODFDIS	–	PCSDEC	PS	MSTR

This register can only be written if the WPEN bit is cleared in the SPI Write Protection Mode Register.

MSTR: Master/Slave Mode

0: SPI is in Slave mode

1: SPI is in Master mode

PS: Peripheral Select

0: Fixed Peripheral Select

1: Variable Peripheral Select

PCSDEC: Chip Select Decode

0: The chip select lines are directly connected to a peripheral device.

1: The four NPCS chip select lines are connected to a 4-bit to 16-bit decoder.

When PCSDEC = 1, up to 15 chip select signals can be generated with the four NPCS lines using an external 4-bit to 16-bit decoder. The chip select registers define the characteristics of the 15 chip selects, with the following rules:

SPI_CSR0 defines peripheral chip select signals 0 to 3.

SPI_CSR1 defines peripheral chip select signals 4 to 7.

SPI_CSR2 defines peripheral chip select signals 8 to 11.

SPI_CSR3 defines peripheral chip select signals 12 to 14.

MODFDIS: Mode Fault Detection

0: Mode fault detection enabled

1: Mode fault detection disabled

WDRBT: Wait Data Read Before Transfer

0: No Effect. In Master mode, a transfer can be initiated regardless of the SPI_RDR state.

1: In Master mode, a transfer can start only if the SPI_RDR is empty, i.e., does not contain any unread data. This mode prevents overrun error in reception.

LLB: Local Loopback Enable

0: Local loopback path disabled.

1: Local loopback path enabled.

LLB controls the local loopback on the data shift register for testing in Master mode only (MISO is internally connected on MOSI).

© 2017 Microchip Technology Inc.

PCS: Peripheral Chip Select

This field is only used if fixed peripheral select is active (PS = 0).

If SPI_MR.PCSDEC = 0:

PCS = xxx0NPCS[3:0] = 1110

PCS = xx01NPCS[3:0] = 1101

PCS = x011NPCS[3:0] = 1011

PCS = 0111NPCS[3:0] = 0111

PCS = 1111forbidden (no peripheral is selected)

(x = don't care)

If SPI_MR.PCSDEC = 1:

NPCS[3:0] output signals = PCS.

DLYBCS: Delay Between Chip Selects

This field defines the delay between the inactivation and the activation of NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times.

If DLYBCS is lower than 6, six peripheral clock periods are inserted by default.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Chip Selects} = \frac{\text{DLYBCS}}{f_{\text{peripheral clock}}}$$

© 2017 Microchip Technology Inc.

35.8.3 SPI Receive Data Register

Name: SPI_RDR

Address:0xF0000008 (0), 0xF0004008 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	PCS			
15	14	13	12	11	10	9	8
RD							
7	6	5	4	3	2	1	0
RD							

RD: Receive Data

Data received by the SPI Interface is stored in this register in a right-justified format. Unused bits are read as zero.

PCS: Peripheral Chip Select

In Master mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits are read as zero.

Note: When using Variable Peripheral Select mode (PS = 1 in SPI_MR), it is mandatory to set the SPI_MR.WDRBT bit if the PCS field must be processed in SPI_RDR.

35.8.4 SPI Transmit Data Register

Name: SPI_TDR

Address:0xF000000C (0), 0xF000400C (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER

23	22	21	20	19	18	17	16
–	–	–	–	PCS			

15	14	13	12	11	10	9	8
TD							

7	6	5	4	3	2	1	0
TD							

TD: Transmit Data

Data to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

PCS: Peripheral Chip Select

This field is only used if variable peripheral select is active (PS = 1).

If SPI_MR.PCSDEC = 0:

PCS = xxx0NPCS[3:0] = 1110

PCS = xx01NPCS[3:0] = 1101

PCS = x011NPCS[3:0] = 1011

PCS = 0111NPCS[3:0] = 0111

PCS = 1111forbidden (no peripheral is selected)

(x = don’t care)

If SPI_MR.PCSDEC = 1:

NPCS[3:0] output signals = PCS.

LASTXFER: Last Transfer

0: No effect

1: The current NPCS is de-asserted after the transfer of the character written in TD. When SPI_CSRx.CSAAT is set, the communication with the current serial peripheral can be closed by raising the corresponding NPCS line as soon as TD transfer is completed.

This field is only used if variable peripheral select is active (SPI_MR.PS = 1).

© 2017 Microchip Technology Inc.

35.8.5 SPI Status Register

Name: SPI_SR

Address:0xF0000010 (0), 0xF0004010 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	SPIENS
15	14	13	12	11	10	9	8
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	MODF	TDRE	RDRF

RDRF: Receive Data Register Full (cleared by reading SPI_RDR)

0: No data has been received since the last read of SPI_RDR.

1: Data has been received and the received data has been transferred from the shift register to SPI_RDR since the last read of SPI_RDR.

TDRE: Transmit Data Register Empty (cleared by writing SPI_TDR)

0: Data has been written to SPI_TDR and not yet transferred to the shift register.

1: The last data written in the SPI_TDR has been transferred to the shift register.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to 1.

MODF: Mode Fault Error (cleared on read)

0: No mode fault has been detected since the last read of SPI_SR.

1: A mode fault occurred since the last read of SPI_SR.

OVRES: Overrun Error Status (cleared on read)

0: No overrun has been detected since the last read of SPI_SR.

1: An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the shift register since the last read of the SPI_RDR.

NSSR: NSS Rising (cleared on read)

0: No rising edge detected on NSS pin since the last read of SPI_SR.

1: A rising edge occurred on NSS pin since the last read of SPI_SR.

TXEMPTY: Transmission Registers Empty (cleared by writing SPI_TDR)

0: As soon as data is written in SPI_TDR.

1: SPI_TDR and internal shift register are empty. If a transfer delay has been defined, TXEMPTY is set after the end of this delay.

UNDES: Underrun Error Status (Slave mode only) (cleared on read)

0: No underrun has been detected since the last read of SPI_SR.

1: A transfer starts whereas no data has been loaded in SPI_TDR.

SPIENS: SPI Enable Status

0: SPI is disabled.

1: SPI is enabled.

35.8.6 SPI Interrupt Enable Register

Name: SPI_IER

Address:0xF0000014 (0), 0xF0004014 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	MODF	TDRE	RDRF

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

RDRF: Receive Data Register Full Interrupt Enable

TDRE: SPI Transmit Data Register Empty Interrupt Enable

MODF: Mode Fault Error Interrupt Enable

OVRES: Overrun Error Interrupt Enable

NSSR: NSS Rising Interrupt Enable

TXEMPTY: Transmission Registers Empty Enable

UNDES: Underrun Error Interrupt Enable

© 2017 Microchip Technology Inc.

35.8.7 SPI Interrupt Disable Register

Name: SPI_IDR

Address:0xF0000018 (0), 0xF0004018 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	MODF	TDRE	RDRF

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

RDRF: Receive Data Register Full Interrupt Disable

TDRE: SPI Transmit Data Register Empty Interrupt Disable

MODF: Mode Fault Error Interrupt Disable

OVRES: Overrun Error Interrupt Disable

NSSR: NSS Rising Interrupt Disable

TXEMPTY: Transmission Registers Empty Disable

UNDES: Underrun Error Interrupt Disable

35.8.8 SPI Interrupt Mask Register

Name: SPI_IMR

Address:0xF000001C (0), 0xF000401C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	MODF	TDRE	RDRF

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

RDRF: Receive Data Register Full Interrupt Mask

TDRE: SPI Transmit Data Register Empty Interrupt Mask

MODF: Mode Fault Error Interrupt Mask

OVRES: Overrun Error Interrupt Mask

NSSR: NSS Rising Interrupt Mask

TXEMPTY: Transmission Registers Empty Mask

UNDES: Underrun Error Interrupt Mask

© 2017 Microchip Technology Inc.

35.8.9 SPI Chip Select Register

Name: SPI_CSRx [x=0..3]

Address:0xF0000030 (0), 0xF0004030 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
DLYBCT							
23	22	21	20	19	18	17	16
DLYBS							
15	14	13	12	11	10	9	8
SCBR							
7	6	5	4	3	2	1	0
BITS				CSAAT	CSNAAT	NCPHA	CPOL

This register can only be written if the WPEN bit is cleared in the SPI Write Protection Mode Register.

Note: SPI_CSRx registers must be written even if the user wants to use the default reset values. The BITS field is not updated with the translated value unless the register is written.

CPOL: Clock Polarity

0: The inactive state value of SPCK is logic level zero.

1: The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

NCPHA: Clock Phase

0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)

0: The Peripheral Chip Select Line does not rise between two transfers if the SPI_TDR is reloaded before the end of the first transfer and if the two transfers occur on the same chip select.

1: The Peripheral Chip Select Line rises systematically after each transfer performed on the same slave. It remains inactive after the end of transfer for a minimal duration of:

$$\frac{DLYBCS}{f_{\text{peripheral clock}}}$$ (If field DLYBCS is lower than 6, a minimum of six periods is introduced.)

CSAAT: Chip Select Active After Transfer

0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.

1: The Peripheral Chip Select Line does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.

BITS: Bits Per Transfer

(See the note below the register bitmap.)

The BITS field determines the number of data bits transferred. Reserved values should not be used.

Value	Name	Description
0	8_BIT	8 bits for transfer
1	9_BIT	9 bits for transfer
2	10_BIT	10 bits for transfer
3	11_BIT	11 bits for transfer
4	12_BIT	12 bits for transfer
5	13_BIT	13 bits for transfer
6	14_BIT	14 bits for transfer
7	15_BIT	15 bits for transfer
8	16_BIT	16 bits for transfer
9	–	Reserved
10	–	Reserved
11	–	Reserved
12	–	Reserved
13	–	Reserved
14	–	Reserved
15	–	Reserved

SCBR: Serial Clock Bit Rate

In Master mode, the SPI Interface uses a modulus counter to derive the SPCK bit rate from the peripheral clock. The bit rate is selected by writing a value from1 to 255 in the SCBR field. The following equation determines the SPCK bit rate:

SCBR = $f_{peripheral\ clock}$ / SPCK Bit Rate

Programming the SCBR field to 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

If BRSRCCLK = 1 in SPI_MR, SCBR must be programmed with a value greater than 1.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

Note: If one of the SCBR fields in SPI_CSRx is set to 1, the other SCBR fields in SPI_CSRx must be set to 1 as well, if they are used to process transfers. If they are not used to transfer data, they can be set at any value.

DLYBS: Delay Before SPCK

This field defines the delay from NPCS falling edge (activation) to the first valid SPCK transition.

When DLYBS = 0, the delay is half the SPCK clock period.

Otherwise, the following equation determines the delay:

DLYBS = Delay Before SPCK × $f_{peripheral\ clock}$

DLYBCT: Delay Between Consecutive Transfers

This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

When DLYBCT = 0, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

Otherwise, the following equation determines the delay:

DLYBCT = Delay Between Consecutive Transfers × $f_{peripheral\ clock}$ / 32

© 2017 Microchip Technology Inc.

35.8.10 SPI Write Protection Mode Register

Name:SPI_WPMR

Address:0xF00000E4 (0), 0xF00040E4 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

WPEN: Write Protection Enable

0: Disables the write protection if WPKEY corresponds to 0x535049 (“SPI” in ASCII)

1: Enables the write protection if WPKEY corresponds to 0x535049 (“SPI” in ASCII)

See Section 35.7.5 “Register Write Protection” for the list of registers that can be write-protected.

WPKEY: Write Protection Key

Value	Name	Description
0x535049	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

35.8.11 SPI Write Protection Status Register

Name:SPI_WPSR

Address:0xF00000E8 (0), 0xF00040E8 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
WPVSRC							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

WPVS: Write Protection Violation Status

0: No write protection violation has occurred since the last read of SPI_WPSR.

1: A write protection violation has occurred since the last read of SPI_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

WPVSRC: Write Protection Violation Source

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

© 2017 Microchip Technology Inc.

36. Timer Counter (TC)

36.1 Description

A Timer Counter (TC) module includes three identical TC channels. The number of implemented TC modules is device-specific.

Each TC channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

Each channel has three external clock inputs, five internal clock inputs and two multi-purpose input/output signals which can be configured by the user. Each channel drives an internal interrupt signal which can be programmed to generate processor interrupts.

The TC block has two global registers which act upon all TC channels:

- Block Control Register (TC_BCR)—allows channels to be started simultaneously with the same instruction
- Block Mode Register (TC_BMR)—defines the external clock inputs for each channel, allowing them to be chained

36.2 Embedded Characteristics

- Total number of TC channels: 6
- TC channel size: 32-bit
- Wide range of functions including:
  - Frequency measurement
  - Event counting
  - Interval measurement
  - Pulse generation
  - Delay timing
  - Pulse Width Modulation
  - Up/down capabilities
  - 2-bit gray up/down count for stepper motor
- Each channel is user-configurable and contains:
  - Three external clock inputs
  - Five Internal clock inputs
  - Two multi-purpose input/output signals acting as trigger event
- Internal interrupt signal
- Read of the Capture registers by the DMAC
- Register Write Protection

36.3 Block Diagram

Table 36-1: Timer Counter Clock Assignment

Name	Definition
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5(1)	SLCK

Note 1: When SLCK is selected for Peripheral Clock (CSS = 0 in PMC Master Clock Register), SLCK input is equivalent to Peripheral Clock.

Figure 36-1: Timer Counter Block Diagram

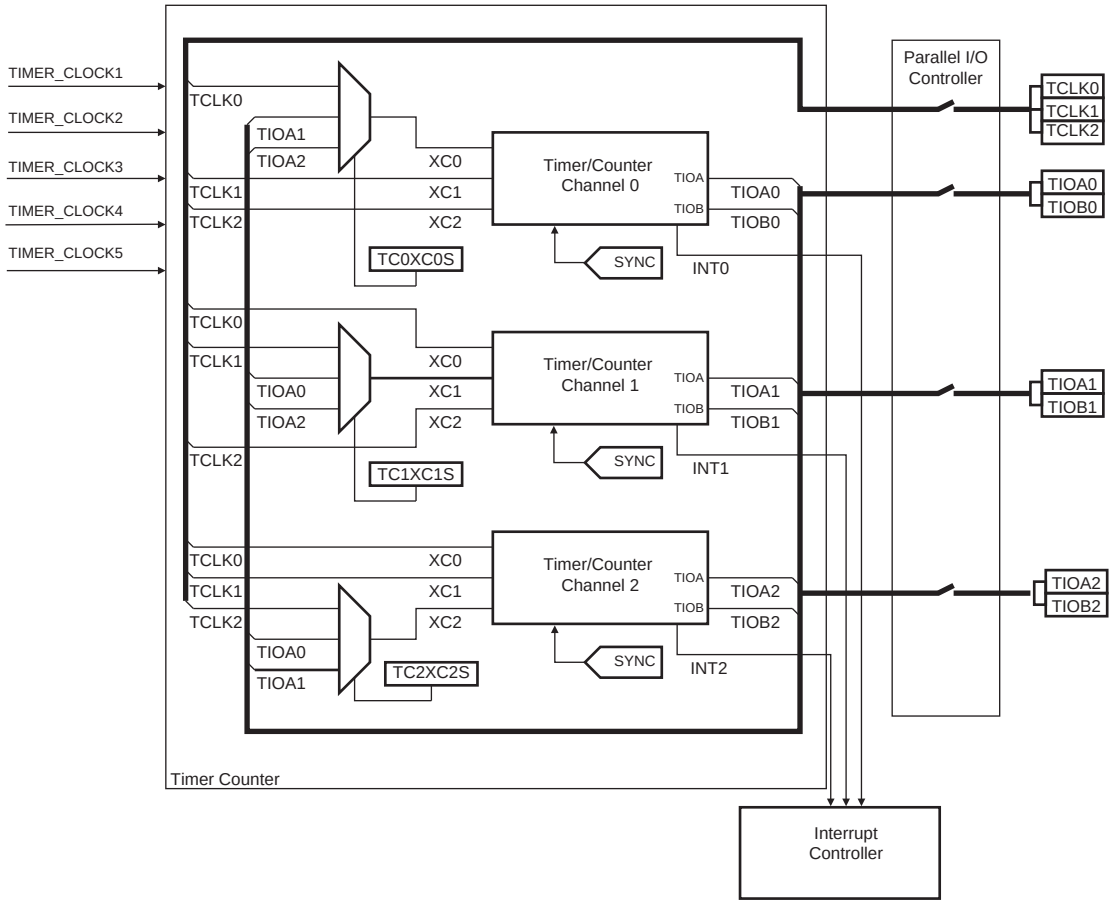


Table 36-2: Signal Description

Block/Channel	Signal Name	Description
Channel Signal	XC0, XC1, XC2	External Clock Inputs
	TIOA	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Output
	TIOB	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Input/Output
	INT	Interrupt Signal Output (internal signal)
	SYNC	Synchronization Input Signal (from configuration register)

© 2017 Microchip Technology Inc.

36.4 Pin List

Table 36-3: Pin List

Pin Name	Description	Type
TCLK0–TCLK2	External Clock Input	Input
TIOA0–TIOA2	I/O Line A	I/O
TIOB0–TIOB2	I/O Line B	I/O

36.5 Product Dependencies

36.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the TC pins to their peripheral functions.

Table 36-4: I/O Lines

Instance	Signal	I/O Line	Peripheral
TC0	TCLK0	PA24	A
TC0	TCLK1	PA25	A
TC0	TCLK2	PA26	A
TC0	TIOA0	PA21	A
TC0	TIOA1	PA22	A
TC0	TIOA2	PA23	A
TC0	TIOB0	PA27	A
TC0	TIOB1	PA28	A
TC0	TIOB2	PA29	A
TC1	TCLK3	PC4	C
TC1	TCLK4	PC7	C
TC1	TCLK5	PC14	C
TC1	TIOA3	PC2	C
TC1	TIOA4	PC5	C
TC1	TIOA5	PC12	C
TC1	TIOB3	PC3	C
TC1	TIOB4	PC6	C
TC1	TIOB5	PC13	C

36.5.2 Power Management

The TC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the Timer Counter clock.

36.5.3 Interrupt Sources

The TC has an interrupt line connected to the interrupt controller. Handling the TC interrupt requires programming the interrupt controller before configuring the TC.

Table 36-5: Peripheral IDs

Instance	ID
TC0	17
TC1	17

36.6 Functional Description

36.6.1 Description

All channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Table 36-6 “Register Mapping”.

36.6.2 32-bit Counter

Each 32-bit channel is organized around a 32-bit counter. The value of the counter is incremented at each positive edge of the selected clock. When the counter has reached the value 2^{32}-1 and passes to zero, an overflow occurs and the COVFS bit in the TC Status Register (TC_SR) is set.

The current value of the counter is accessible in real time by reading the TC Counter Value Register (TC_CV). The counter can be reset by a trigger. In this case, the counter value passes to zero on the next valid edge of the selected clock.

36.6.3 Clock Selection

At block level, input clock signals of each channel can either be connected to the external inputs TCLK0, TCLK1 or TCLK2, or be connected to the internal I/O signals TIOA0, TIOA1 or TIOA2 for chaining by programming the TC Block Mode Register (TC_BMR). See Figure 36-2.

Each channel can independently select an internal or external clock source for its counter:

- External clock signals(1): XC0, XC1 or XC2
- Internal clock signals: MCK/2, MCK/8, MCK/32, MCK/128, SLCK

This selection is made by the TCCLKS bits in the TC Channel Mode Register (TC_CMR).

The selected clock can be inverted with the CLKI bit in the TC_CMR. This allows counting on the opposite edges of the clock.

The burst function allows the clock to be validated when an external signal is high. The BURST parameter in the TC_CMR defines this signal (none, XC0, XC1, XC2). See Figure 36-3.

Note 1: In all cases, if an external clock is used, the duration of each of its levels must be longer than the peripheral clock period. The external clock frequency must be at least 2.5 times lower than the peripheral clock.

© 2017 Microchip Technology Inc.

Figure 36-2: Clock Chaining Selection

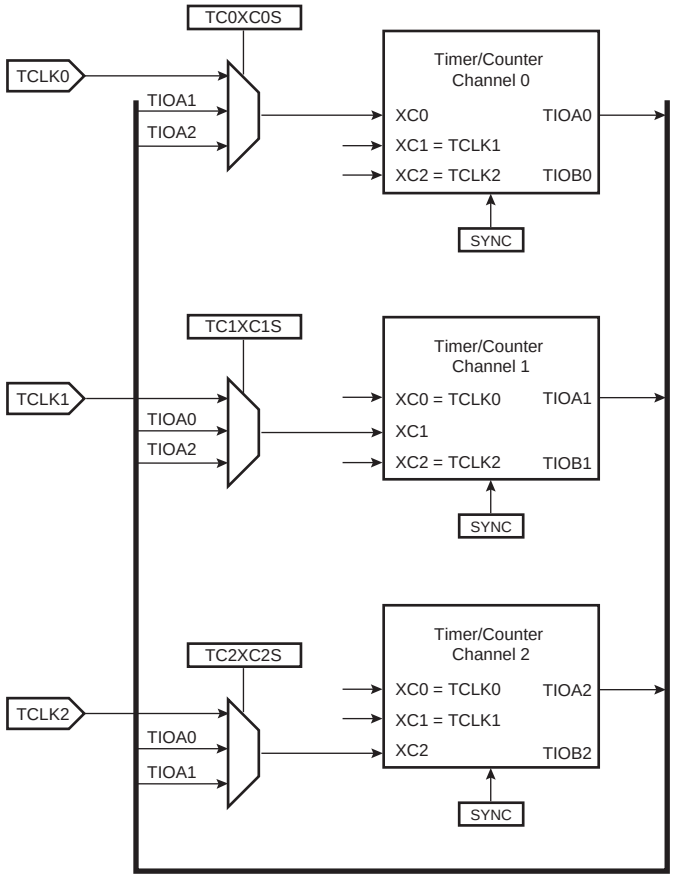
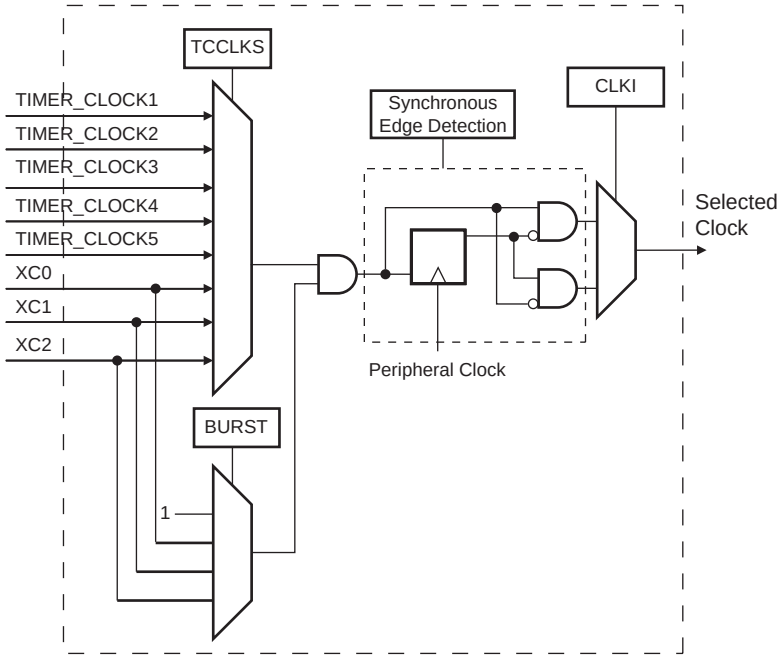


Figure 36-3: Clock Selection

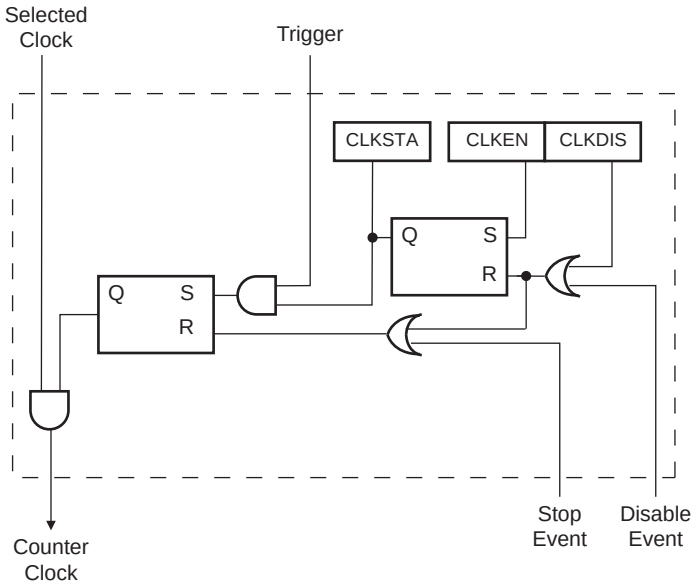


36.6.4 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 36-4.

- The clock can be enabled or disabled by the user with the CLKEN and the CLKDIS commands in the TC Channel Control Register (TC_CCR). In Capture mode it can be disabled by an RB load event if LDBDIS is set to 1 in the TC_CMR. In Waveform mode, it can be disabled by an RC Compare event if CPCDIS is set to 1 in TC_CMR. When disabled, the start or the stop actions have no effect: only a CLKEN command in the TC_CCR can re-enable the clock. When the clock is enabled, the CLKSTA bit is set in the TC_SR.
- The clock can also be started or stopped: a trigger (software, synchro, external or compare) always starts the clock. The clock can be stopped by an RB load event in Capture mode (LDBSTOP = 1 in TC_CMR) or an RC compare event in Waveform mode (CPCSTOP = 1 in TC_CMR). The start and the stop commands are effective only if the clock is enabled.

Figure 36-4: Clock Control



36.6.5 Operating Modes

Each channel can operate independently in two different modes:

- Capture mode provides measurement on signals.
- Waveform mode provides wave generation.

The TC operating mode is programmed with the WAVE bit in the TC_CMR.

In Capture mode, TIOA and TIOB are configured as inputs.

In Waveform mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

36.6.6 Trigger

A trigger resets the counter and starts the counter clock. Three types of triggers are common to both modes, and a fourth external trigger is available to each mode.

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

The following triggers are common to both modes:

- Software Trigger: Each channel has a software trigger, available by setting SWTRG in TC_CCR.
- SYNC: Each channel has a synchronization signal SYNC. When asserted, this signal has the same effect as a software trigger. The SYNC signals of all channels are asserted simultaneously by writing TC_BCR (Block Control) with SYNC set.
- Compare RC Trigger: RC is implemented in each channel and can provide a trigger when the counter value matches the RC value if CPCTRG is set in the TC_CMR.

© 2017 Microchip Technology Inc.

The channel can also be configured to have an external trigger. In Capture mode, the external trigger signal can be selected between TIOA and TIOB. In Waveform mode, an external event can be programmed on one of the following signals: TIOB, XC0, XC1 or XC2. This external event can then be programmed to perform a trigger by setting bit ENETRG in the TC_CMR.

If an external trigger is used, the duration of the pulses must be longer than the peripheral clock period in order to be detected.

36.6.7 Capture Mode

Capture mode is entered by clearing the WAVE bit in the TC_CMR.

Capture mode allows the TC channel to perform measurements such as pulse timing, frequency, period, duty cycle and phase on TIOA and TIOB signals which are considered as inputs.

Figure 36-6 shows the configuration of the TC channel when programmed in Capture mode.

36.6.8 Capture Registers A and B

Registers A and B (RA and RB) are used as capture registers. They can be loaded with the counter value when a programmable event occurs on the signal TIOA.

The LDRA field in the TC_CMR defines the TIOA selected edge for the loading of register A, and the LDRB field defines the TIOA selected edge for the loading of Register B.

RA is loaded only if it has not been loaded since the last trigger or if RB has been loaded since the last loading of RA.

RB is loaded only if RA has been loaded since the last trigger or the last loading of RB.

Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS bit) in the TC_SR. In this case, the old value is overwritten.

When DMA is used, the RAB register address must be configured as source address of the transfer. The RAB register provides the next unread value from Register A and Register B. It may be read by the DMA after a request has been triggered upon loading Register A or Register B.

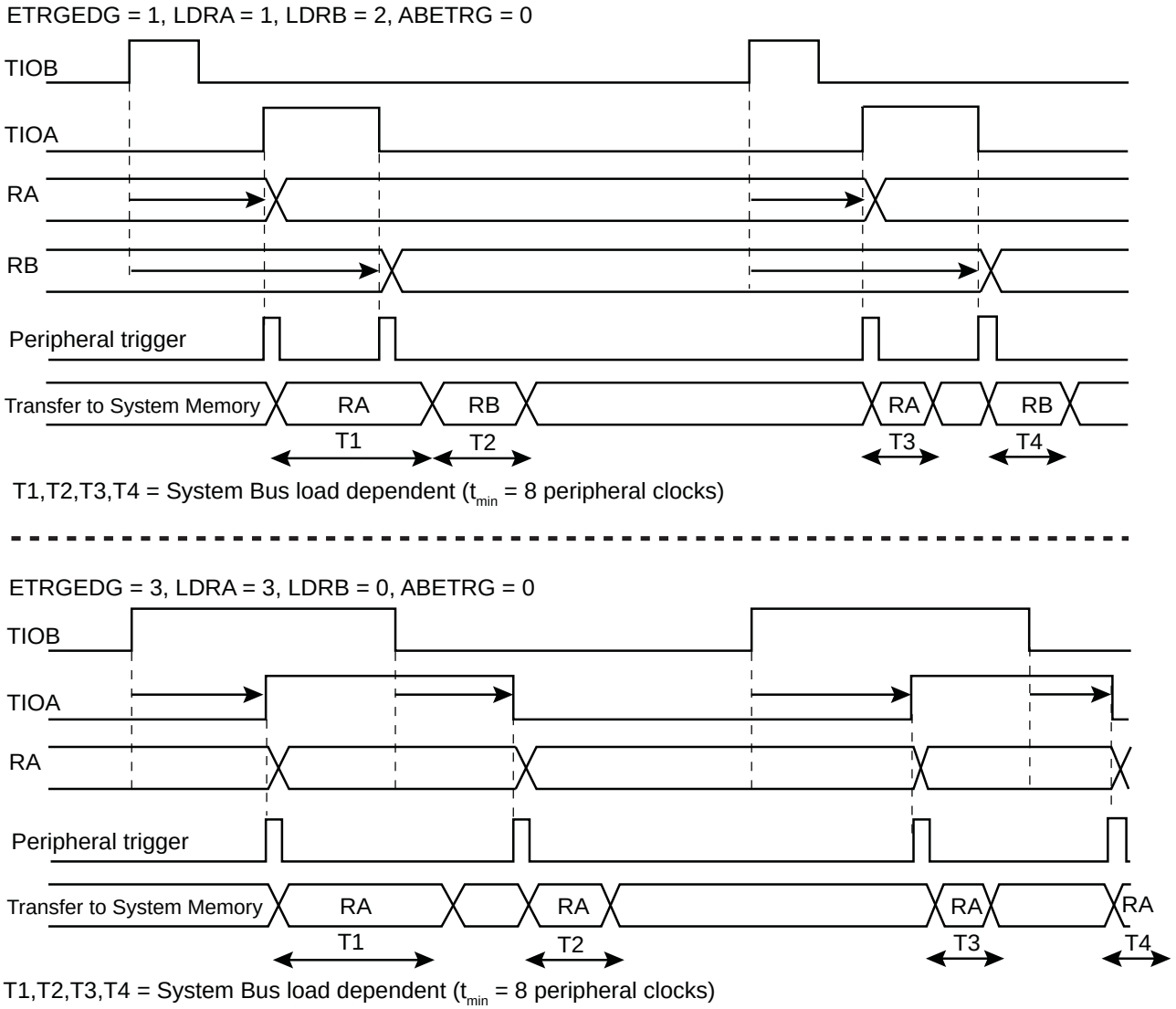
36.6.9 Transfer with DMAC

The DMAC can only perform access from timer to system memory.

Figure 36-5 illustrates how TC_RA and TC_RB can be loaded in the system memory without CPU intervention.

© 2017 Microchip Technology Inc.

Figure 36-5: Example of Transfer with DMAC

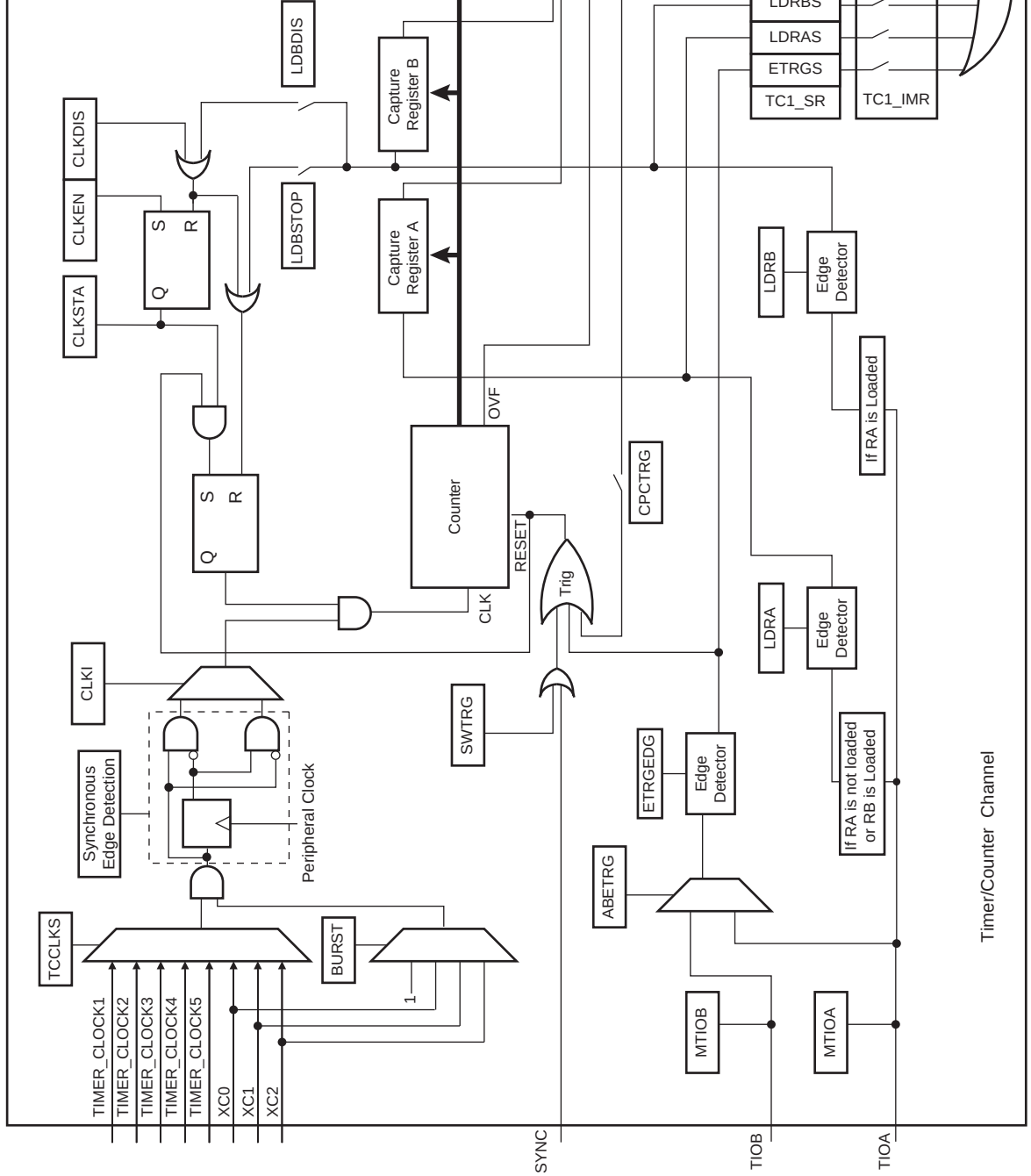


36.6.10 Trigger Conditions

In addition to the SYNC signal, the software trigger and the RC compare trigger, an external trigger can be defined.

The ABETRG bit in the TC_CMR selects TIOA or TIOB input signal as an external trigger . The External Trigger Edge Selection parameter (ETRGEDG field in TC_CMR) defines the edge (rising, falling, or both) detected to generate an external trigger. If ETRGEDG = 0 (none), the external trigger is disabled.

© 2017 Microchip Technology Inc.


LDRBS
LDRAS
ETRGS
TC1_SR
TC1_IMR
LDBDIS
Capture Register B
CLKDIS
CLKEN
CLKSTA
LDBSTOP
Capture Register A
LDRB
Edge Detector
If RA is Loaded
OVF
Counter
CPCTRG
RESET
Trig
CLK
LDRA
Edge Detector
CLKI
SWTRG
ETRGEDG
Edge Detector
Synchronous Edge Detection
Peripheral Clock
If RA is not loaded or RB is Loaded
ABETRG
TCCLKS
BURST
1
MTIOB
MTIOA
TIMER_CLOCK1
TIMER_CLOCK2
TIMER_CLOCK3
TIMER_CLOCK4
TIMER_CLOCK5
XC0
XC1
XC2
SYNC
TIOB
TIOA
Timer/Counter Channel

36.6.11 Waveform Mode

Waveform mode is entered by setting the TC_CMRx.WAVE bit.

In Waveform mode, the TC channel generates one or two PWM signals with the same frequency and independently programmable duty cycles, or generates different types of one-shot or repetitive pulses.

In this mode, TIOA is configured as an output and TIOB is defined as an output if it is not used as an external event (EEVT parameter in TC_CMR).

Figure 36-7 shows the configuration of the TC channel when programmed in Waveform operating mode.

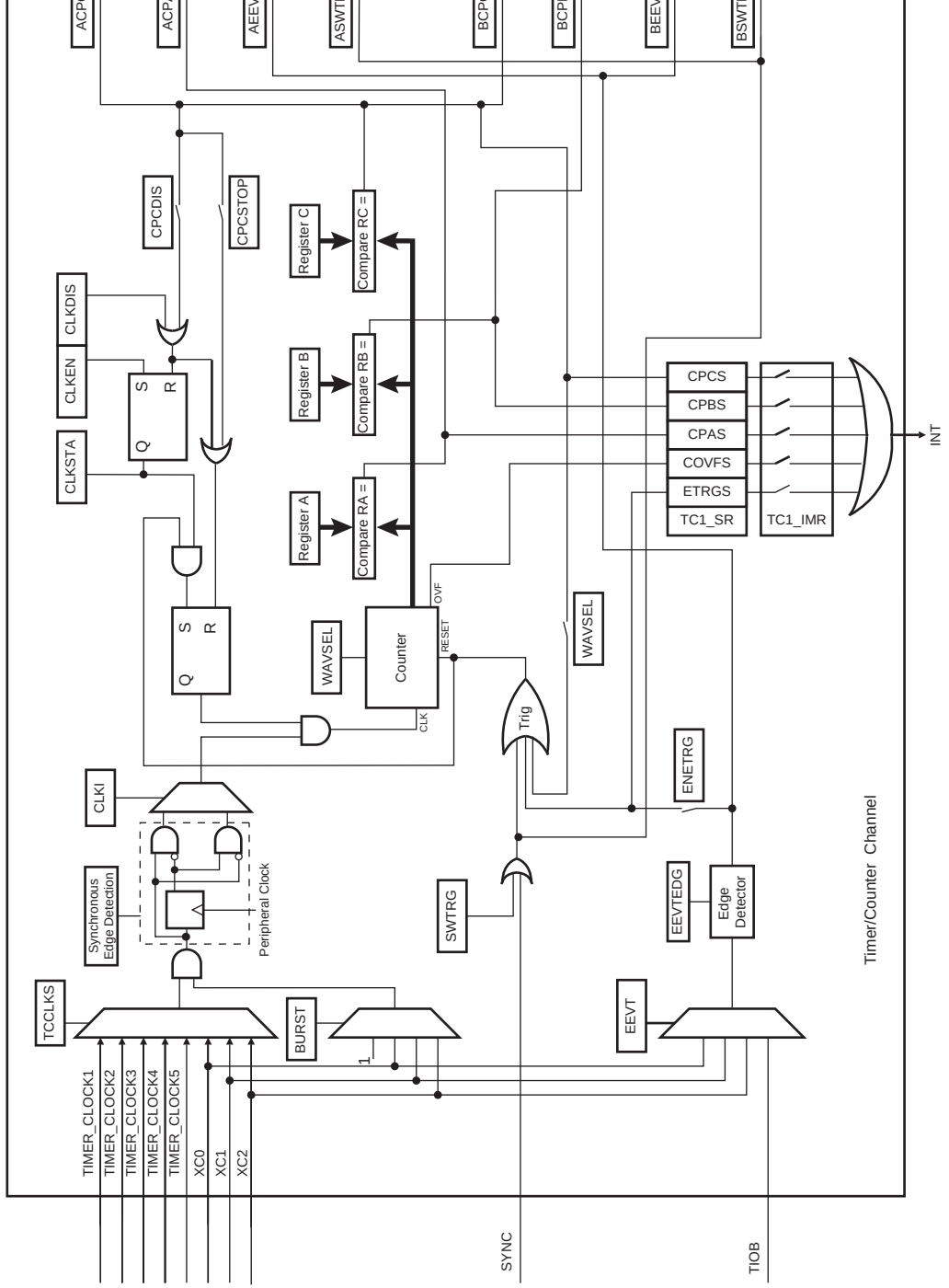
36.6.12 Waveform Selection

Depending on the WAVSEL parameter in TC_CMR, the behavior of TC_CV varies.

With any selection, TC_RA, TC_RB and TC_RC can all be used as compare registers.

RA Compare is used to control the TIOA output, RB Compare is used to control the TIOB output (if correctly configured) and RC Compare is used to control TIOA and/or TIOB outputs.

© 2017 Microchip Technology Inc.


ACPC
ACPA
AEEV
ASWTRG
BCPB
BCPA
BEEV
BSWTRG
CPCDIS
CPCSTOP
Register C
Compare RC =
CLKDIS
CLKEN
CLKSTA
S
R
Q
Register B
Compare RB =
Register A
Compare RA =
CPCS
CPBS
CPAS
COVFS
ETRGS
TC1_SR
TC1_IMR
INT
WAVSEL
Counter
OVF
RESET
CLK
WAVSEL
Trig
ENETRG
CLKI
Synchronous Edge Detection
Peripheral Clock
SWTRG
EEVTEDG
Edge Detector
TCCLKS
BURST
1
EEVT
Timer/Counter Channel
TIMER_CLOCK1
TIMER_CLOCK2
TIMER_CLOCK3
TIMER_CLOCK4
TIMER_CLOCK5
XC0
XC1
XC2
SYNC
TIOB

36.6.12.1 WAVSEL = 00

When WAVSEL = 00, the value of TC_CV is incremented from 0 to 2^{32}-1. Once 2^{32}-1 has been reached, the value of TC_CV is reset. Incrementation of TC_CV starts again and the cycle continues. See Figure 36-8.

An external event trigger or a software trigger can reset the value of TC_CV. It is important to note that the trigger may occur at any time. See Figure 36-9.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 36-8: WAVSEL = 00 without Trigger

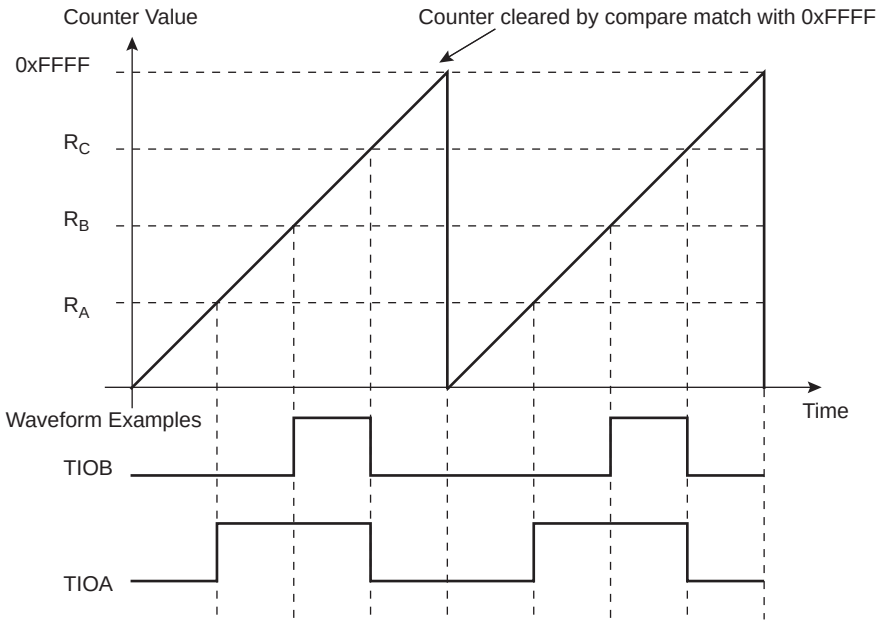
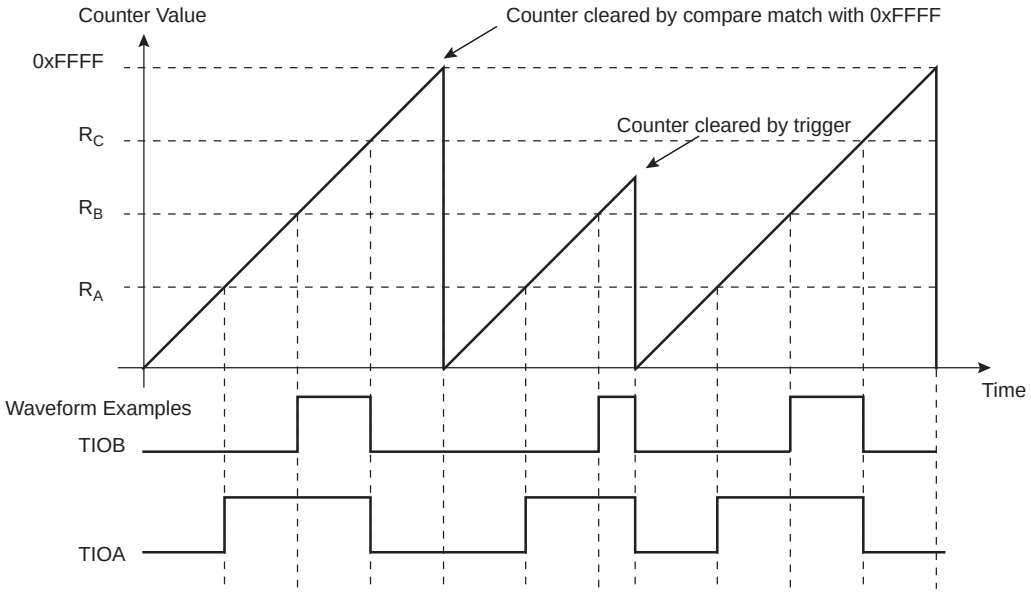


Figure 36-9: WAVSEL = 00 with Trigger



© 2017 Microchip Technology Inc.

36.6.12.2 WAVSEL = 10

When WAVSEL = 10, the value of TC_CV is incremented from 0 to the value of RC, then automatically reset on a RC Compare. Once the value of TC_CV has been reset, it is then incremented and so on. See Figure 36-10.

It is important to note that TC_CV can be reset at any time by an external event or a software trigger if both are programmed correctly. See Figure 36-11.

In addition, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 36-10: WAVSEL = 10 without Trigger

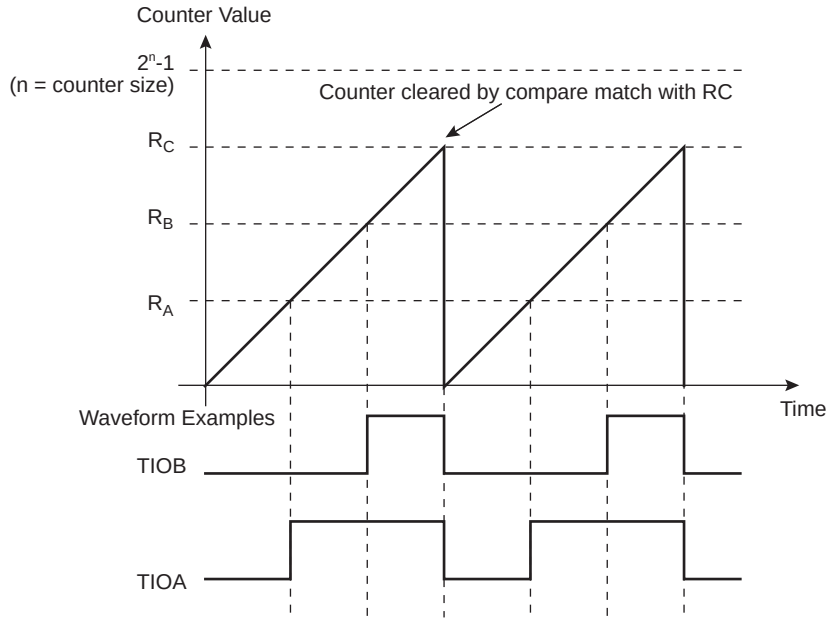
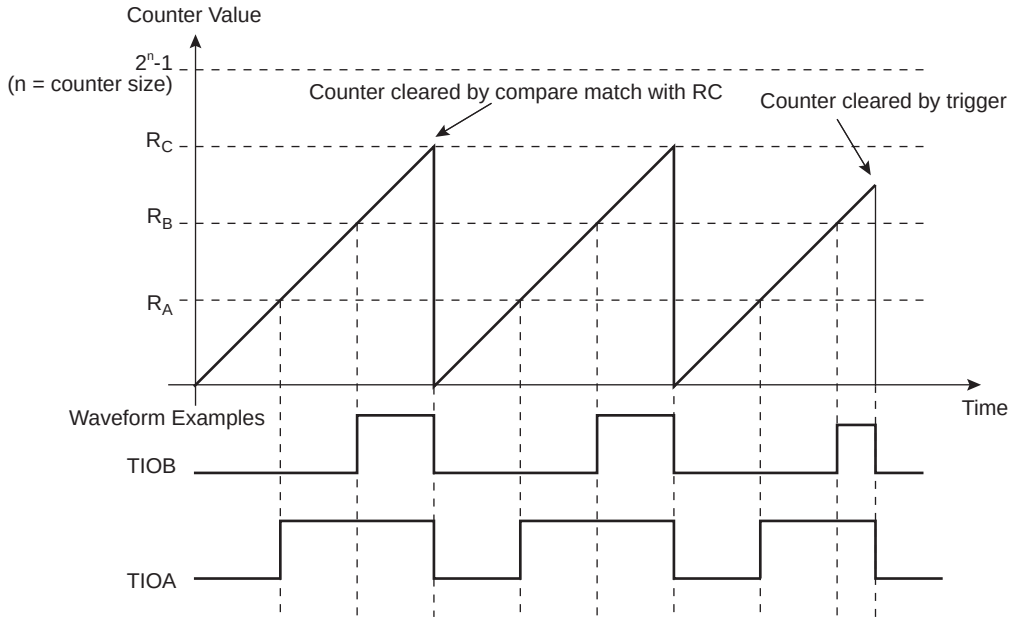


Figure 36-11: WAVSEL = 10 with Trigger



36.6.12.3 WAVSEL = 01

When WAVSEL = 01, the value of TC_CV is incremented from 0 to 2^{32}-1 . Once 2^{32}-1 is reached, the value of TC_CV is decremented to 0, then re-incremented to 2^{32}-1 and so on. See Figure 36-12.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 36-13.

RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 36-12: WAVSEL = 01 without Trigger

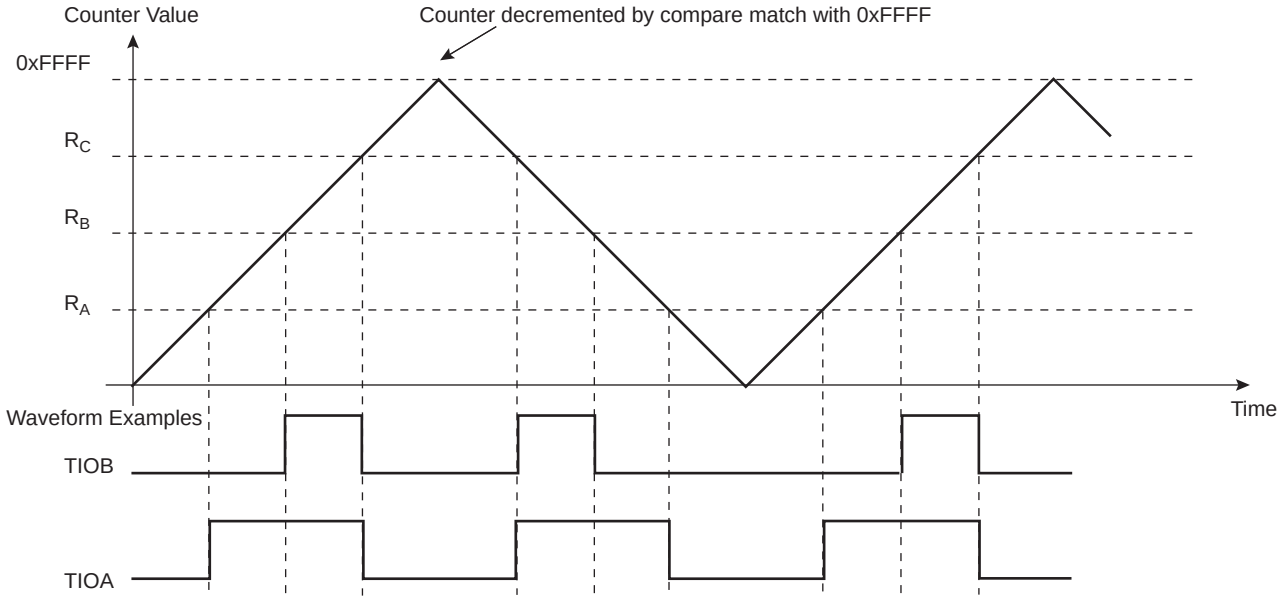
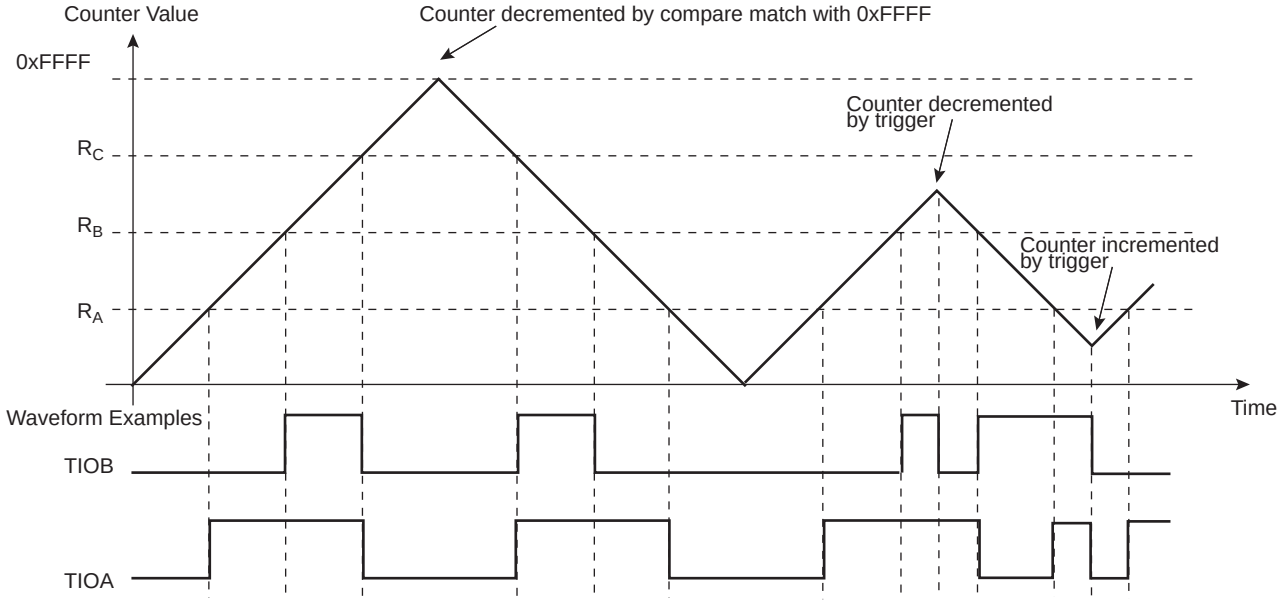


Figure 36-13: WAVSEL = 01 with Trigger



© 2017 Microchip Technology Inc.

36.6.12.4 WAVSEL = 11

When WAVSEL = 11, the value of TC_CV is incremented from 0 to RC. Once RC is reached, the value of TC_CV is decremented to 0, then re-incremented to RC and so on. See Figure 36-14.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 36-15.

RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 36-14: WAVSEL = 11 without Trigger

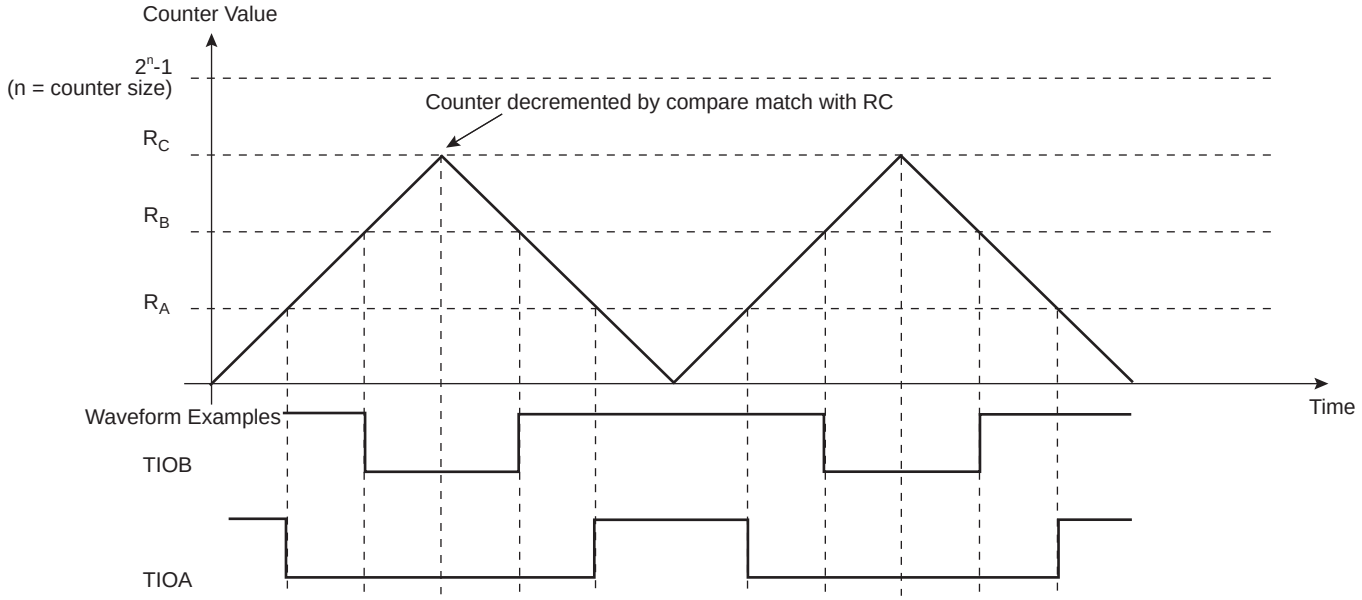
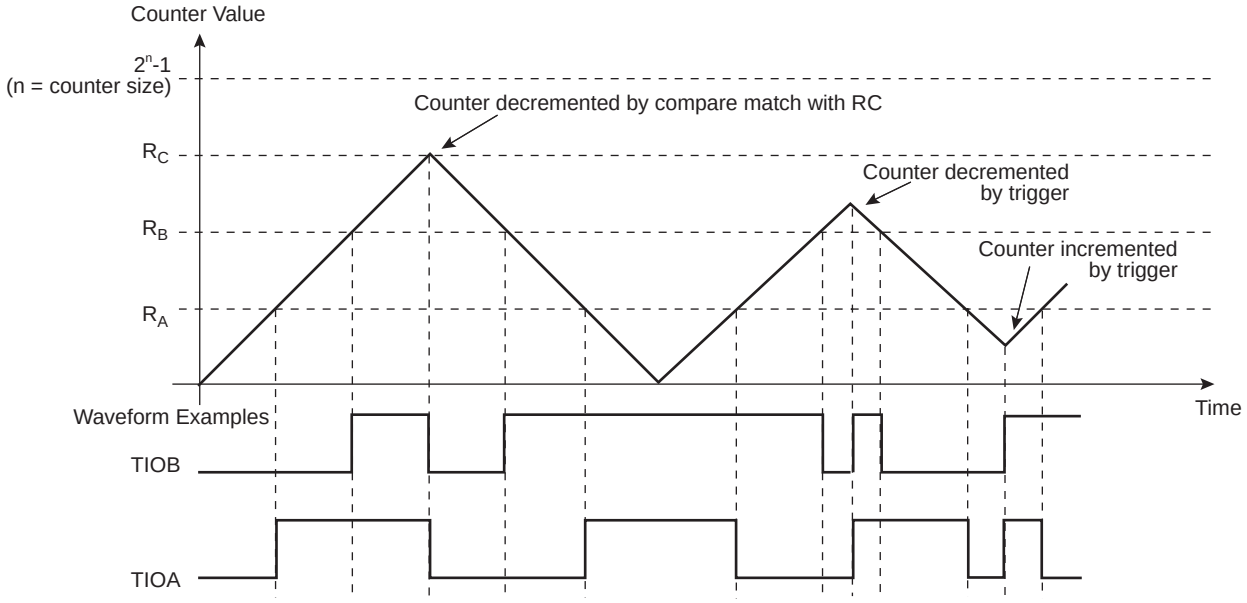


Figure 36-15: WAVSEL = 11 with Trigger



36.6.13 External Event/Trigger Conditions

An external event can be programmed to be detected on one of the clock sources (XC0, XC1, XC2) or TIOB. The external event selected can then be used as a trigger.

The EEVT parameter in TC_CMR selects the external trigger. The EEVTEDG parameter defines the trigger edge for each of the possible external triggers (rising, falling or both). If EEVTEDG is cleared (none), no external event is defined.

If TIOB is defined as an external event signal (EEVT = 0), TIOB is no longer used as an output and the compare register B is not used to generate waveforms and subsequently no IRQs. In this case the TC channel can only generate a waveform on TIOA.

When an external event is defined, it can be used as a trigger by setting bit ENETRG in the TC_CMR.

As in Capture mode, the SYNC signal and the software trigger are also available as triggers. RC Compare can also be used as a trigger depending on the parameter WAVSEL.

36.6.14 Output Controller

The output controller defines the output level changes on TIOA and TIOB following an event. TIOB control is used only if TIOB is defined as output (not as an external event).

The following events control TIOA and TIOB: software trigger, external event and RC compare. RA compare controls TIOA and RB compare controls TIOB. Each of these events can be programmed to set, clear or toggle the output as defined in the corresponding parameter in TC_CMR.

36.6.15 2-bit Gray Up/Down Counter for Stepper Motor

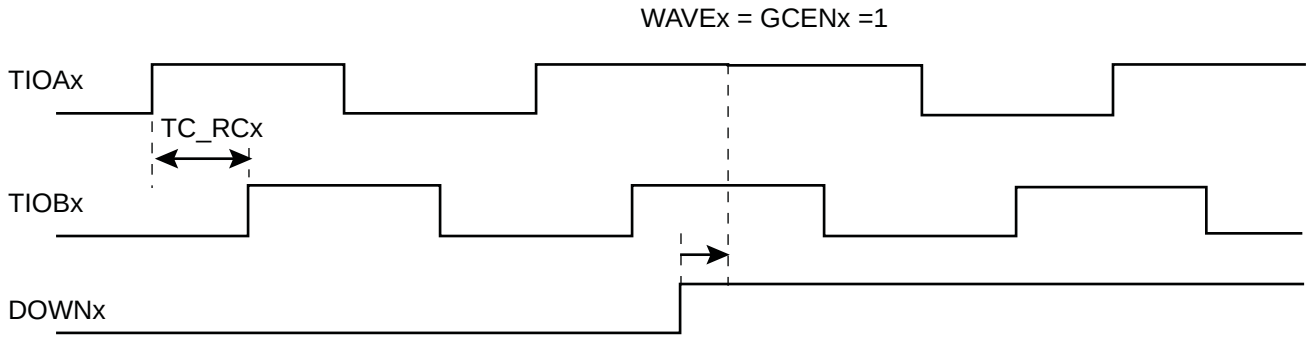
Each channel can be independently configured to generate a 2-bit gray count waveform on corresponding TIOA, TIOB outputs by means of the GCEN bit in TC_SMMRx.

Up or Down count can be defined by writing bit DOWN in TC_SMMRx.

It is mandatory to configure the channel in Waveform mode in the TC_CMR.

The period of the counters can be programmed in TC_RCx.

Figure 36-16: 2-bit Gray Up/Down Counter



36.6.16 Register Write Protection

To prevent any single software error from corrupting TC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the TC Write Protection Mode Register (TC_WPMR).

The Timer Counter clock of the first channel must be enabled to access TC_WPMR.

The following registers can be write-protected:

- TC Block Mode Register
- TC Channel Mode Register: Capture Mode
- TC Channel Mode Register: Waveform Mode
- TC Stepper Motor Mode Register
- TC Register A
- TC Register B
- TC Register C

© 2017 Microchip Technology Inc.

36.7 Timer Counter (TC) User Interface

Table 36-6: Register Mapping

Offset[1]	Register	Name	Access	Reset
0x00 + channel * 0x40 + 0x00	Channel Control Register	TC_CCR	Write-only	–
0x00 + channel * 0x40 + 0x04	Channel Mode Register	TC_CMR	Read/Write	0
0x00 + channel * 0x40 + 0x08	Stepper Motor Mode Register	TC_SMMR	Read/Write	0
0x00 + channel * 0x40 + 0x0C	Register AB	TC_RAB	Read-only	0
0x00 + channel * 0x40 + 0x10	Counter Value	TC_CV	Read-only	0
0x00 + channel * 0x40 + 0x14	Register A	TC_RA	Read/Write[2]	0
0x00 + channel * 0x40 + 0x18	Register B	TC_RB	Read/Write[2]	0
0x00 + channel * 0x40 + 0x1C	Register C	TC_RC	Read/Write	0
0x00 + channel * 0x40 + 0x20	Status Register	TC_SR	Read-only	0
0x00 + channel * 0x40 + 0x24	Interrupt Enable Register	TC_IER	Write-only	–
0x00 + channel * 0x40 + 0x28	Interrupt Disable Register	TC_IDR	Write-only	–
0x00 + channel * 0x40 + 0x2C	Interrupt Mask Register	TC_IMR	Read-only	0
0xC0	Block Control Register	TC_BCR	Write-only	–
0xC4	Block Mode Register	TC_BMR	Read/Write	0
0xC8–0xD4	Reserved	–	–	–
0xD8	Reserved	–	–	–
0xE4	Write Protection Mode Register	TC_WPMR	Read/Write	0
0xE8–0xFC	Reserved	–	–	–

Note 1: Channel index ranges from 0 to 2.

2: Read-only if TC_CMRx.WAVE = 0

36.7.1 TC Channel Control Register

Name:TC_CCRx [x=0..2]

Address:0xF8008000 (0)[0], 0xF8008040 (0)[1], 0xF8008080 (0)[2], 0xF800C000 (1)[0], 0xF800C040 (1)[1], 0xF800C080 (1)[2]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	SWTRG	CLKDIS	CLKEN

CLKEN: Counter Clock Enable Command

0: No effect.

1: Enables the clock if CLKDIS is not 1.

CLKDIS: Counter Clock Disable Command

0: No effect.

1: Disables the clock.

SWTRG: Software Trigger Command

0: No effect.

1: A software trigger is performed: the counter is reset and the clock is started.

© 2017 Microchip Technology Inc.

36.7.2 TC Channel Mode Register: Capture Mode

Name:TC_CMRx [x=0..2] (CAPTURE_MODE)

Address:0xF8008004 (0)[0], 0xF8008044 (0)[1], 0xF8008084 (0)[2], 0xF800C004 (1)[0], 0xF800C044 (1)[1], 0xF800C084 (1)[2]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	LDRB		LDRA	

15	14	13	12	11	10	9	8
WAVE	CPCTRG	–	–	–	ABETRG	ETRGEDG	

7	6	5	4	3	2	1	0
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

TCCLKS: Clock Selection

Value	Name	Description
0	TIMER_CLOCK1	Clock selected: internal MCK/2 clock signal (from PMC)
1	TIMER_CLOCK2	Clock selected: internal MCK/8 clock signal (from PMC)
2	TIMER_CLOCK3	Clock selected: internal MCK/32 clock signal (from PMC)
3	TIMER_CLOCK4	Clock selected: internal MCK/128 clock signal (from PMC)
4	TIMER_CLOCK5	Clock selected: internal SLCK clock signal (from PMC)
5	XC0	Clock selected: XC0
6	XC1	Clock selected: XC1
7	XC2	Clock selected: XC2

CLKI: Clock Invert

0: Counter is incremented on rising edge of the clock.

1: Counter is incremented on falling edge of the clock.

BURST: Burst Signal Selection

Value	Name	Description
0	NONE	The clock is not gated by an external signal.
1	XC0	XC0 is ANDed with the selected clock.
2	XC1	XC1 is ANDed with the selected clock.
3	XC2	XC2 is ANDed with the selected clock.

LDBSTOP: Counter Clock Stopped with RB Loading

0: Counter clock is not stopped when RB loading occurs.

1: Counter clock is stopped when RB loading occurs.

LDBDIS: Counter Clock Disable with RB Loading

0: Counter clock is not disabled when RB loading occurs.

1: Counter clock is disabled when RB loading occurs.

ETRGEDG: External Trigger Edge Selection

Value	Name	Description
0	NONE	The clock is not gated by an external signal.
1	RISING	Rising edge
2	FALLING	Falling edge
3	EDGE	Each edge

ABETRG: TIOA or TIOB External Trigger Selection

0: TIOB is used as an external trigger.

1: TIOA is used as an external trigger.

CPCTRG: RC Compare Trigger Enable

0: RC Compare has no effect on the counter and its clock.

1: RC Compare resets the counter and starts the counter clock.

WAVE: Waveform Mode

0: Capture mode is enabled.

1: Capture mode is disabled (Waveform mode is enabled).

LDRA: RA Loading Edge Selection

Value	Name	Description
0	NONE	None
1	RISING	Rising edge of TIOA
2	FALLING	Falling edge of TIOA
3	EDGE	Each edge of TIOA

LDRB: RB Loading Edge Selection

Value	Name	Description
0	NONE	None
1	RISING	Rising edge of TIOA
2	FALLING	Falling edge of TIOA
3	EDGE	Each edge of TIOA

© 2017 Microchip Technology Inc.

36.7.3 TC Channel Mode Register: Waveform Mode

Name:TC_CMRx [x=0..2] (WAVEFORM_MODE)

Address:0xF8008004 (0)[0], 0xF8008044 (0)[1], 0xF8008084 (0)[2], 0xF800C004 (1)[0], 0xF800C044 (1)[1], 0xF800C084 (1)[2]

Access: Read/Write

31	30	29	28	27	26	25	24
BSWTRG		BEEVT		BCPC		BCPB	

23	22	21	20	19	18	17	16
ASWTRG		AEEVT		ACPC		ACPA	

15	14	13	12	11	10	9	8
WAVE	WAVSEL		ENETRG	EEVT		EEVTEDG	

7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST		CLKI	TCCLKS		

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

TCCLKS: Clock Selection

Value	Name	Description
0	TIMER_CLOCK1	Clock selected: internal MCK/2 clock signal (from PMC)
1	TIMER_CLOCK2	Clock selected: internal MCK/8 clock signal (from PMC)
2	TIMER_CLOCK3	Clock selected: internal MCK/32 clock signal (from PMC)
3	TIMER_CLOCK4	Clock selected: internal MCK/128 clock signal (from PMC)
4	TIMER_CLOCK5	Clock selected: internal SLCK clock signal (from PMC)
5	XC0	Clock selected: XC0
6	XC1	Clock selected: XC1
7	XC2	Clock selected: XC2

CLKI: Clock Invert

0: Counter is incremented on rising edge of the clock.

1: Counter is incremented on falling edge of the clock.

BURST: Burst Signal Selection

Value	Name	Description
0	NONE	The clock is not gated by an external signal.
1	XC0	XC0 is ANDed with the selected clock.
2	XC1	XC1 is ANDed with the selected clock.
3	XC2	XC2 is ANDed with the selected clock.

CPCSTOP: Counter Clock Stopped with RC Compare

0: Counter clock is not stopped when counter reaches RC.

1: Counter clock is stopped when counter reaches RC.

© 2017 Microchip Technology Inc.

CPCDIS: Counter Clock Disable with RC Compare

0: Counter clock is not disabled when counter reaches RC.

1: Counter clock is disabled when counter reaches RC.

EEVTEDG: External Event Edge Selection

Value	Name	Description
0	NONE	None
1	RISING	Rising edge
2	FALLING	Falling edge
3	EDGE	Each edge

EEVT: External Event Selection

Signal selected as external event.

Value	Name	Description	TIOB Direction
0	TIOB	TIOB(1)	Input
1	XC0	XC0	Output
2	XC1	XC1	Output
3	XC2	XC2	Output

Note 1: If TIOB is chosen as the external event signal, it is configured as an input and no longer generates waveforms and subsequently no IRQs.

ENETRG: External Event Trigger Enable

0: The external event has no effect on the counter and its clock.

1: The external event resets the counter and starts the counter clock.

Note: Whatever the value programmed in ENETRG, the selected external event only controls the TIOA output and TIOB if not used as input (trigger event input or other input used).

WAVSEL: Waveform Selection

Value	Name	Description
0	UP	UP mode without automatic trigger on RC Compare
1	UPDOWN	UPDOWN mode without automatic trigger on RC Compare
2	UP_RC	UP mode with automatic trigger on RC Compare
3	UPDOWN_RC	UPDOWN mode with automatic trigger on RC Compare

WAVE: Waveform Mode

0: Waveform mode is disabled (Capture mode is enabled).

1: Waveform mode is enabled.

ACPA: RA Compare Effect on TIOA

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

© 2017 Microchip Technology Inc.

ACPC: RC Compare Effect on TIOA

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

AEEVT: External Event Effect on TIOA

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

ASWTRG: Software Trigger Effect on TIOA

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

BCPB: RB Compare Effect on TIOB

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

BCPC: RC Compare Effect on TIOB

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

BEEVT: External Event Effect on TIOB

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

© 2017 Microchip Technology Inc.

BSWTRG: Software Trigger Effect on TIOB

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

© 2017 Microchip Technology Inc.

36.7.4 TC Stepper Motor Mode Register

Name:TC_SMMRx [x=0..2]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	DOWN	GCEN

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

GCEN: Gray Count Enable

0: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by internal counter of channel x.

1: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by a 2-bit gray counter.

DOWN: Down Count

0: Up counter.

1: Down counter.

36.7.5 TC Register AB

Name:TC_RABx [x=0..2]

Address:0xF800800C (0)[0], 0xF800804C (0)[1], 0xF800808C (0)[2], 0xF800C00C (1)[0], 0xF800C04C (1)[1], 0xF800C08C (1)[2]

Access: Read-only

31	30	29	28	27	26	25	24
RAB							

23	22	21	20	19	18	17	16
RAB							

15	14	13	12	11	10	9	8
RAB							

7	6	5	4	3	2	1	0
RAB							

RAB: Register A or Register B

RAB contains the next unread capture Register A or Register B value in real time. It is usually read by the DMA after a request due to a valid load edge on TIOA.

When DMA is used, the RAB register address must be configured as source address of the transfer.

© 2017 Microchip Technology Inc.

36.7.6 TC Counter Value Register

Name:TC_CVx [x=0..2]

Address:0xF8008010 (0)[0], 0xF8008050 (0)[1], 0xF8008090 (0)[2], 0xF800C010 (1)[0], 0xF800C050 (1)[1], 0xF800C090 (1)[2]

Access: Read-only

31	30	29	28	27	26	25	24
CV							

23	22	21	20	19	18	17	16
CV							

15	14	13	12	11	10	9	8
CV							

7	6	5	4	3	2	1	0
CV							

CV: Counter Value

CV contains the counter value in real time.

36.7.7 TC Register A

Name:TC_RAx [x=0..2]

Address:0xF8008014 (0)[0], 0xF8008054 (0)[1], 0xF8008094 (0)[2], 0xF800C014 (1)[0], 0xF800C054 (1)[1], 0xF800C094 (1)[2]

Access: Read-only if TC_CMRx.WAVE = 0, Read/Write if TC_CMRx.WAVE = 1

31	30	29	28	27	26	25	24
RA							

23	22	21	20	19	18	17	16
RA							

15	14	13	12	11	10	9	8
RA							

7	6	5	4	3	2	1	0
RA							

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

RA: Register A

RA contains the Register A value in real time.

© 2017 Microchip Technology Inc.

36.7.8 TC Register B

Name:TC_RBx [x=0..2]

Address:0xF8008018 (0)[0], 0xF8008058 (0)[1], 0xF8008098 (0)[2], 0xF800C018 (1)[0], 0xF800C058 (1)[1], 0xF800C098 (1)[2]

Access: Read-only if TC_CMRx.WAVE = 0, Read/Write if TC_CMRx.WAVE = 1

31	30	29	28	27	26	25	24
RB							

23	22	21	20	19	18	17	16
RB							

15	14	13	12	11	10	9	8
RB							

7	6	5	4	3	2	1	0
RB							

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

RB: Register B

RB contains the Register B value in real time.

36.7.9 TC Register C

Name:TC_RCx [x=0..2]

Address:0xF800801C (0)[0], 0xF800805C (0)[1], 0xF800809C (0)[2], 0xF800C01C (1)[0], 0xF800C05C (1)[1], 0xF800C09C (1)[2]

Access: Read/Write

31	30	29	28	27	26	25	24
RC							

23	22	21	20	19	18	17	16
RC							

15	14	13	12	11	10	9	8
RC							

7	6	5	4	3	2	1	0
RC							

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

RC: Register C

RC contains the Register C value in real time.

© 2017 Microchip Technology Inc.

36.7.10 TC Status Register

Name:TC_SRx [x=0..2]

Address:0xF8008020 (0)[0], 0xF8008060 (0)[1], 0xF80080A0 (0)[2], 0xF800C020 (1)[0], 0xF800C060 (1)[1], 0xF800C0A0 (1)[2]

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	MTIOB	MTIOA	CLKSTA
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

COVFS: Counter Overflow Status (cleared on read)

0: No counter overflow has occurred since the last read of the Status Register.

1: A counter overflow has occurred since the last read of the Status Register.

LOVRS: Load Overrun Status (cleared on read)

0: Load overrun has not occurred since the last read of the Status Register or TC_CMRx.WAVE = 1.

1: RA or RB have been loaded at least twice without any read of the corresponding register since the last read of the Status Register, if TC_CMRx.WAVE = 0.

CPAS: RA Compare Status (cleared on read)

0: RA Compare has not occurred since the last read of the Status Register or TC_CMRx.WAVE = 0.

1: RA Compare has occurred since the last read of the Status Register, if TC_CMRx.WAVE = 1.

CPBS: RB Compare Status (cleared on read)

0: RB Compare has not occurred since the last read of the Status Register or TC_CMRx.WAVE = 0.

1: RB Compare has occurred since the last read of the Status Register, if TC_CMRx.WAVE = 1.

CPCS: RC Compare Status (cleared on read)

0: RC Compare has not occurred since the last read of the Status Register.

1: RC Compare has occurred since the last read of the Status Register.

LDRAS: RA Loading Status (cleared on read)

0: RA Load has not occurred since the last read of the Status Register or TC_CMRx.WAVE = 1.

1: RA Load has occurred since the last read of the Status Register, if TC_CMRx.WAVE = 0.

LDRBS: RB Loading Status (cleared on read)

0: RB Load has not occurred since the last read of the Status Register or TC_CMRx.WAVE = 1.

1: RB Load has occurred since the last read of the Status Register, if TC_CMRx.WAVE = 0.

ETRGS: External Trigger Status (cleared on read)

0: External trigger has not occurred since the last read of the Status Register.

1: External trigger has occurred since the last read of the Status Register.

CLKSTA: Clock Enabling Status

0: Clock is disabled.

1: Clock is enabled.

MTIOA: TIOA Mirror

0: TIOA is low. If TC_CMRx.WAVE = 0, this means that TIOA pin is low. If TC_CMRx.WAVE = 1, this means that TIOA is driven low.

1: TIOA is high. If TC_CMRx.WAVE = 0, this means that TIOA pin is high. If TC_CMRx.WAVE = 1, this means that TIOA is driven high.

MTIOB: TIOB Mirror

0: TIOB is low. If TC_CMRx.WAVE = 0, this means that TIOB pin is low. If TC_CMRx.WAVE = 1, this means that TIOB is driven low.

1: TIOB is high. If TC_CMRx.WAVE = 0, this means that TIOB pin is high. If TC_CMRx.WAVE = 1, this means that TIOB is driven high.

© 2017 Microchip Technology Inc.

36.7.11 TC Interrupt Enable Register

Name:TC_IERx [x=0..2]

Address:0xF8008024 (0)[0], 0xF8008064 (0)[1], 0xF80080A4 (0)[2], 0xF800C024 (1)[0], 0xF800C064 (1)[1], 0xF800C0A4 (1)[2]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

COVFS: Counter Overflow

0: No effect.

1: Enables the Counter Overflow Interrupt.

LOVRS: Load Overrun

0: No effect.

1: Enables the Load Overrun Interrupt.

CPAS: RA Compare

0: No effect.

1: Enables the RA Compare Interrupt.

CPBS: RB Compare

0: No effect.

1: Enables the RB Compare Interrupt.

CPCS: RC Compare

0: No effect.

1: Enables the RC Compare Interrupt.

LDRAS: RA Loading

0: No effect.

1: Enables the RA Load Interrupt.

LDRBS: RB Loading

0: No effect.

1: Enables the RB Load Interrupt.

ETRGS: External Trigger

0: No effect.

1: Enables the External Trigger Interrupt.

36.7.12 TC Interrupt Disable Register

Name:TC_IDRx [x=0..2]

Address:0xF8008028 (0)[0], 0xF8008068 (0)[1], 0xF80080A8 (0)[2], 0xF800C028 (1)[0], 0xF800C068 (1)[1], 0xF800C0A8 (1)[2]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

COVFS: Counter Overflow

0: No effect.

1: Disables the Counter Overflow Interrupt.

LOVRS: Load Overrun

0: No effect.

1: Disables the Load Overrun Interrupt (if TC_CMRx.WAVE = 0).

CPAS: RA Compare

0: No effect.

1: Disables the RA Compare Interrupt (if TC_CMRx.WAVE = 1).

CPBS: RB Compare

0: No effect.

1: Disables the RB Compare Interrupt (if TC_CMRx.WAVE = 1).

CPCS: RC Compare

0: No effect.

1: Disables the RC Compare Interrupt.

LDRAS: RA Loading

0: No effect.

1: Disables the RA Load Interrupt (if TC_CMRx.WAVE = 0).

LDRBS: RB Loading

0: No effect.

1: Disables the RB Load Interrupt (if TC_CMRx.WAVE = 0).

ETRGS: External Trigger

0: No effect.

1: Disables the External Trigger Interrupt.

© 2017 Microchip Technology Inc.

36.7.13 TC Interrupt Mask Register

Name:TC_IMRx [x=0..2]

Address:0xF800802C (0)[0], 0xF800806C (0)[1], 0xF80080AC (0)[2], 0xF800C02C (1)[0], 0xF800C06C (1)[1], 0xF800C0AC (1)[2]

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

COVFS: Counter Overflow

0: The Counter Overflow Interrupt is disabled.

1: The Counter Overflow Interrupt is enabled.

LOVRS: Load Overrun

0: The Load Overrun Interrupt is disabled.

1: The Load Overrun Interrupt is enabled.

CPAS: RA Compare

0: The RA Compare Interrupt is disabled.

1: The RA Compare Interrupt is enabled.

CPBS: RB Compare

0: The RB Compare Interrupt is disabled.

1: The RB Compare Interrupt is enabled.

CPCS: RC Compare

0: The RC Compare Interrupt is disabled.

1: The RC Compare Interrupt is enabled.

LDRAS: RA Loading

0: The Load RA Interrupt is disabled.

1: The Load RA Interrupt is enabled.

LDRBS: RB Loading

0: The Load RB Interrupt is disabled.

1: The Load RB Interrupt is enabled.

ETRGS: External Trigger

0: The External Trigger Interrupt is disabled.

1: The External Trigger Interrupt is enabled.

36.7.14 TC Block Control Register

Name:TC_BCR

Address:0xF80080C0 (0), 0xF800C0C0 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SYNC

SYNC: Synchro Command

0: No effect.

1: Asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.

© 2017 Microchip Technology Inc.

36.7.15 TC Block Mode Register

Name:TC_BMR

Address:0xF80080C4 (0), 0xF800C0C4 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TC2XC2S		TC1XC1S		TC0XC0S	

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

TC0XC0S: External Clock Signal 0 Selection

Value	Name	Description
0	TCLK0	Signal connected to XC0: TCLK0
1	–	Reserved
2	TIOA1	Signal connected to XC0: TIOA1
3	TIOA2	Signal connected to XC0: TIOA2

TC1XC1S: External Clock Signal 1 Selection

Value	Name	Description
0	TCLK1	Signal connected to XC1: TCLK1
1	–	Reserved
2	TIOA0	Signal connected to XC1: TIOA0
3	TIOA2	Signal connected to XC1: TIOA2

TC2XC2S: External Clock Signal 2 Selection

Value	Name	Description
0	TCLK2	Signal connected to XC2: TCLK2
1	–	Reserved
2	TIOA0	Signal connected to XC2: TIOA0
3	TIOA1	Signal connected to XC2: TIOA1

© 2017 Microchip Technology Inc.

36.7.16 TC Write Protection Mode Register

Name:TC_WPMR

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

WPEN: Write Protection Enable

0: Disables the write protection if WPKEY corresponds to 0x54494D (“TIM” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x54494D (“TIM” in ASCII).

The Timer Counter clock of the first channel must be enabled to access this register.

See Section 36.6.16 “Register Write Protection” for a list of registers that can be write-protected and Timer Counter clock conditions.

WPKEY: Write Protection Key

Value	Name	Description
0x54494D	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

© 2017 Microchip Technology Inc.

37. Pulse Width Modulation Controller (PWM)

37.1 Description

The PWM macrocell controls several channels independently. Each channel controls one square output waveform. Characteristics of the output waveform such as period, duty-cycle and polarity are configurable through the user interface. Each channel selects and uses one of the clocks provided by the clock generator. The clock generator provides several clocks resulting from the division of the PWM macrocell master clock.

All PWM macrocell accesses are made through APB mapped registers.

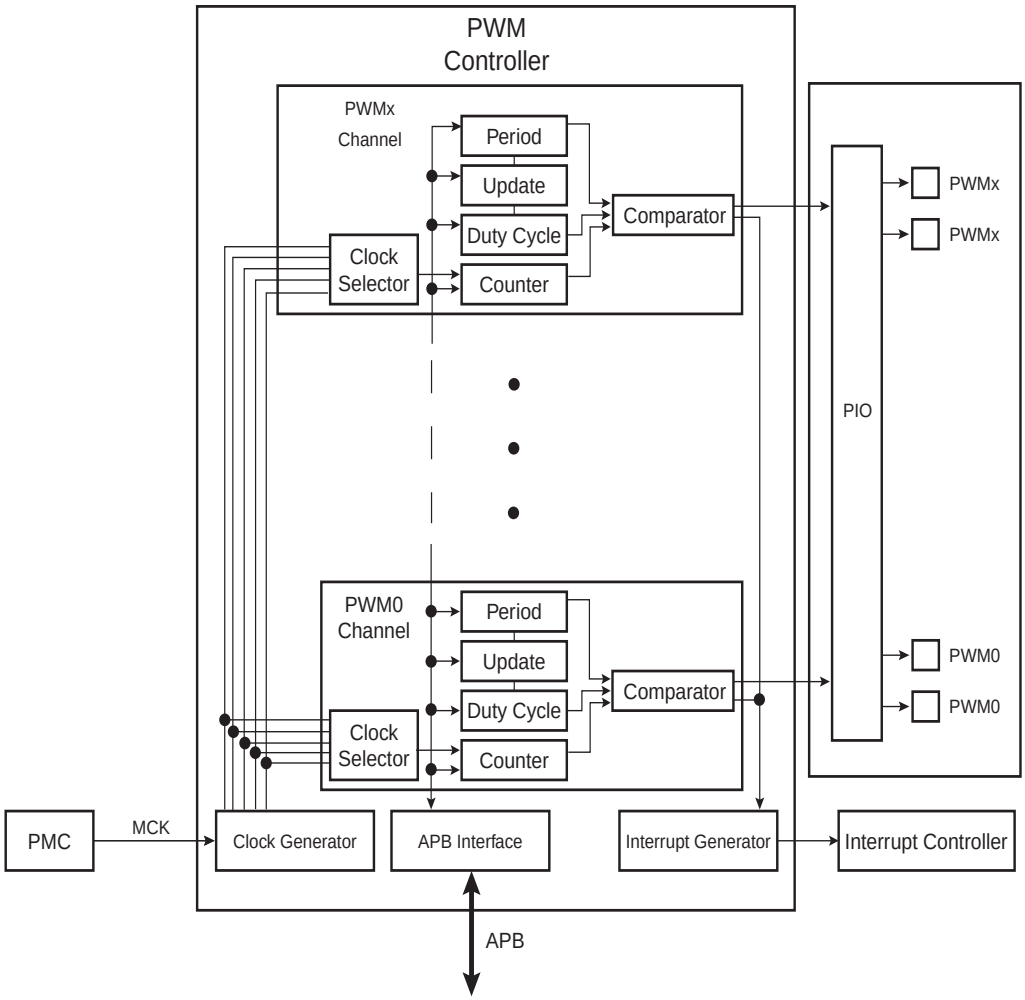
Channels can be synchronized, to generate non overlapped waveforms. All channels integrate a double buffering system in order to prevent an unexpected output waveform while modifying the period or the duty-cycle.

37.2 Embedded characteristics

- 4 Channels
- One 32-bit Counter Per Channel
- Common Clock Generator Providing Thirteen Different Clocks
  - A Modulo n Counter Providing Eleven Clocks
  - Two Independent Linear Dividers Working on Modulo n Counter Outputs
- Independent Channels
  - Independent Enable Disable Command for Each Channel
  - Independent Clock Selection for Each Channel
  - Independent Period and Duty Cycle for Each Channel
  - Double Buffering of Period or Duty Cycle for Each Channel
  - Programmable Selection of The Output Waveform Polarity for Each Channel
  - Programmable Center or Left Aligned Output Waveform for Each Channel Block Diagram

37.3 Block Diagram

Figure 37-1: Pulse Width Modulation Controller Block Diagram



37.4 I/O Lines Description

Each channel outputs one waveform on one external I/O line.

Table 37-1: I/O Line Description

Name	Description	Type
PWMx	PWM Waveform Output for channel x	Output

© 2017 Microchip Technology Inc.

37.5 Product Dependencies

37.5.1 I/O Lines

The pins used for interfacing the PWM may be multiplexed with PIO lines. The programmer must first program the PIO controller to assign the desired PWM pins to their peripheral function. If I/O lines of the PWM are not used by the application, they can be used for other purposes by the PIO controller.

All of the PWM outputs may or may not be enabled. If an application requires only four channels, then only four PIO lines will be assigned to PWM outputs.

Table 37-2: I/O Lines

Instance	Signal	I/O Line	Peripheral
PWM	PWM0	PB11	B
PWM	PWM0	PC10	C
PWM	PWM0	PC18	C
PWM	PWM1	PB12	B
PWM	PWM1	PC11	C
PWM	PWM1	PC19	C
PWM	PWM2	PB13	B
PWM	PWM2	PC20	C
PWM	PWM3	PB14	B
PWM	PWM3	PC21	C

37.5.2 Power Management

The PWM is not continuously clocked. The programmer must first enable the PWM clock in the Power Management Controller (PMC) before using the PWM. However, if the application does not require PWM operations, the PWM clock can be stopped when not needed and be restarted later. In this case, the PWM will resume its operations where it left off.

All the PWM registers except PWM_CDTY and PWM_CPRD can be read without the PWM peripheral clock enabled. All the registers can be written without the peripheral clock enabled.

37.5.3 Interrupt Sources

The PWM interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the PWM interrupt requires the Interrupt Controller to be programmed first. Note that it is not recommended to use the PWM interrupt line in edge sensitive mode.

Table 37-3: Peripheral IDs

Instance	ID
PWM	18

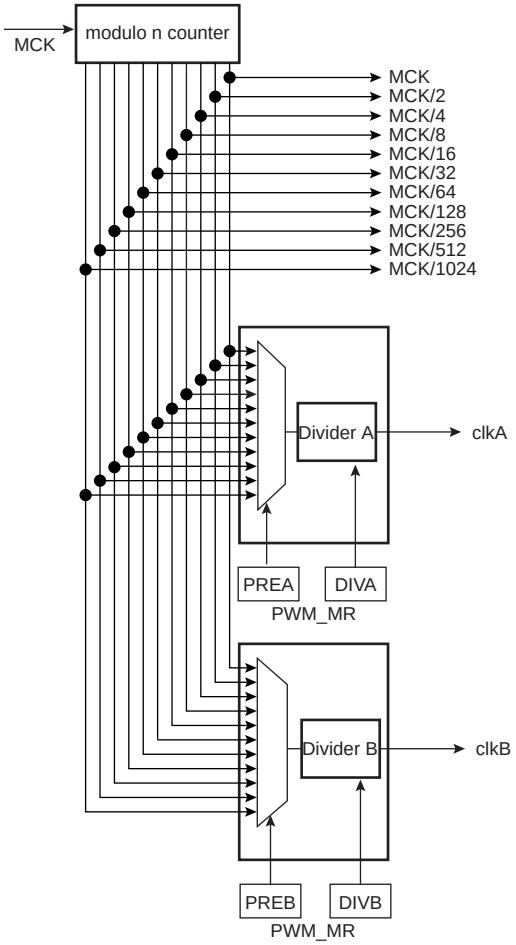
37.6 Functional Description

The PWM macrocell is primarily composed of a clock generator module and 4 channels.

- Clocked by the system clock, MCK, the clock generator module provides 13 clocks.
- Each channel can independently choose one of the clock generator outputs.
- Each channel generates an output waveform with attributes that can be defined independently for each channel through the user interface registers.

37.6.1 PWM Clock Generator

Figure 37-2: Functional View of the Clock Generator Block Diagram



Caution: Before using the PWM macrocell, the programmer must first enable the PWM clock in the Power Management Controller (PMC).

The PWM macrocell master clock, MCK, is divided in the clock generator module to provide different clocks available for all channels. Each channel can independently select one of the divided clocks.
The clock generator is divided in three blocks:

- a modulo n counter which provides 11 clocks: F_{MCK}, $F_{MCK}/2$, $F_{MCK}/4$, $F_{MCK}/8$, $F_{MCK}/16$, $F_{MCK}/32$, $F_{MCK}/64$, $F_{MCK}/128$, $F_{MCK}/256$, $F_{MCK}/512$, $F_{MCK}/1024$
- two linear dividers (1, 1/2, 1/3,... 1/255) that provide two separate clocks: clkA and clkB

Each linear divider can independently divide one of the clocks of the modulo n counter. The selection of the clock to be divided is made according to the PREA (PREB) field of the PWM Mode register (PWM_MR). The resulting clock clkA (clkB) is the clock selected divided by DIVA (DIVB) field value in the PWM Mode register (PWM_MR).

After a reset of the PWM controller, DIVA (DIVB) and PREA (PREB) in the PWM Mode register are set to 0. This implies that after reset clkA (clkB) are turned off.

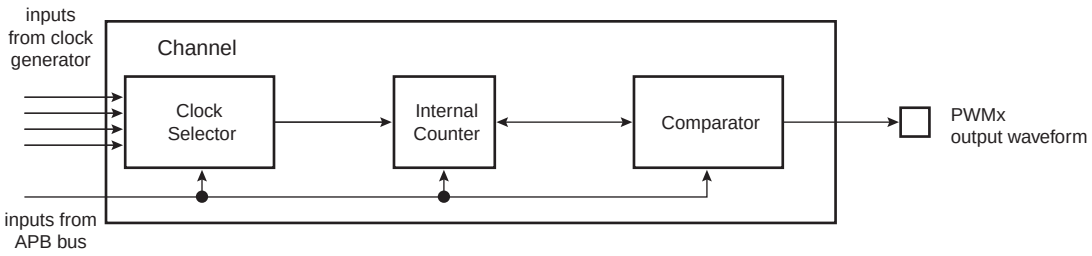
© 2017 Microchip Technology Inc.

At reset, all clocks provided by the modulo n counter are turned off except clock “clk”. This situation is also true when the PWM master clock is turned off through the Power Management Controller.

37.6.2 PWM Channel

37.6.2.1 Block Diagram

Figure 37-3: Functional View of the Channel Block Diagram



Each of the 4 channels is composed of three blocks:

- A clock selector which selects one of the clocks provided by the clock generator described in Section 37.6.1, PWM Clock Generator.
- An internal counter clocked by the output of the clock selector. This internal counter is incremented or decremented according to the channel configuration and comparators events. The size of the internal counter is 32 bits.
- A comparator used to generate events according to the internal counter value. It also computes the PWMx output waveform according to the configuration.

37.6.2.2 Waveform Properties

The different properties of output waveforms are:

- the ***internal clock selection***. The internal channel counter is clocked by one of the clocks provided by the clock generator described in the previous section. This channel parameter is defined in the CPRE field of the PWM_CMRx register. This field is reset at 0.
- the ***waveform period***. This channel parameter is defined in the CPRD field of the PWM_CPRDx register.
  - If the waveform is left aligned, then the output waveform period depends on the counter source clock and can be calculated:
  By using the Master Clock (MCK) divided by an X given prescaler value
  (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024), the resulting period formula will be:

  $$\frac{(X \times CPRD)}{MCK}$$

  By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

  $$\frac{(X*CPRD*DIVA)}{MCK} \text{ or } \frac{(X*CPRD*DIVB)}{MCK}$$

  If the waveform is center aligned then the output waveform period depends on the counter source clock and can be calculated:
  By using the Master Clock (MCK) divided by an X given prescaler value
  (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

  $$\frac{(2 \times X \times CPRD)}{MCK}$$

  By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

  $$\frac{(2*X*CPRD*DIVA)}{MCK} \text{ or } \frac{(2*X*CPRD*DIVB)}{MCK}$$

- the ***waveform duty cycle***. This channel parameter is defined in the CDTY field of the PWM_CDTYx register.
  If the waveform is left aligned then:

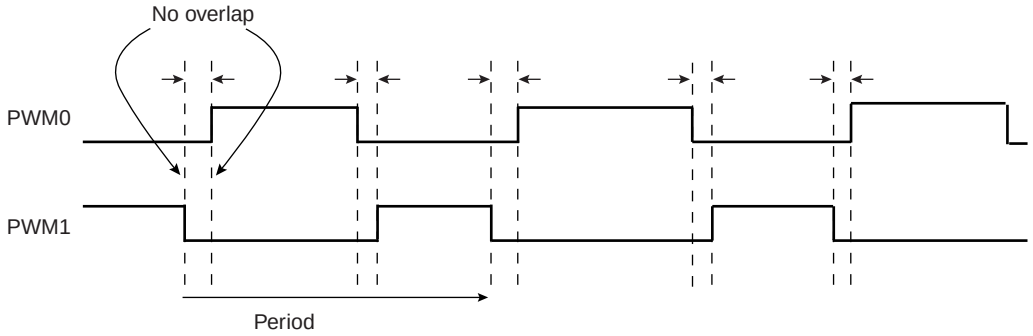
  $$\text{duty cycle} = \frac{(period - 1/\text{fchannel\_x\_clock} \times CDTY)}{period}$$

  If the waveform is center aligned, then:

  $$\text{duty cycle} = \frac{((period/2) - 1/\text{fchannel\_x\_clock} \times CDTY))}{(period/2)}$$

- the ***waveform polarity.*** At the beginning of the period, the signal can be at high or low level. This property is defined in the CPOL field of the PWM_CMRx register. By default the signal starts by a low level.
- the ***waveform alignment***. The output waveform can be left or center aligned. Center aligned waveforms can be used to generate non overlapped waveforms. This property is defined in the CALG field of the PWM_CMRx register. The default mode is left aligned.

Figure 37-4: Non Overlapped Center Aligned Waveforms



Note 1: See Figure 37-5 for a detailed description of center aligned waveforms.

When center aligned, the internal channel counter increases up to CPRD and.decreases down to 0. This ends the period.

When left aligned, the internal channel counter increases up to CPRD and is reset. This ends the period.

Thus, for the same CPRD value, the period for a center aligned channel is twice the period for a left aligned channel.

Waveforms are fixed at 0 when:

- CDTY = CPRD and CPOL = 0
- CDTY = 0 and CPOL = 1

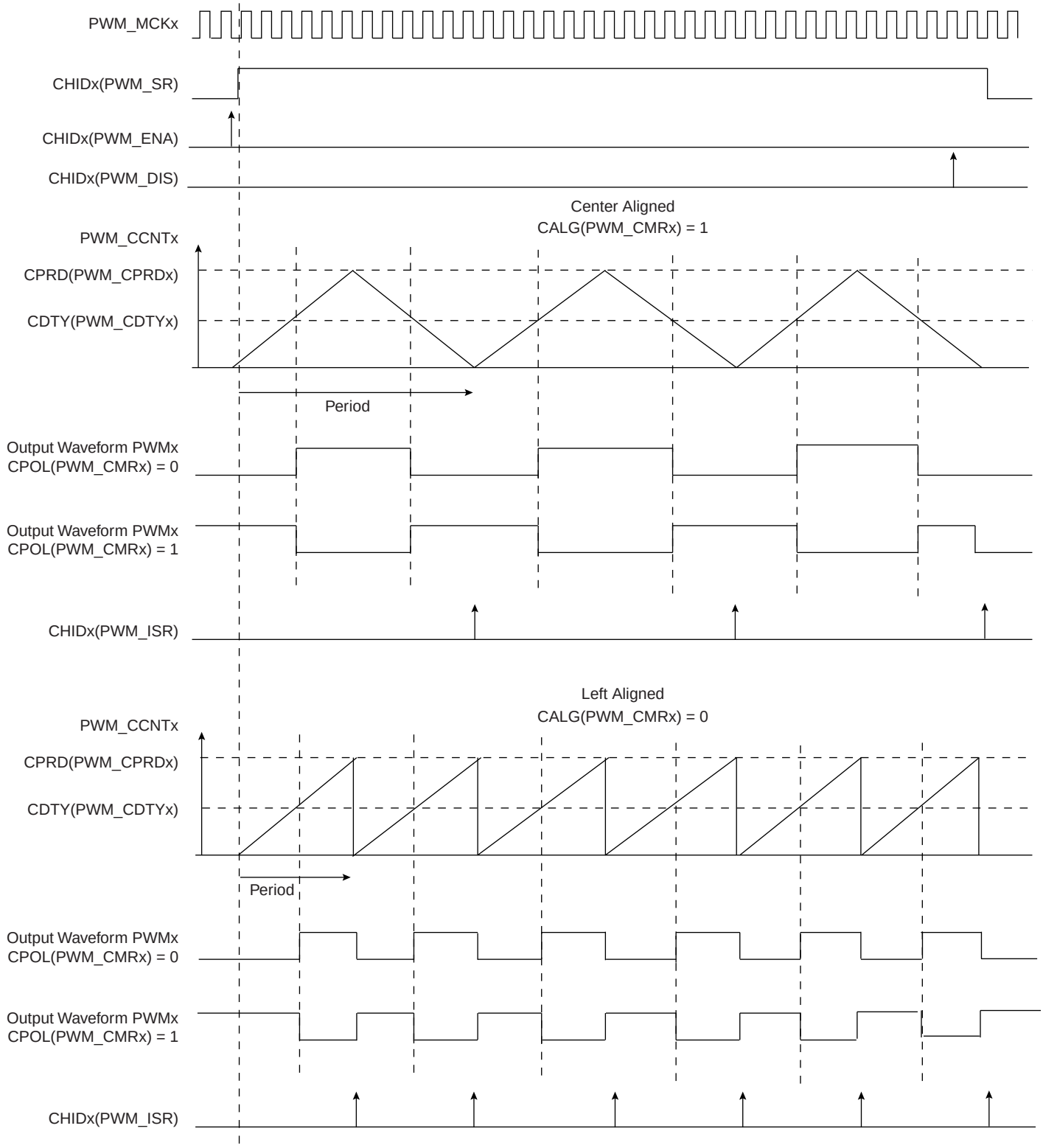
Waveforms are fixed at 1 (once the channel is enabled) when:

- CDTY = 0 and CPOL = 0
- CDTY = CPRD and CPOL = 1

The waveform polarity must be set before enabling the channel. This immediately affects the channel output level. Changes on channel polarity are not taken into account while the channel is enabled.

© 2017 Microchip Technology Inc.

Figure 37-5: Waveform Properties



37.6.3 PWM Controller Operations

37.6.3.1 Initialization

Before enabling the output channel, this channel must have been configured by the software application:

- Configuration of the clock generator if DIVA and DIVB are required
- Selection of the clock for each channel (CPRE field in the PWM_CMRx register)
- Configuration of the waveform alignment for each channel (CALG field in the PWM_CMRx register)
- Configuration of the period for each channel (CPRD in the PWM_CPRDx register). Writing in PWM_CPRDx Register is possible while the channel is disabled. After validation of the channel, the user must use PWM_CUPDx Register to update PWM_CPRDx as explained below.
- Configuration of the duty cycle for each channel (CDTY in the PWM_CDTYx register). Writing in PWM_CDTYx Register is possible while the channel is disabled. After validation of the channel, the user must use PWM_CUPDx Register to update PWM_CDTYx as explained below.
- Configuration of the output waveform polarity for each channel (CPOL in the PWM_CMRx register)
- Enable Interrupts (Writing CHIDx in the PWM_IER register)
- Enable the PWM channel (Writing CHIDx in the PWM_ENA register)

It is possible to synchronize different channels by enabling them at the same time by means of writing simultaneously several CHIDx bits in the PWM_ENA register.

- In such a situation, all channels may have the same clock selector configuration and the same period specified.

37.6.3.2 Source Clock Selection Criteria

The large number of source clocks can make selection difficult. The relationship between the value in the Period Register (PWM_CPRDx) and the Duty Cycle Register (PWM_CDTYx) can help the user in choosing. The event number written in the Period Register gives the PWM accuracy. The Duty Cycle quantum cannot be lower than *1/PWM_CPRDx* value. The higher the value of PWM_CPRDx, the greater the PWM accuracy.

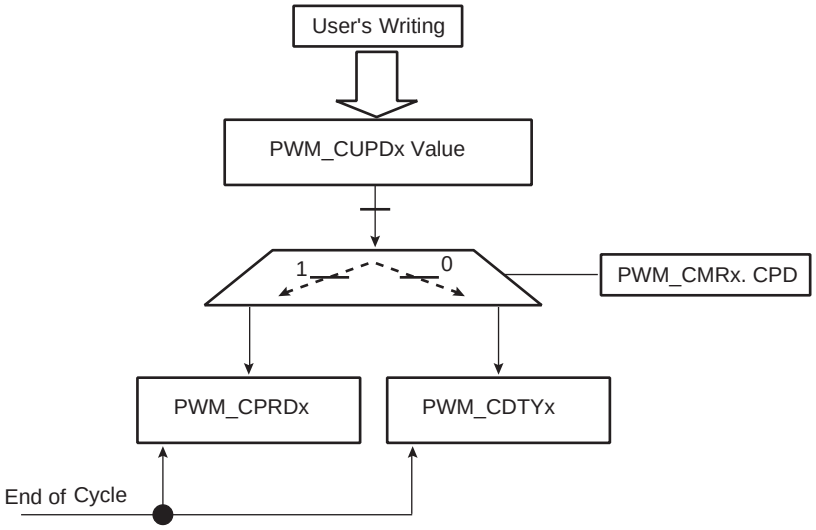
For example, if the user sets 15 (in decimal) in PWM_CPRDx, the user is able to set a value between 1 up to 14 in PWM_CDTYx Register. The resulting duty cycle quantum cannot be lower than 1/15 of the PWM period.

37.6.3.3 Changing the Duty Cycle or the Period

It is possible to modulate the output waveform duty cycle or period.

To prevent unexpected output waveform, the user must use the update register (PWM_CUPDx) to change waveform parameters while the channel is still enabled. The user can write a new period value or duty cycle value in the update register (PWM_CUPDx). This register holds the new value until the end of the current cycle and updates the value for the next cycle. Depending on the CPD field in the PWM_CMRx register, PWM_CUPDx either updates PWM_CPRDx or PWM_CDTYx. Note that even if the update register is used, the period must not be smaller than the duty cycle.

Figure 37-6: Synchronized Period or Duty Cycle Update



© 2017 Microchip Technology Inc.

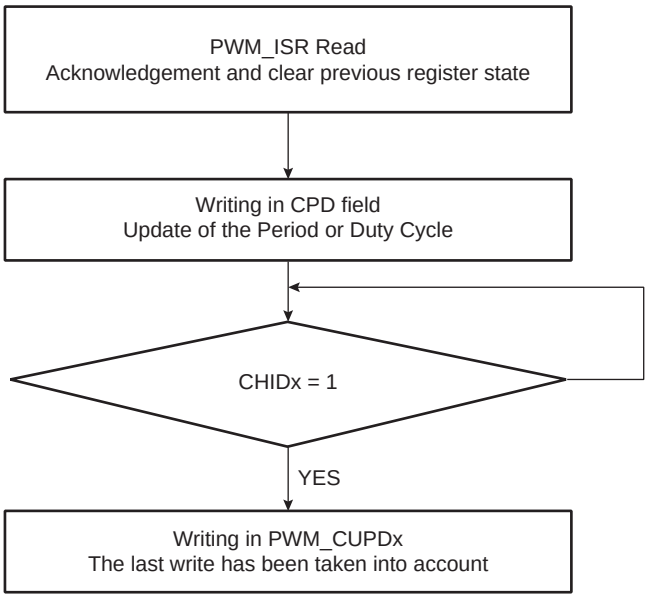
To prevent overwriting the PWM_CUPDx by software, the user can use status events in order to synchronize his software. Two methods are possible. In both, the user must enable the dedicated interrupt in PWM_IER at PWM Controller level.

The first method (polling method) consists of reading the relevant status bit in PWM_ISR Register according to the enabled channel(s). See Figure 37-7.

The second method uses an Interrupt Service Routine associated with the PWM channel.

Note: Reading the PWM_ISR register automatically clears CHIDx flags.

Figure 37-7: Polling Method



Note: Polarity and alignment can be modified only when the channel is disabled.

37.6.3.4 Interrupts

Depending on the interrupt mask in the PWM_IMR register, an interrupt is generated at the end of the corresponding channel period. The interrupt remains active until a read operation in the PWM_ISR register occurs.

A channel interrupt is enabled by setting the corresponding bit in the PWM_IER register. A channel interrupt is disabled by setting the corresponding bit in the PWM_IDR register.

37.7 Pulse Width Modulation Controller (PWM) User Interface

Table 37-4: Register Mapping[1]

Offset	Register	Name	Access	Reset
0x00	PWM Mode Register	PWM_MR	Read/Write	0
0x04	PWM Enable Register	PWM_ENA	Write-only	–
0x08	PWM Disable Register	PWM_DIS	Write-only	–
0x0C	PWM Status Register	PWM_SR	Read-only	0
0x10	PWM Interrupt Enable Register	PWM_IER	Write-only	–
0x14	PWM Interrupt Disable Register	PWM_IDR	Write-only	–
0x18	PWM Interrupt Mask Register	PWM_IMR	Read-only	0
0x1C	PWM Interrupt Status Register	PWM_ISR	Read-only	0
0x20 - 0xFC	Reserved	–	–	–
0x100 - 0x1FC	Reserved	–	–	–
0x200 + ch_num * 0x20 + 0x00	PWM Channel Mode Register	PWM_CMR	Read/Write	0x0
0x200 + ch_num * 0x20 + 0x04	PWM Channel Duty Cycle Register	PWM_CDTY	Read/Write	0x0
0x200 + ch_num * 0x20 + 0x08	PWM Channel Period Register	PWM_CPRD	Read/Write	0x0
0x200 + ch_num * 0x20 + 0x0C	PWM Channel Counter Register	PWM_CCNT	Read-only	0x0
0x200 + ch_num * 0x20 + 0x10	PWM Channel Update Register	PWM_CUPD	Write-only	–

Note 1: Some registers are indexed with “ch_num” index ranging from 0 to 3.

© 2017 Microchip Technology Inc.

37.7.1 PWM Mode Register

Name:PWM_MR

Address:0xF8034000

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	PREB			

23	22	21	20	19	18	17	16
DIVB							

15	14	13	12	11	10	9	8
–	–	–	–	PREA			

7	6	5	4	3	2	1	0
DIVA							

DIVA, DIVB: CLKA, CLKB Divide Factor

Value	Name	Description
0	CLK_OFF	CLKA, CLKB clock is turned off
1	CLK_DIV1	CLKA, CLKB clock is clock selected by PREA, PREB
2-255	–	CLKA, CLKB clock is clock selected by PREA, PREB divided by DIVA, DIVB factor.

PREA, PREB

Value	Name	Description
0000	MCK	Master Clock
0001	MCKDIV2	Master Clock divided by 2
0010	MCKDIV4	Master Clock divided by 4
0011	MCKDIV8	Master Clock divided by 8
0100	MCKDIV16	Master Clock divided by 16
0101	MCKDIV32	Master Clock divided by 32
0110	MCKDIV64	Master Clock divided by 64
0111	MCKDIV128	Master Clock divided by 128
1000	MCKDIV256	Master Clock divided by 256
1001	MCKDIV512	Master Clock divided by 512
1010	MCKDIV1024	Master Clock divided by 1024

Values which are not listed in the table must be considered as “reserved”.

37.7.2 PWM Enable Register

Name:PWM_ENA

Address:0xF8034004

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

CHIDx: Channel ID

0: No effect.

1: Enable PWM output for channel x.

© 2017 Microchip Technology Inc.

37.7.3 PWM Disable Register

Name:PWM_DIS

Address:0xF8034008

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

CHIDx: Channel ID

0: No effect.

1: Disable PWM output for channel x.

37.7.4 PWM Status Register

Name:PWM_SR

Address:0xF803400C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

CHIDx: Channel ID

0: PWM output for channel x is disabled.

1: PWM output for channel x is enabled.

© 2017 Microchip Technology Inc.

37.7.5 PWM Interrupt Enable Register

Name:PWM_IER

Address:0xF8034010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

CHIDx: Channel ID.

0: No effect.

1: Enable interrupt for PWM channel x.

37.7.6 PWM Interrupt Disable Register

Name:PWM_IDR

Address:0xF8034014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

CHIDx: Channel ID.

0: No effect.

1: Disable interrupt for PWM channel x.

© 2017 Microchip Technology Inc.

37.7.7 PWM Interrupt Mask Register

Name:PWM_IMR

Address:0xF8034018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

CHIDx: Channel ID.

0: Interrupt for PWM channel x is disabled.

1: Interrupt for PWM channel x is enabled.

37.7.8 PWM Interrupt Status Register

Name:PWM_ISR

Address:0xF803401C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

CHIDx: Channel ID

0: No new channel period has been achieved since the last read of the PWM_ISR register.

1: At least one new channel period has been achieved since the last read of the PWM_ISR register.

Note: Reading PWM_ISR automatically clears CHIDx flags.

© 2017 Microchip Technology Inc.

37.7.9 PWM Channel Mode Register

Name:PWM_CMR[0..3]

Address:0xF8034200 [0], 0xF8034220 [1], 0xF8034240 [2], 0xF8034260 [3]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	CPD	CPOL	CALG
7	6	5	4	3	2	1	0
–	–	–	–	CPRE			

CPRE: Channel Pre-scaler

Value	Name	Description
0000	MCK	Master Clock
0001	MCKDIV2	Master Clock divided by 2
0010	MCKDIV4	Master Clock divided by 4
0011	MCKDIV8	Master Clock divided by 8
0100	MCKDIV16	Master Clock divided by 16
0101	MCKDIV32	Master Clock divided by 32
0110	MCKDIV64	Master Clock divided by 64
0111	MCKDIV128	Master Clock divided by 128
1000	MCKDIV256	Master Clock divided by 256
1001	MCKDIV512	Master Clock divided by 512
1010	MCKDIV1024	Master Clock divided by 1024
1011	CLKA	Clock A
1100	CLKB	Clock B

Values which are not listed in the table must be considered as “reserved”.

CALG: Channel Alignment

0: The period is left aligned.

1: The period is center aligned.

CPOL: Channel Polarity

0: The output waveform starts at a low level.

1: The output waveform starts at a high level.

CPD: Channel Update Period

0: Writing to the PWM_CUPDx will modify the duty cycle at the next period start event.

1: Writing to the PWM_CUPDx will modify the period at the next period start event.

© 2017 Microchip Technology Inc.

37.7.10 PWM Channel Duty Cycle Register

Name:PWM_CDTY[0..3]

Address:0xF8034204 [0], 0xF8034224 [1], 0xF8034244 [2], 0xF8034264 [3]

Access: Read/Write

31	30	29	28	27	26	25	24
CDTY							

23	22	21	20	19	18	17	16
CDTY							

15	14	13	12	11	10	9	8
CDTY							

7	6	5	4	3	2	1	0
CDTY							

Only the first 32 bits (internal channel counter size) are significant.

CDTY: Channel Duty Cycle

Defines the waveform duty cycle. This value must be defined between 0 and CPRD (PWM_CPRx).

© 2017 Microchip Technology Inc.

37.7.11 PWM Channel Period Register

Name:PWM_CPRD[0..3]

Address:0xF8034208 [0], 0xF8034228 [1], 0xF8034248 [2], 0xF8034268 [3]

Access: Read/Write

31	30	29	28	27	26	25	24
CPRD							

23	22	21	20	19	18	17	16
CPRD							

15	14	13	12	11	10	9	8
CPRD							

7	6	5	4	3	2	1	0
CPRD							

Only the first 32 bits (internal channel counter size) are significant.

CPRD: Channel Period

If the waveform is left-aligned, then the output waveform period depends on the counter source clock and can be calculated:

– By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(X \times CPRD)}{MCK}$$

– By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPD \times DIVA)}{MCK} \text{ or } \frac{(CRPD \times DIVAB)}{MCK}$$

If the waveform is center-aligned, then the output waveform period depends on the counter source clock and can be calculated:

– By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

– By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

© 2017 Microchip Technology Inc.

37.7.12 PWM Channel Counter Register

Name:PWM_CCNT[0..3]

Address:0xF803420C [0], 0xF803422C [1], 0xF803424C [2], 0xF803426C [3]

Access: Read-only

31	30	29	28	27	26	25	24
CNT							

23	22	21	20	19	18	17	16
CNT							

15	14	13	12	11	10	9	8
CNT							

7	6	5	4	3	2	1	0
CNT							

CNT: Channel Counter Register

Internal counter value. This register is reset when:

- the channel is enabled (writing CHIDx in the PWM_ENA register).
- the counter reaches CPRD value defined in the PWM_CPRDx register if the waveform is left aligned.

© 2017 Microchip Technology Inc.

37.7.13 PWM Channel Update Register

Name:PWM_CUPD[0..3]

Address:0xF8034210 [0], 0xF8034230 [1], 0xF8034250 [2], 0xF8034270 [3]

Access: Write-only

31	30	29	28	27	26	25	24
CUPD							

23	22	21	20	19	18	17	16
CUPD							

15	14	13	12	11	10	9	8
CUPD							

7	6	5	4	3	2	1	0
CUPD							

CUPD: Channel Update Register

This register acts as a double buffer for the period or the duty cycle. This prevents an unexpected waveform when modifying the waveform period or duty-cycle.

Only the first 32 bits (internal channel counter size) are significant.

When CPD field of PWM_CMRx register = 0, the duty-cycle (CDTY of PWM_CDTYx register) is updated with the CUPD value at the beginning of the next period.

When CPD field of PWM_CMRx register = 1, the period (CPRD of PWM_CPRDx register) is updated with the CUPD value at the beginning of the next period.

38. Two-wire Interface (TWI)

38.1 Description

The Microchip Two-wire Interface (TWI) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 Kbits per second, based on a byte-oriented transfer format. It can be used with any Microchip Two-wire Interface bus Serial EEPROM and I²C compatible device such as a Real Time Clock (RTC), Dot Matrix/Graphic LCD Controllers and temperature sensor. The TWI is programmable as a master or a slave with sequential or single-byte access. Multiple master capability is supported.

A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.

Table 38-1 lists the compatibility level of the Microchip Two-wire Interface in Master mode and a full I^2C compatible device.

Table 38-1: Microchip TWI Compatibility with I^2C Standard

I^2C Standard	Microchip TWI
Standard Mode Speed (100 kHz)	Supported
Fast Mode Speed (400 kHz)	Supported
7- or 10-bit Slave Addressing	Supported
START byte(1)	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NACK Management	Supported
Slope Control and Input Filtering (Fast mode)	Not Supported
Clock Stretching/Synchronization	Supported
Multi Master Capability	Supported

Note 1: START + b000000001 + Ack + Sr

38.2 Embedded Characteristics

- Compatible with Microchip Two-wire Interface Serial Memory and I^2C Compatible Devices(1)
- One, Two or Three Bytes for Slave Address
- Sequential Read/Write Operations
- Master, Multi-master and Slave Mode Operation
- Bit Rate: Up to 400 Kbit/s
- General Call Supported in Slave Mode
- Connection to DMA Controller (DMA) Channel Capabilities Optimizes Data Transfers
- Register Write Protection

Note 1: See Table 38-1 for details on compatibility with I²C Standard.

© 2017 Microchip Technology Inc.

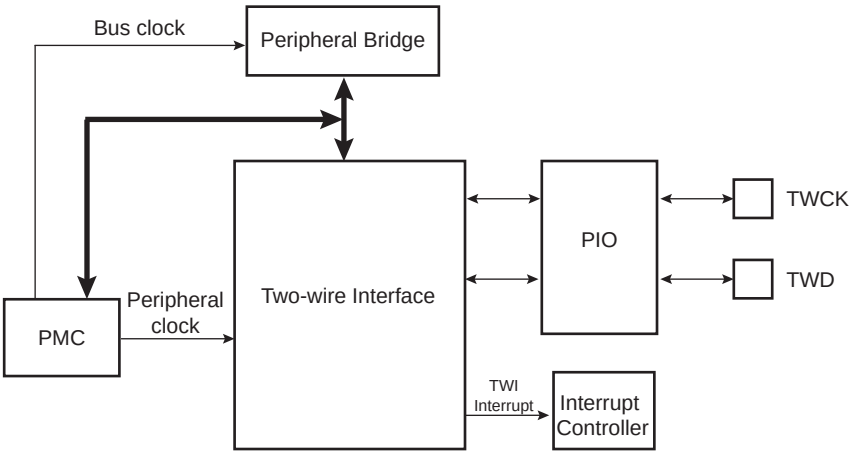
38.3 List of Abbreviations

Table 38-2: Abbreviations

Abbreviation	Description
TWI	Two-wire Interface
A	Acknowledge
NA	Non Acknowledge
P	Stop
S	Start
Sr	Repeated Start
SADR	Slave Address
ADR	Any address except SADR
R	Read
W	Write

38.4 Block Diagram

Figure 38-1: Block Diagram



38.5 I/O Lines Description

Table 38-3: I/O Lines Description

Name	Description	Type
TWD	Two-wire Serial Data (drives external serial data line – SDA)	Input/Output
TWCK	Two-wire Serial Clock (drives external serial clock line – SCL)	Input/Output

38.6 Product Dependencies

38.6.1 I/O Lines

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor. When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWI, the user must program the PIO Controller to dedicate TWD and TWCK as peripheral lines.

The user must not program TWD and TWCK as open-drain. This is already done by the hardware.

Table 38-4: I/O Lines

Instance	Signal	I/O Line	Peripheral
TWI0	TWCK0	PA31	A
TWI0	TWD0	PA30	A
TWI1	TWCK1	PC1	C
TWI1	TWD1	PC0	C

38.6.2 Power Management

The TWI may be clocked through the Power Management Controller (PMC), thus the user must first configure the PMC to enable the TWI clock.

38.6.3 Interrupt Sources

The TWI has an interrupt line connected to the Interrupt Controller. In order to handle interrupts, the Interrupt Controller must be programmed before configuring the TWI.

Table 38-5: Peripheral IDs

Instance	ID
TWI0	9
TWI1	10

38.7 Functional Description

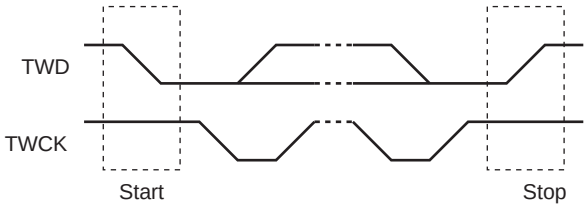
38.7.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited (see Figure 38-3).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 38-2).

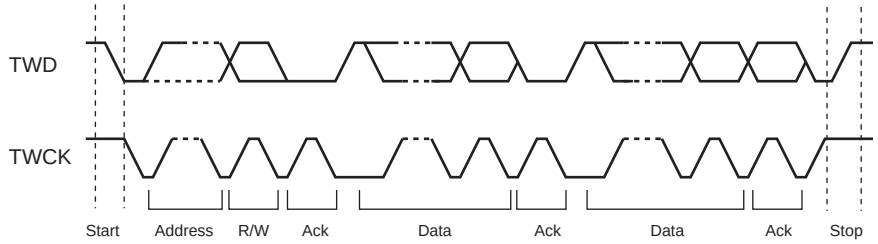
- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines the STOP condition.

Figure 38-2: START and STOP Conditions



© 2017 Microchip Technology Inc.

Figure 38-3: Transfer Format



38.7.2 Modes of Operation

The TWI has different modes of operations:

- Master transmitter mode
- Master receiver mode
- Multi-master transmitter mode
- Multi-master receiver mode
- Slave transmitter mode
- Slave receiver mode

These modes are described in the following sections.

38.7.3 Master Mode

38.7.3.1 Definition

The master is the device that starts a transfer, generates a clock and stops it.

38.7.3.2 Programming Master Mode

The following fields must be programmed before entering Master mode:

1. TWI_MMR.DADR (+ IADRSZ + IADR if a 10-bit device is addressed): The device address is used to access slave devices in Read or Write mode.
2. TWI_CWGR.CKDIV + CHDIV + CLDIV: Clock waveform.
3. TWI_CR.SVDIS: Disables the Slave mode
4. TWI_CR.MSEN: Enables the Master mode

Note: If the TWI is already in Master mode, the device address (DADR) can be configured without disabling the Master mode.

38.7.3.3 Master Transmitter Mode

After the master initiates a START condition when writing into the Transmit Holding register (TWI_THR), it sends a 7-bit slave address, configured in the Master Mode register (DADR in TWI_MMR), to notify the slave device. The bit following the slave address indicates the transfer direction—0 in this case (MREAD = 0 in TWI_MMR).

The TWI transfers require the slave to acknowledge each received byte. During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. If the slave does not acknowledge the byte, then the Not Acknowledge flag (NACK) is set in the TWI Status Register (TWI_SR) of the master and a STOP condition is sent. The NACK flag must be cleared by reading the TWI Status Register (TWI_SR) before the next write into the TWI Transmit Holding Register (TWI_THR). As with the other status bits, an interrupt can be generated if enabled in the Interrupt Enable register (TWI_IER). If the slave acknowledges the byte, the data written in the TWI_THR is then shifted in the internal shifter and transferred. When an acknowledge is detected, the TXRDY bit is set until a new write in the TWI_THR.

When no more data is written into the TWI_THR, the master generates a STOP condition to end the transfer. A TXCOMP bit value of one in the TWI_SR indicates that the transfer has completed. See Figure 38-4, Figure 38-5, and Figure 38-6.

To clear the TXRDY flag, first set the bit TWI_CR.MSDIS, then set the bit TWI_CR.MSEN.

Figure 38-4: Master Write with One Data Byte

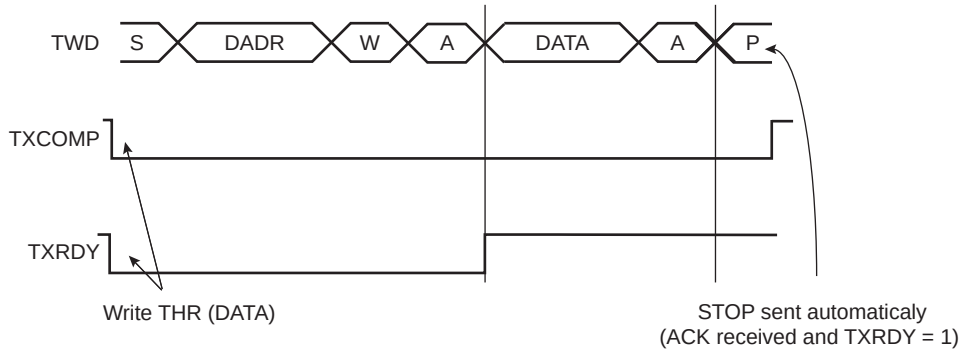


Figure 38-5: Master Write with Multiple Data Byte

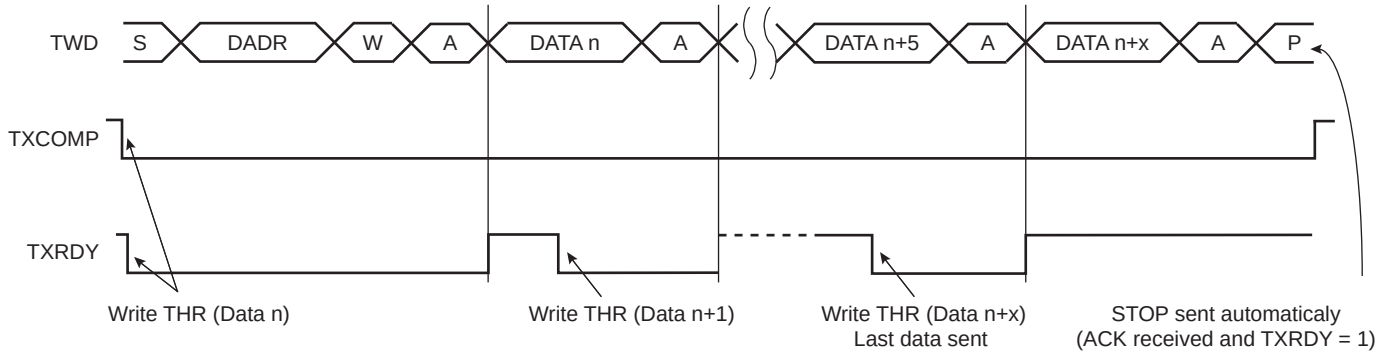
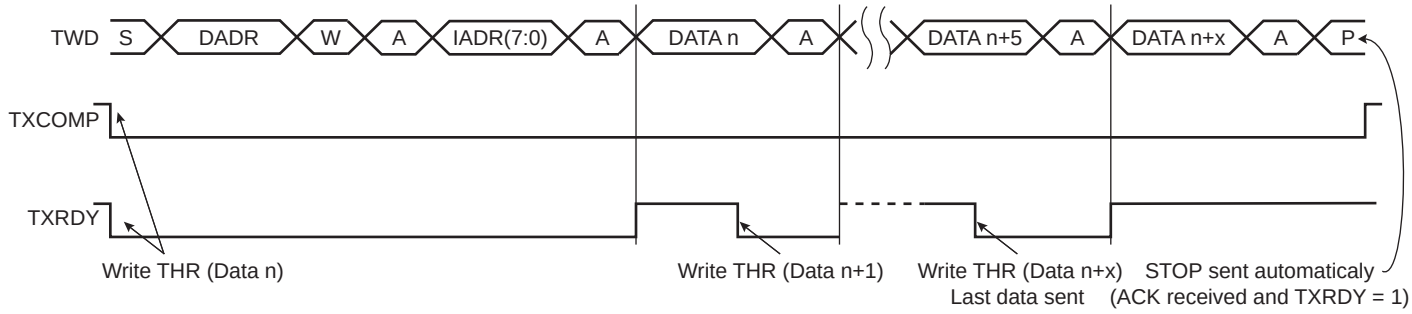


Figure 38-6: Master Write with One Byte Internal Address and Multiple Data Bytes



38.7.3.4 Master Receiver Mode

The read sequence begins by setting the START bit. After the START condition has been sent, the master sends a 7-bit slave address to notify the slave device. The bit following the slave address indicates the transfer direction—1 in this case (MREAD = 1 in TWI_MMR). During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the NACK bit in the TWI_SR if the slave does not acknowledge the byte.

If an acknowledge is received, the master is then ready to receive data from the slave. After data has been received, the master sends an acknowledge condition to notify the slave that the data has been received except for the last data. See . When the RXRDY bit is set in the TWI_SR, a character has been received in the Receive Holding Register (TWI_RHR). The RXRDY bit is reset when reading the TWI_RHR.

When a single data byte read is performed, with or without internal address (IADR), the START and STOP bits must be set at the same time. See . When a multiple data byte read is performed, with or without internal address (IADR), the STOP bit must be set after the next-to-last data received. See . For internal address usage, see **Section 38.7.3.5 “Internal Address”**.

© 2017 Microchip Technology Inc.

If the Receive Holding Register (TWI_RHR) is full (RXRDY high) and the master is receiving data, the serial clock line is tied low before receiving the last bit of the data and until the TWI_RHR is read. Once the TWI_RHR is read, the master stops stretching the serial clock line and ends the data reception. See Figure 38-9.

Warning: When receiving multiple bytes in Master read mode, if the next-to-last access is not read (the RXRDY flag remains high), the last access is not completed until TWI_RHR is read. The last access stops on the next-to-last bit. When the TWI_RHR is read, the STOP bit command must be sent within a period of half a bit only, otherwise another read access might occur (spurious access).

A possible workaround is to set the STOP bit before reading the TWI_RHR on the next-to-last access (within the interrupt handler).

Figure 38-7: Master Read with One Data Byte

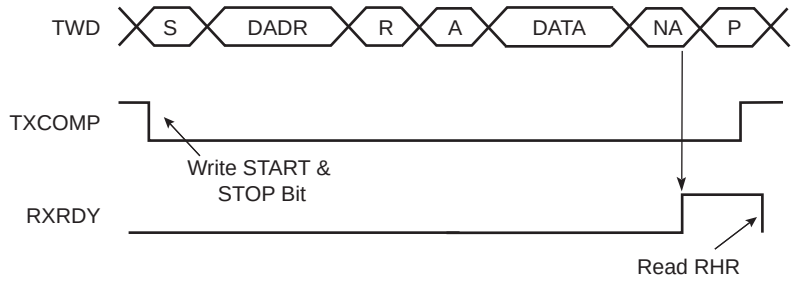


Figure 38-8: Master Read with Multiple Data Bytes

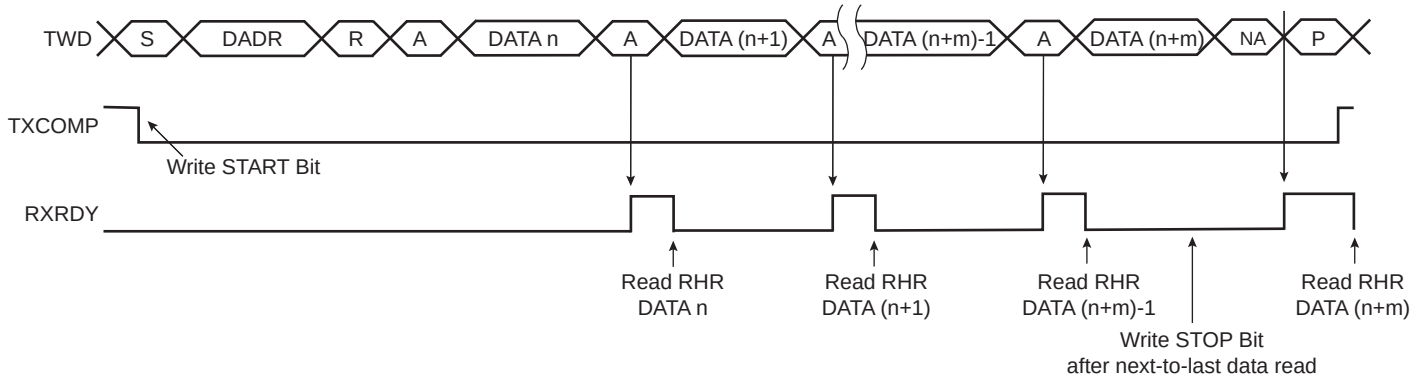
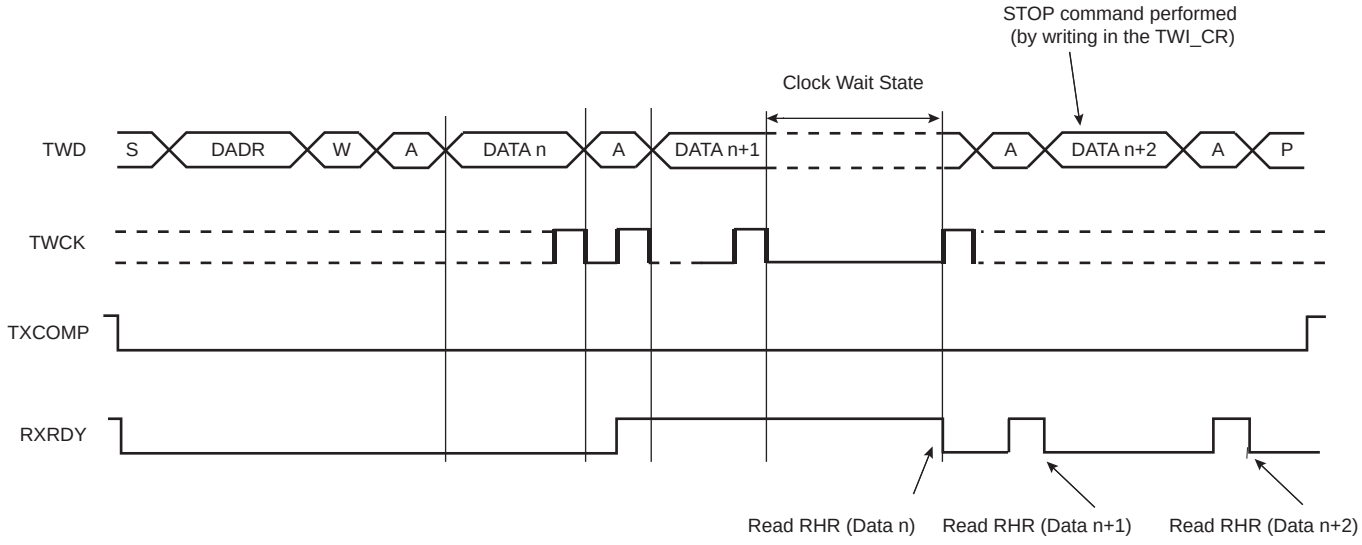


Figure 38-9: Master Read Wait State with Multiple Data Bytes



38.7.3.5 Internal Address

The TWI can perform transfers with 7-bit slave address devices and 10-bit slave address devices.

7-bit Slave Addressing

When addressing 7-bit slave devices, the internal address bytes are used to perform random address (read or write) accesses to reach one or more data bytes, e.g. within a memory page location in a serial memory. When performing read operations with an internal address, the TWI performs a write operation to set the internal address into the slave device, and then switch to Master receiver mode. Note that the second START condition (after sending the IADR) is sometimes called “repeated start” (Sr) in I^2C fully-compatible devices. See Figure 38-11. See Figure 38-10 and Figure 38-12 for master write operation with internal address.

The three internal address bytes are configurable through the Master Mode register (TWI_MMR).

If the slave device supports only a 7-bit address, i.e., no internal address, IADRSZ must be set to 0.

Table 38-6 shows the abbreviations used in Figure 38-10 and Figure 38-11.

Table 38-6: Abbreviations

Abbreviation	Definition
S	Start
Sr	Repeated Start
P	Stop
W	Write
R	Read
A	Acknowledge
NA	Not Acknowledge
DADR	Device Address
IADR	Internal Address

Figure 38-10: Master Write with One, Two or Three Bytes Internal Address and One Data Byte

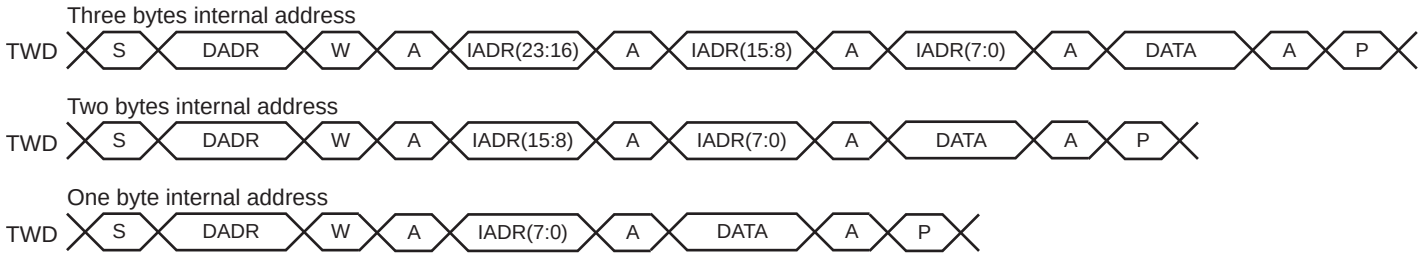
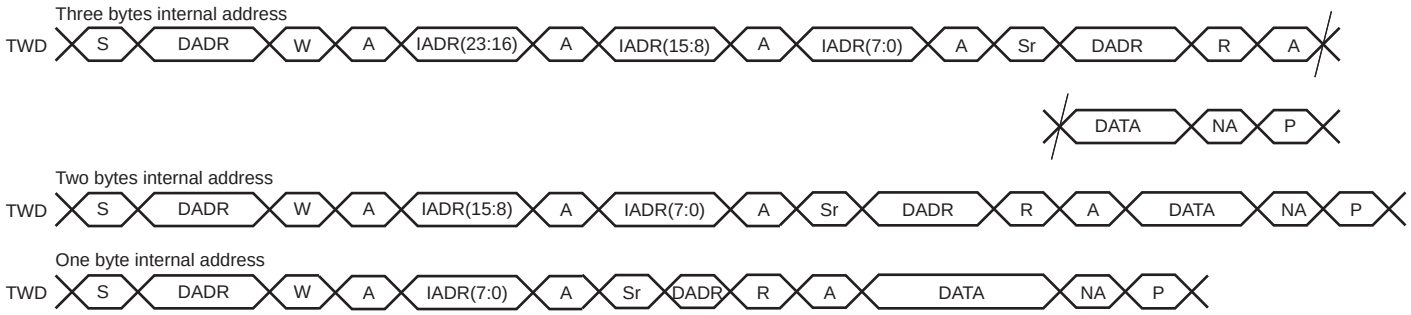


Figure 38-11: Master Read with One, Two or Three Bytes Internal Address and One Data Byte



© 2017 Microchip Technology Inc.

10-bit Slave Addressing

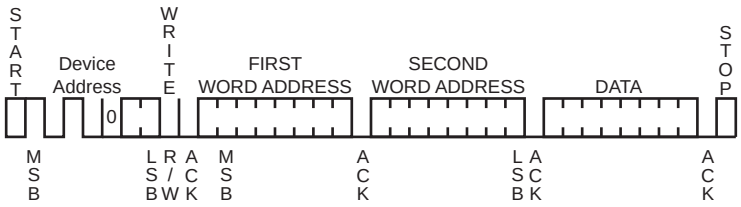
For a slave address higher than seven bits, the user must configure the address size **(**IADRSZ) and set the other slave address bits in the Internal Address register (TWI_IADR). The two remaining internal address bytes, IADR[15:8] and IADR[23:16] can be used the same way as in 7-bit slave addressing.

Example: Address a 10-bit device (10-bit device address is b1 b2 b3 b4 b5 b6 b7 b8 b9 b10)

1. Program IADRSZ = 1,
2. Program DADR with 1 1 1 1 0 b1 b2 (b1 is the MSB of the 10-bit address, b2, etc.)
3. Program TWI_IADR with b3 b4 b5 b6 b7 b8 b9 b10 (b10 is the LSB of the 10-bit address)

Figure 38-12 below shows a byte write to a memory device. This demonstrates the use of internal addresses to access the device.

Figure 38-12: Internal Address Usage



38.7.3.6 Using the DMA Controller

The use of the DMA significantly reduces the CPU load.

To ensure correct implementation, proceed as follows.

Data Transmit with the DMA

1. Initialize the DMA (channels, memory pointers, size -1, etc.).
2. Configure the Master mode (DADR, CKDIV, etc.) or Slave mode.
3. Enable the DMA.
4. Wait for the DMA buffer transfer complete flag.
5. Disable the DMA.
6. Wait for the TXRDY flag in TWI_SR.
7. Set the STOP bit in TWI_CR.
8. Write the last character in TWI_THR.
9. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWI_SR.

Data Receive with the DMA

The DMA transfer size must be defined with the buffer size minus 2. The two remaining characters must be managed without DMA to ensure that the exact number of bytes are received whatever the system bus latency conditions encountered during the end of buffer transfer period.

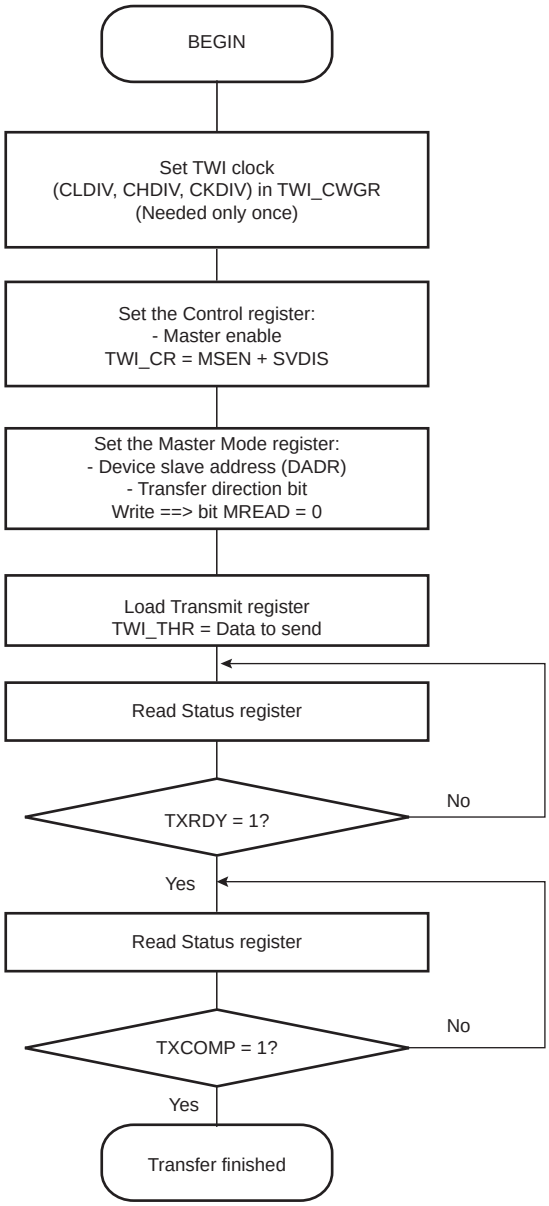
In Slave mode, the number of characters to receive must be known in order to configure the DMA.

1. Initialize the DMA (channels, memory pointers, size -2, etc.);
2. Configure the Master mode (DADR, CKDIV, etc.) or Slave mode.
3. Enable the DMA.
4. (Master Only) Write the START bit in the TWI_CR to start the transfer.
5. Wait for the DMA buffer transfer complete flag.
6. Disable the DMA.
7. Wait for the RXRDY flag in the TWI_SR.
8. Set the STOP bit in TWI_CR.
9. Read the penultimate character in TWI_RHR.
10. Wait for the RXRDY flag in the TWI_SR.
11. Read the last character in TWI_RHR.
12. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWI_SR.

38.7.3.7 Read/Write Flowcharts

The flowcharts in the following figures provide examples of read and write operations. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the Interrupt Enable Register (TWI_IER) be configured first.

Figure 38-13: TWI Write Operation with Single Data Byte without Internal Address



© 2017 Microchip Technology Inc.

Figure 38-14: TWI Write Operation with Single Data Byte and Internal Address

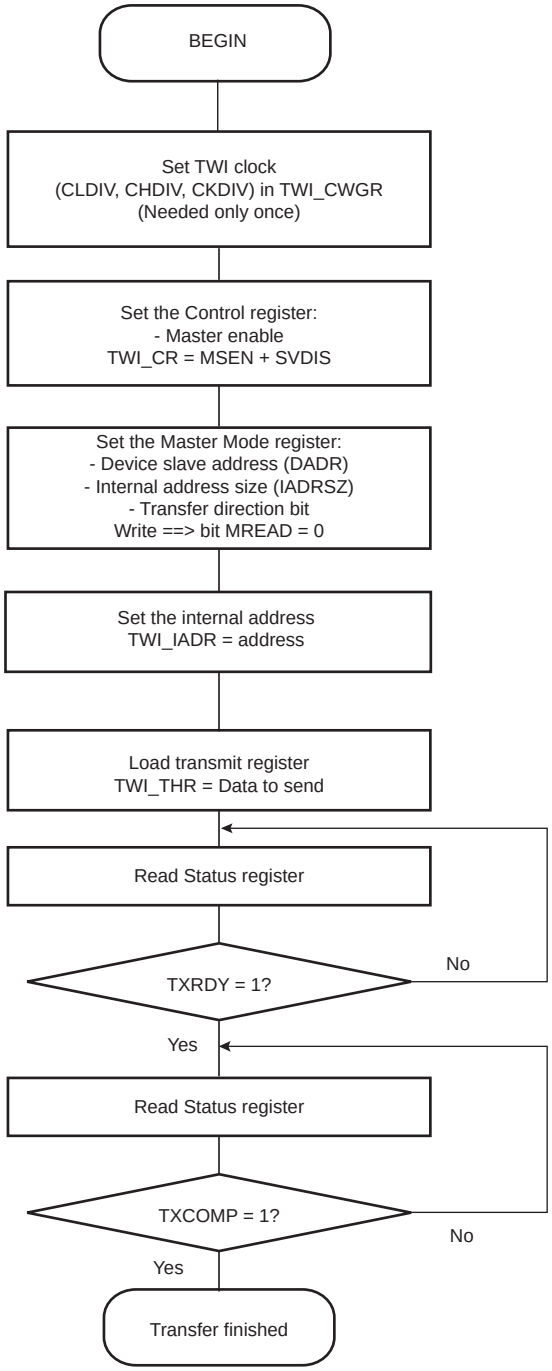
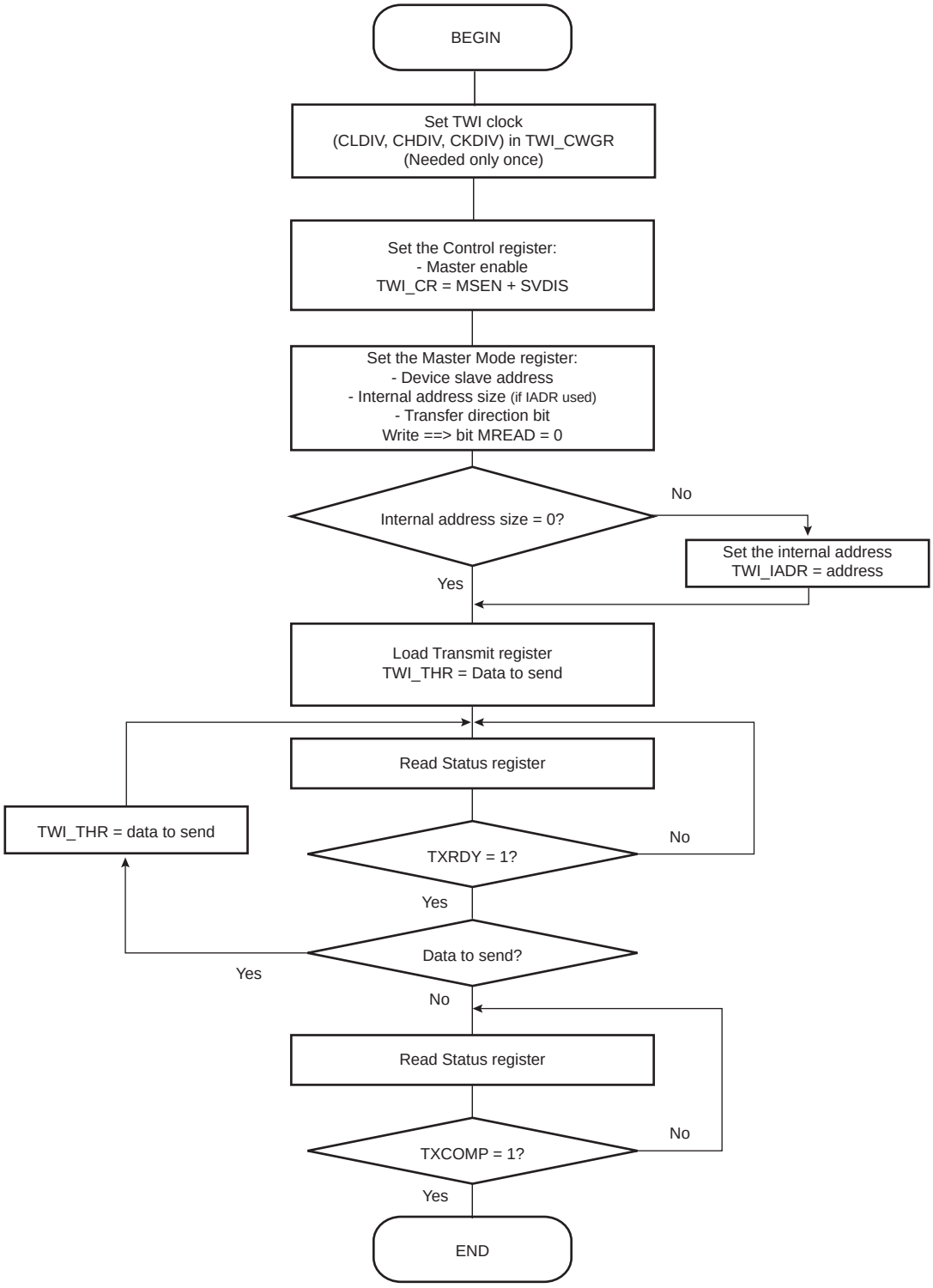


Figure 38-15: TWI Write Operation with Multiple Data Bytes with or without Internal Address



© 2017 Microchip Technology Inc.

Figure 38-16: TWI Read Operation with Single Data Byte without Internal Address

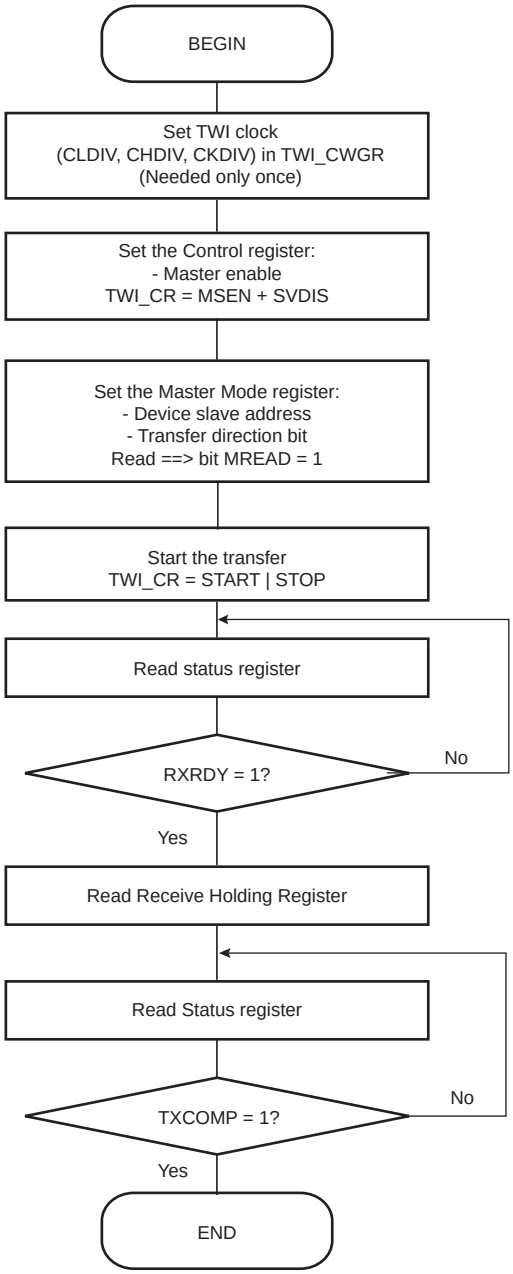
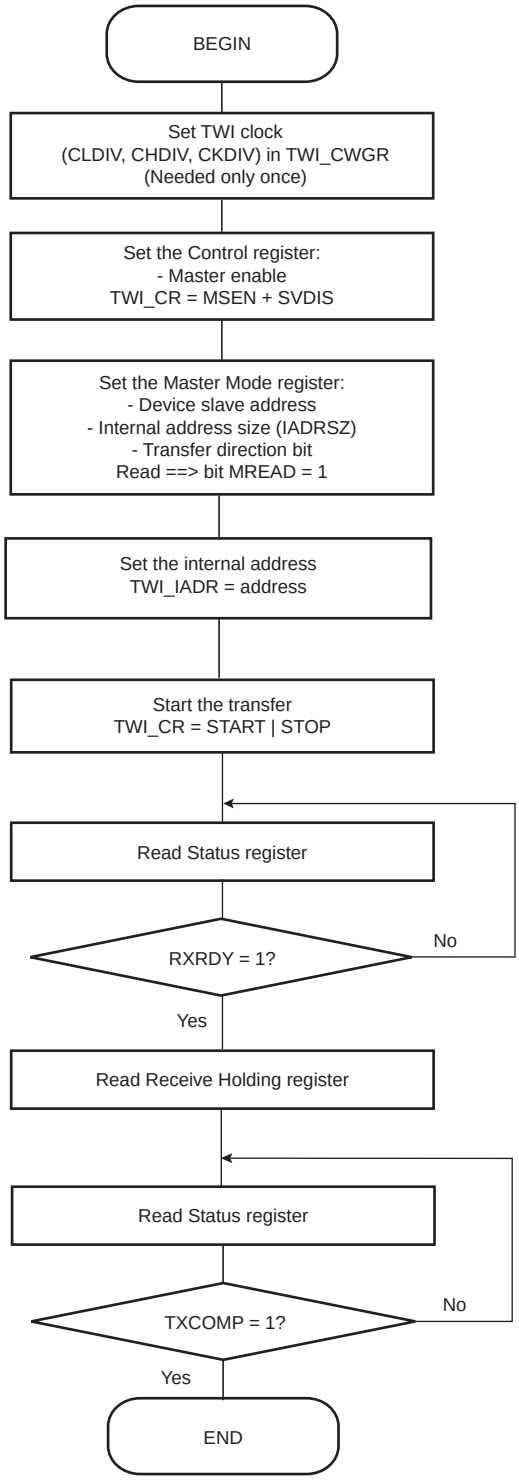
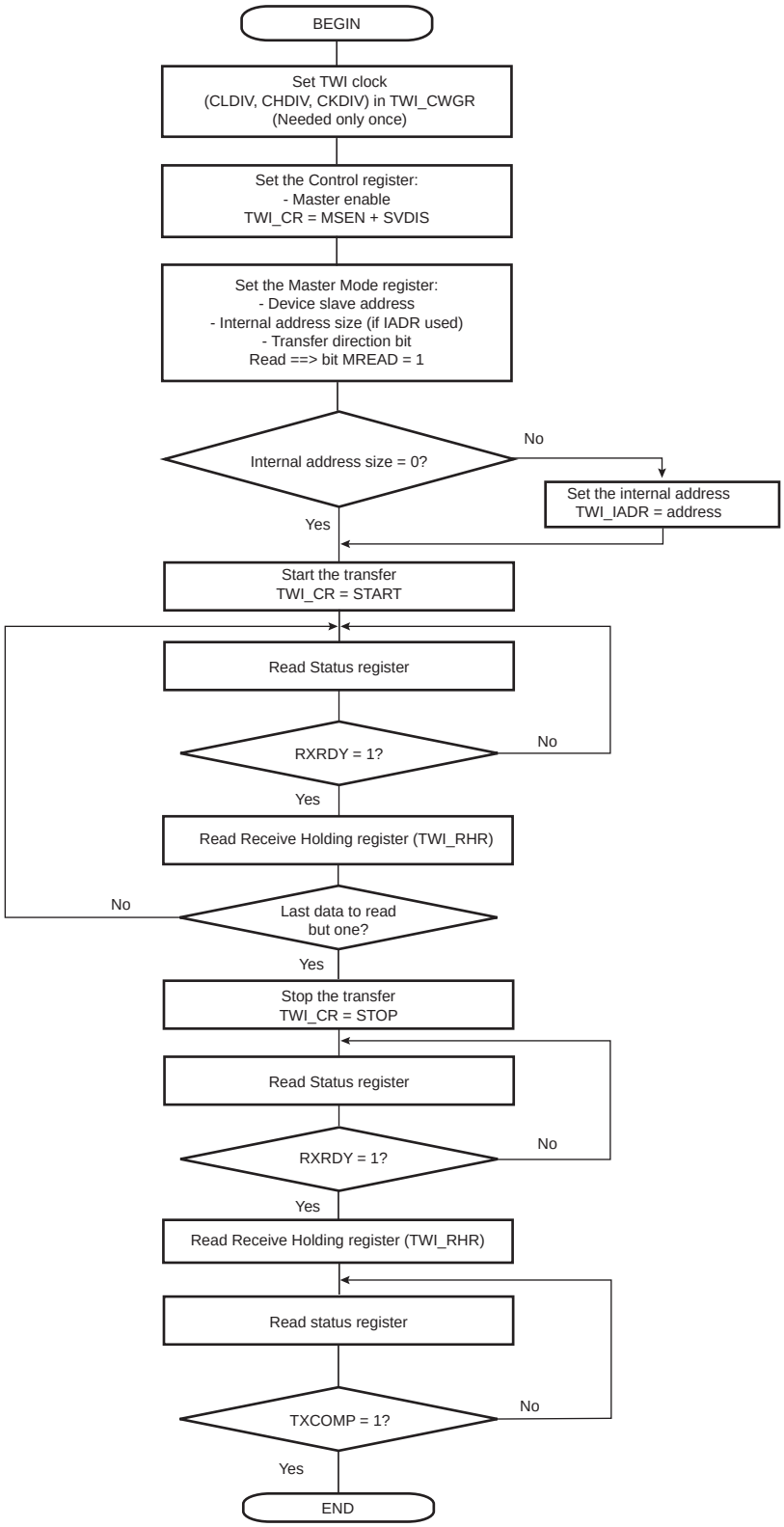


Figure 38-17: TWI Read Operation with Single Data Byte and Internal Address



© 2017 Microchip Technology Inc.

Figure 38-18: TWI Read Operation with Multiple Data Bytes with or without Internal Address



38.7.4 Multi-master Mode

38.7.4.1 Definition

In Multi-master mode, more than one master may handle the bus at the same time without data corruption by using arbitration.

Arbitration starts as soon as two or more masters place information on the bus at the same time, and stops (arbitration is lost) for the master that intends to send a logical one while the other master sends a logical zero.

As soon as a master lose arbitration, it stops sending data and listens to the bus in order to detect a stop. When the stop is detected, the master may put its data on the bus by performing arbitration.

Arbitration is illustrated in Figure 38-20.

38.7.4.2 Two Multi-master Modes

Two Multi-master modes may be distinguished:

1. TWI is considered as a master only and will never be addressed.
2. TWI may be either a master or a slave and may be addressed.

Note: Arbitration is supported in both Multi-master modes.

TWI as Master Only

In this mode, TWI is considered as a Master only (MSEN is always one) and must be driven like a Master with the ARBLST (Arbitration Lost) flag in addition.

If arbitration is lost (ARBLST = 1), the user must reinitiate the data transfer.

If the user starts a transfer (ex.: DADR + START + W + Write in THR) and if the bus is busy, the TWI automatically waits for a STOP condition on the bus to initiate the transfer (see Figure 38-19).

Note: The state of the bus (busy or free) is not shown in the user interface.

TWI as Master or Slave

The automatic reversal from Master to Slave is not supported in case of a lost arbitration.

Then, in the case where TWI may be either a Master or a Slave, the user must manage the pseudo Multi-master mode described in the steps below.

1. Program TWI in Slave mode (SADR + MSDIS + SVEN) and perform a slave access (if TWI is addressed).
2. If the TWI has to be set in Master mode, wait until the TXCOMP flag is at 1.
3. Program the Master mode (DADR + SVDIS + MSEN) and start the transfer (ex: START + Write in THR).
4. As soon as the Master mode is enabled, the TWI scans the bus in order to detect if it is busy or free. When the bus is considered free, TWI initiates the transfer.
5. As soon as the transfer is initiated and until a STOP condition is sent, the arbitration becomes relevant and the user must monitor the ARBLST flag.
6. If the arbitration is lost (ARBLST is set to 1), the user must program the TWI in Slave mode in case the Master that won the arbitration is required to access the TWI.
7. If the TWI has to be set in Slave mode, wait until TXCOMP flag is at 1 and then program the Slave mode.

Note: If the arbitration is lost and the TWI is addressed, the TWI will not acknowledge even if it is programmed in Slave mode as soon as ARBLST is set to 1. Then the Master must repeat SADR.

© 2017 Microchip Technology Inc.

Figure 38-19: Programmer Sends Data While the Bus is Busy

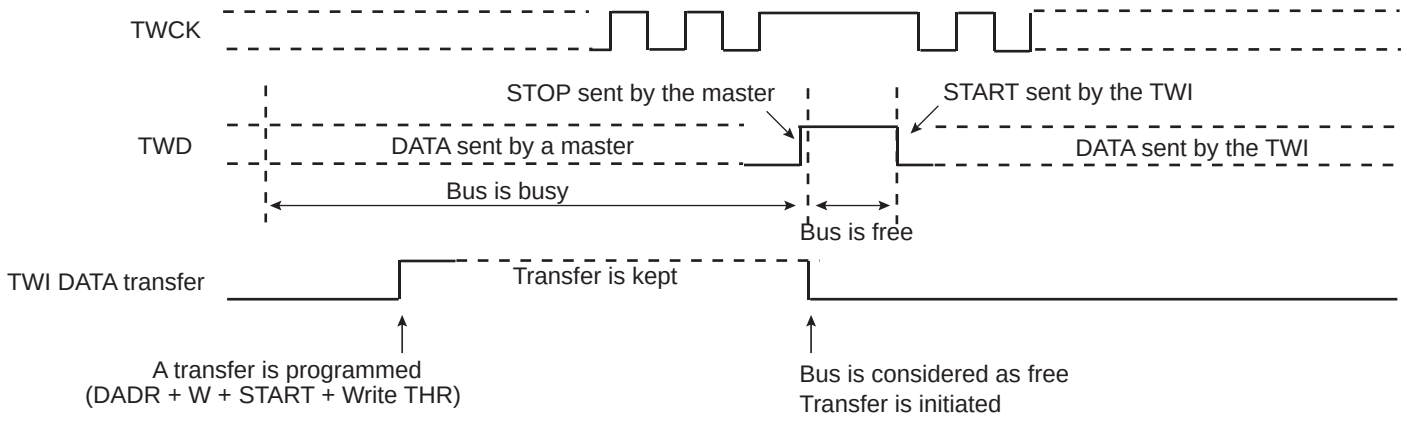
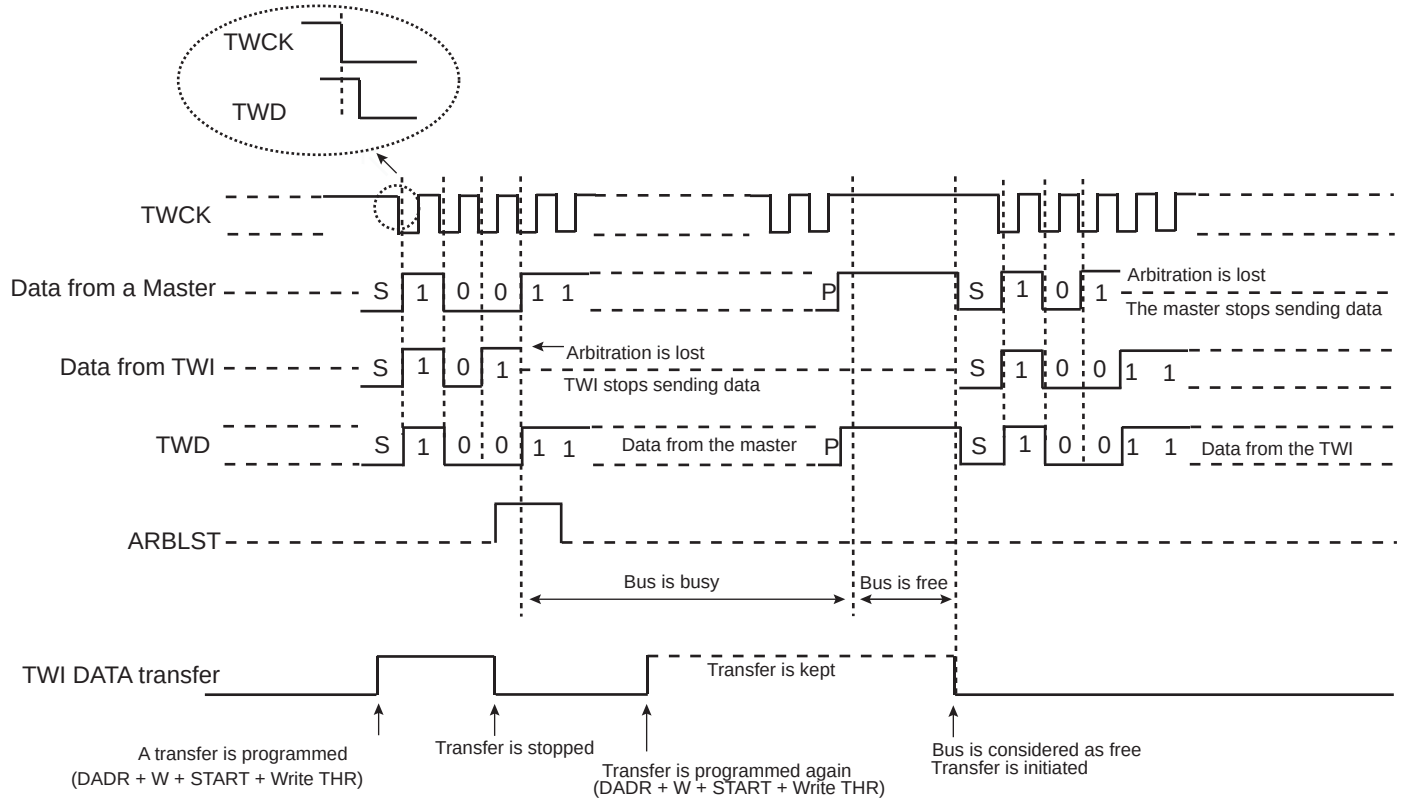
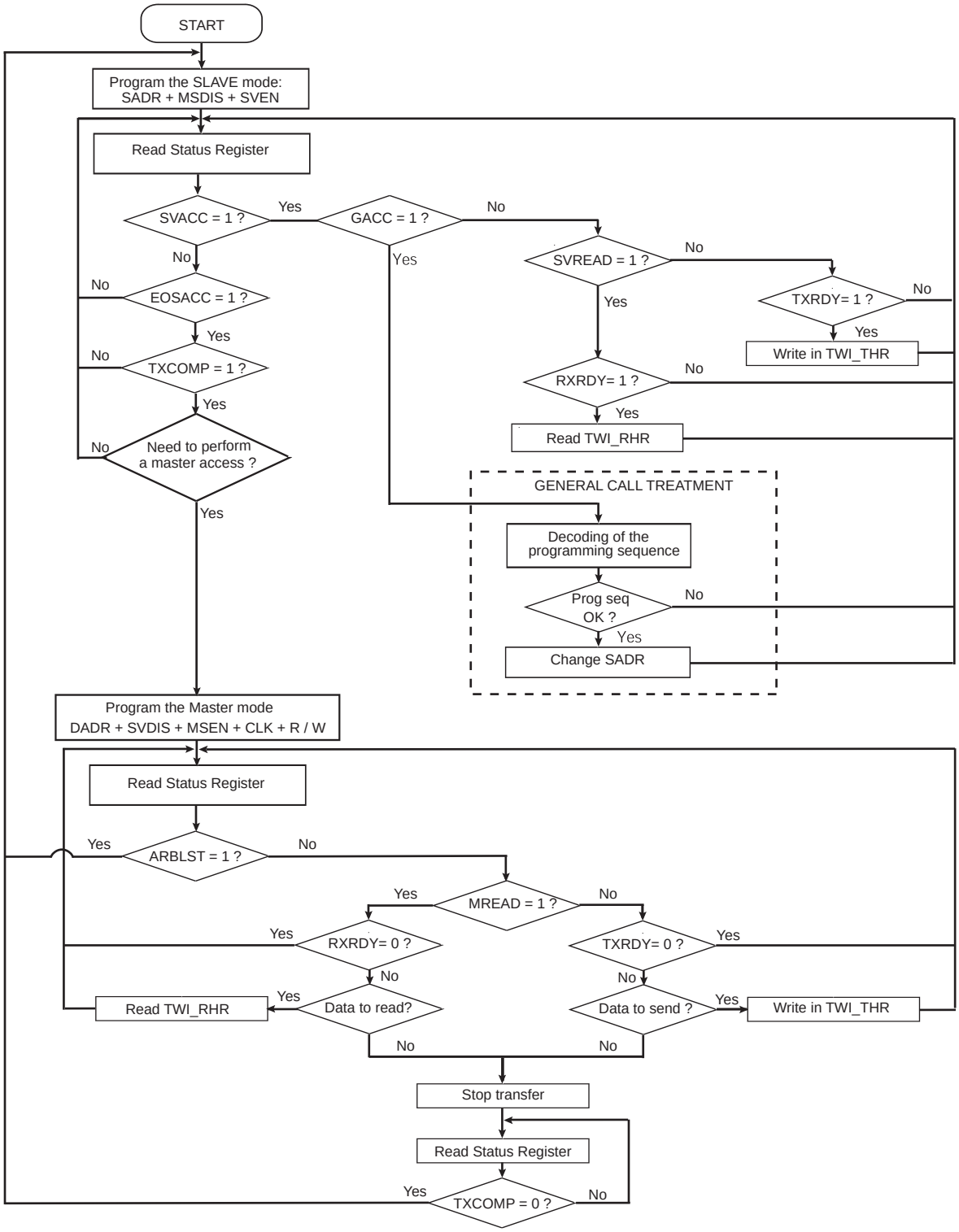


Figure 38-20: Arbitration Cases



The flowchart shown in Figure 38-21 gives an example of read and write operations in Multi-master mode.

Figure 38-21: Multi-master Flowchart



© 2017 Microchip Technology Inc.

38.7.5 Slave Mode

38.7.5.1 Definition

Slave mode is defined as a mode where the device receives the clock and the address from another device called the master.

In this mode, the device never initiates and never completes the transmission (START, REPEATED START and STOP conditions are always provided by the master).

38.7.5.2 Programming Slave Mode

The following fields must be programmed before entering Slave mode:

1. TWI_SMR.SADR: The slave device address is used in order to be accessed by master devices in Read or Write mode.
2. TWI_CR.MSDIS: Disables the Master mode.
3. TWI_CR.SVEN: Enables the Slave mode.

As the device receives the clock, values written in TWI_CWGR are ignored.

38.7.5.3 Receiving Data

After a START or REPEATED START condition is detected and if the address sent by the Master matches with the Slave address programmed in the SADR (Slave Address) field, SVACC (Slave Access) flag is set and SVREAD (Slave Read) indicates the direction of the transfer.

SVACC remains high until a STOP condition or a repeated START is detected. When such a condition is detected, the EOSACC (End Of Slave Access) flag is set.

Read Sequence

In the case of a read sequence (SVREAD is high), TWI transfers data written in TWI_THR (TWI Transmit Holding Register) until a STOP condition or a REPEATED_START and an address different from SADR is detected. Note that at the end of the read sequence TXCOMP (Transmission Complete) flag is set and SVACC reset.

As soon as data is written in TWI_THR, the TXRDY (Transmit Holding Register Ready) flag is reset, and it is set when the internal shifter is empty and the sent data acknowledged or not. If the data is not acknowledged, the NACK flag is set.

Note that a STOP or a REPEATED START always follows a NACK.

To clear the TXRDY flag, first set the bit TWI_CR.SVDIS, then set the bit TWI_CR.SVEN.

See Figure 38-22.

Write Sequence

In the case of a write sequence (SVREAD is low), the RXRDY (Receive Holding Register Ready) flag is set as soon as a character has been received in the TWI_RHR (TWI Receive Holding Register). RXRDY is reset when reading the TWI_RHR.

TWI continues receiving data until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the write sequence TXCOMP flag is set and SVACC reset.

See Figure 38-23.

Clock Synchronization Sequence

If TWI_RHR is not read in time, the TWI performs a clock synchronization.

Clock synchronization information is given by the bit SCLWS (Clock Wait State).

See Figure 38-26.

Clock Stretching Sequence

If TWI_THR is not written in time, the TWI performs a clock stretching.

Clock stretching information is given by the bit SCLWS (Clock Wait State).

See Figure 38-25.

General Call

In the case where a GENERAL CALL is performed, the GACC (General Call Access) flag is set.

After GACC is set, the user must interpret the meaning of the GENERAL CALL and decode the new address programming sequence.

See Figure 38-24.

© 2017 Microchip Technology Inc.

38.7.5.4 Data Transfer

Read Operation

The Read mode is defined as a data requirement from the master.

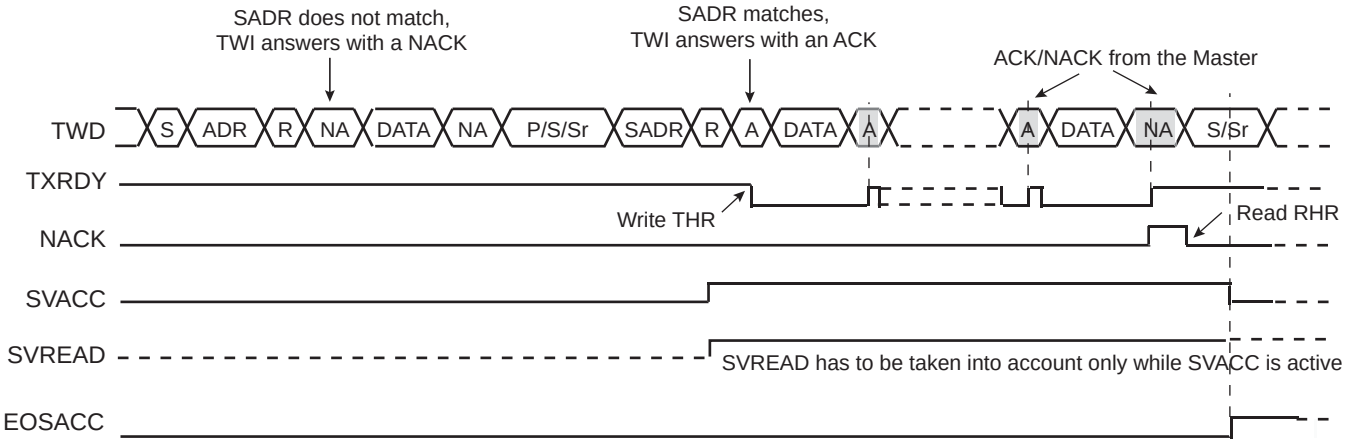
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

Until a STOP or REPEATED START condition is detected, TWI continues sending data loaded in the TWI_THR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 38-22 describes the write operation.

Figure 38-22: Read Access Ordered by a Master



Note 1: When SVACC is low, the state of SVREAD becomes irrelevant.

2: TXRDY is reset when data has been transmitted from TWI_THR to the internal shifter and set when this data has been acknowledged or non acknowledged.

Write Operation

The Write mode is defined as a data transmission from the master.

After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

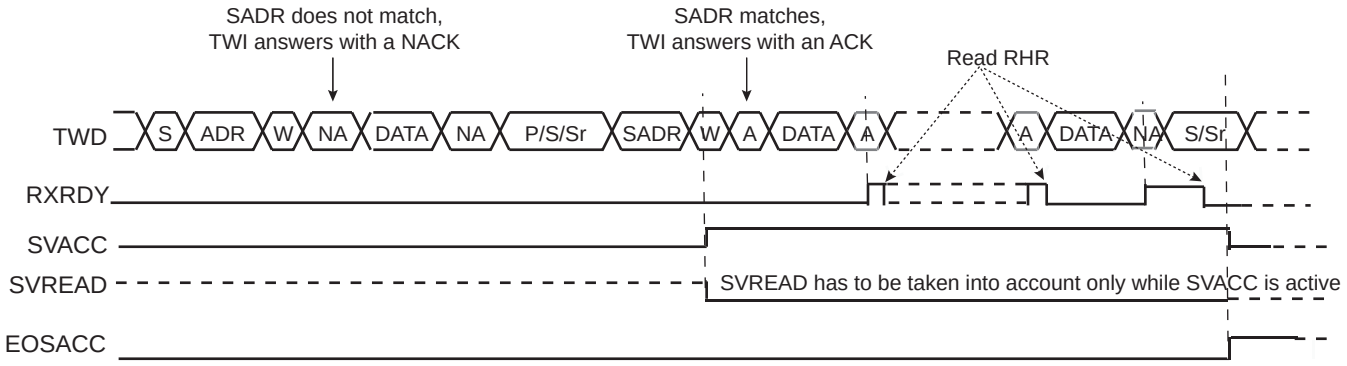
Until a STOP or REPEATED START condition is detected, TWI stores the received data in the TWI_RHR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 38-23 describes the write operation.

© 2017 Microchip Technology Inc.

Figure 38-23: Write Access Ordered by a Master



Note 1: When SVACC is low, the state of SVREAD becomes irrelevant.

2: RXRDY is set when data has been transmitted from the internal shifter to the TWI_RHR and reset when this data is read.

General Call

The general call is performed in order to change the address of the slave.

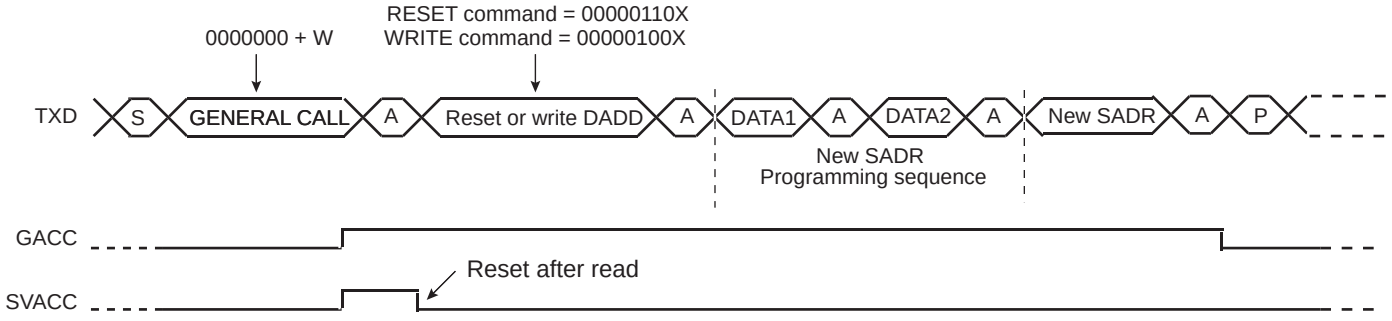
If a GENERAL CALL is detected, GACC is set.

After the detection of GENERAL CALL, it is up to the programmer to decode the commands which come afterwards.

In case of a WRITE command, the programmer has to decode the programming sequence and program a new SADR if the programming sequence matches.

Figure 38-24 describes the GENERAL CALL access.

Figure 38-24: Master Performs a General Call



Note: This method allows the user to create a personal programming sequence by choosing the programming bytes and the number of them. The programming sequence has to be provided to the master.

Clock Synchronization/Stretching

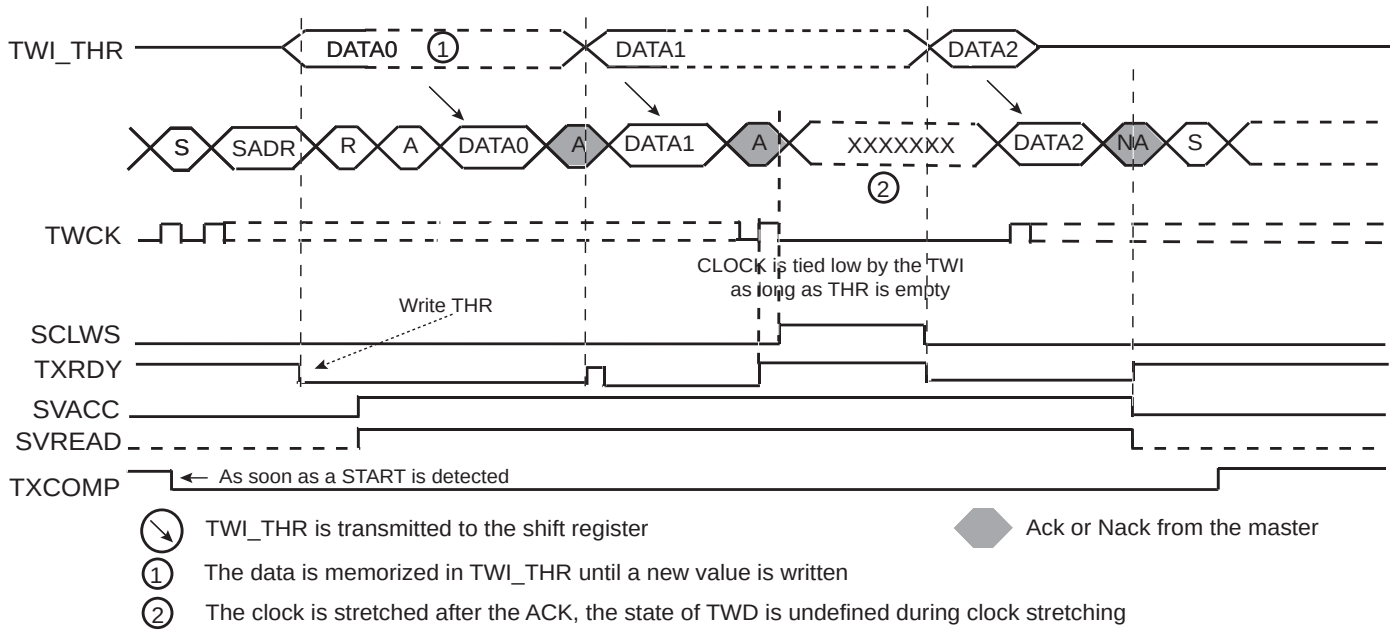
In both Read and Write modes, it may occur that TWI_THR/TWI_RHR buffer is not filled /emptied before transmission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching/synchronization mechanism is implemented.

- Clock Stretching in Read Mode

  The clock is tied low during the acknowledge phase if the internal shifter is empty and if a STOP or REPEATED START condition was not detected. It is tied low until the internal shifter is loaded.

  Figure 38-25 describes clock stretching in Read mode.

Figure 38-25: Clock Stretching in Read Mode



Note 1: TXRDY is reset when data has been written in the TWI_THR to the internal shifter and set when this data has been acknowledged or non acknowledged.

2: At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.

3: SCLWS is automatically set when the clock stretching mechanism is started.

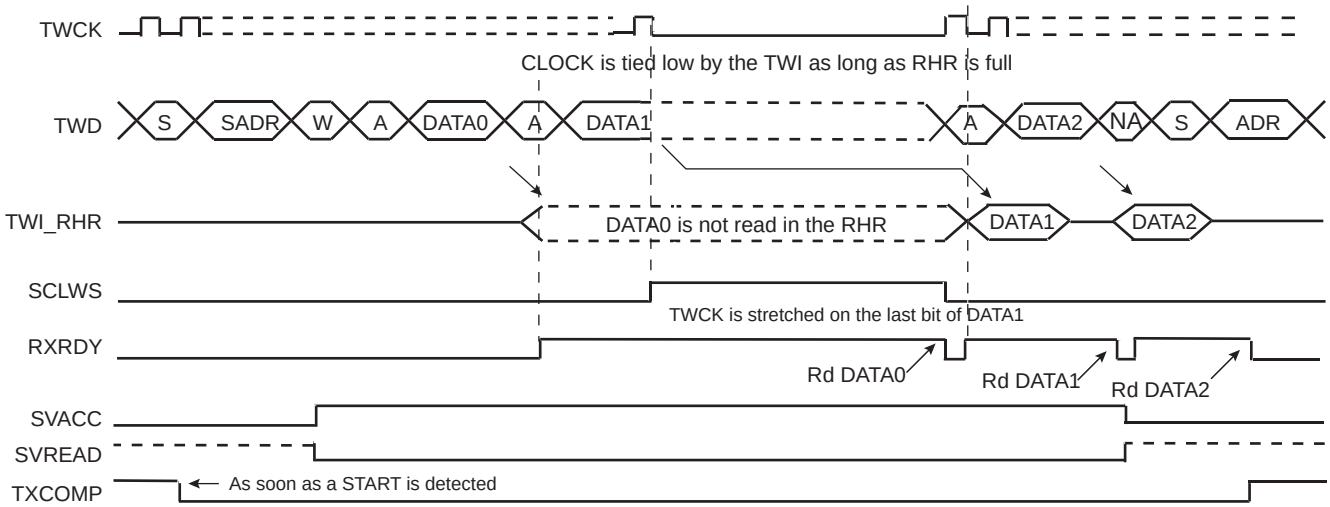
© 2017 Microchip Technology Inc.

- Clock Synchronization in Write Mode

  The clock is tied low outside of the acknowledge phase if the internal shifter and the TWI_RHR is full. If a STOP or REPEATED_START condition was not detected, it is tied low until TWI_RHR is read.

  Figure 38-26 describes the clock synchronization in Write mode.

Figure 38-26: Clock Synchronization in Write Mode



Note 1: At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.

2: SCLWS is automatically set when the clock synchronization mechanism is started and automatically reset when the mechanism is finished.

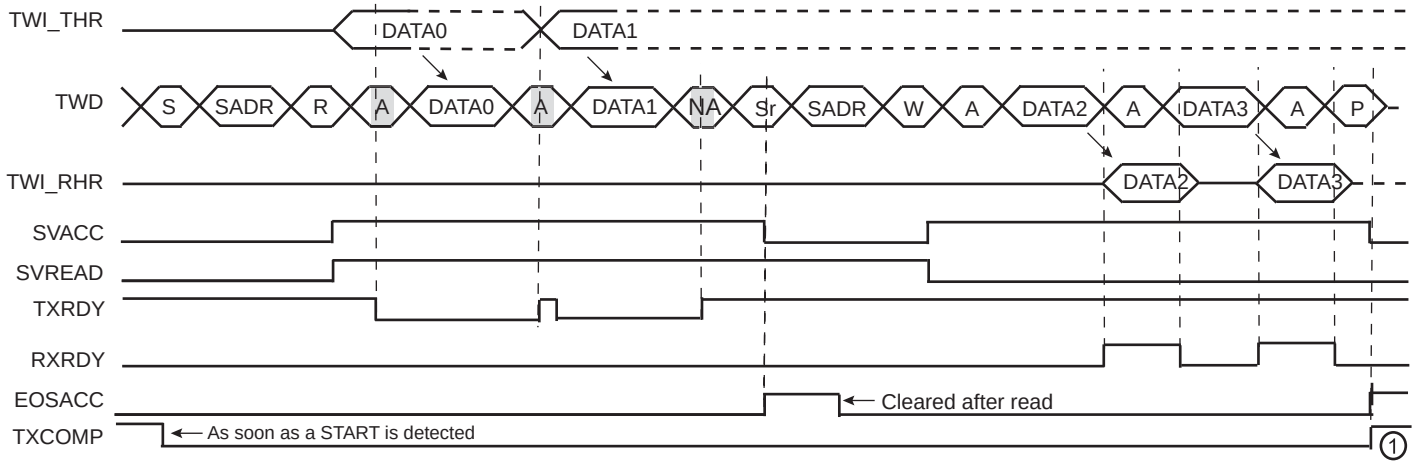
Reversal After a Repeated Start

- Reversal of Read to Write

  The master initiates the communication by a read command and finishes it by a write command.

  Figure 38-27 describes the repeated start + reversal from Read to Write mode.

Figure 38-27: Repeated Start + Reversal from Read to Write Mode



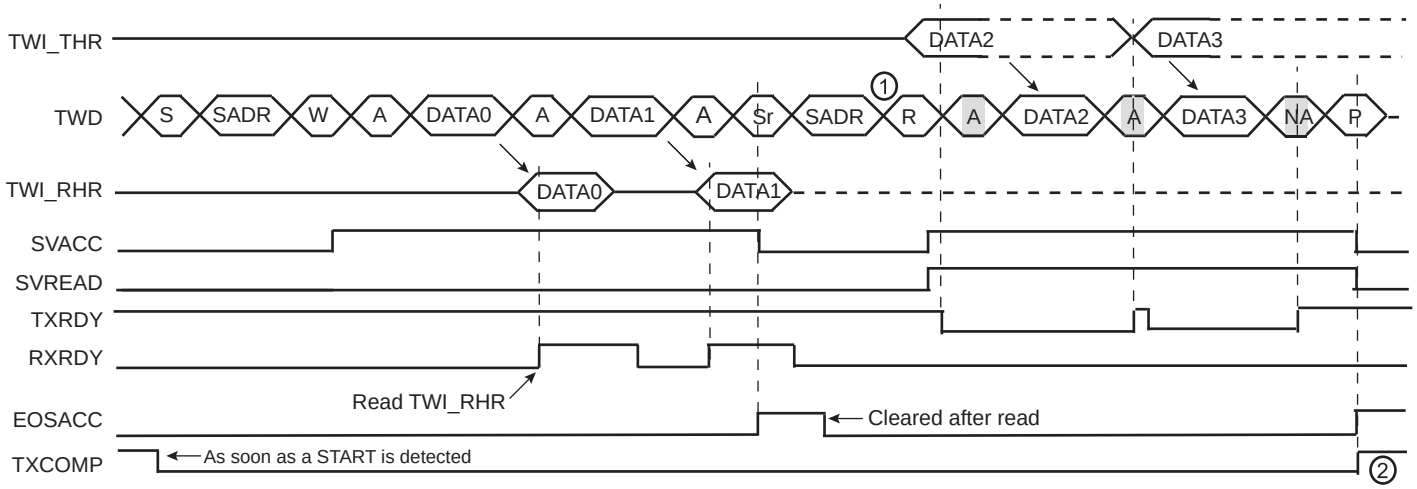
Note 1: TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

- Reversal of Write to Read

  The master initiates the communication by a write command and finishes it by a read command.

  Figure 38-28 describes the repeated start + reversal from Write to Read mode.

Figure 38-28: Repeated Start + Reversal from Write to Read Mode



Note 1: In this case, if TWI_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.

2: TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

© 2017 Microchip Technology Inc.

38.7.5.5 Using the DMA Controller

The use of the DMA significantly reduces the CPU load.

To ensure correct implementation, proceed as follows.

Data Transmit with the DMA

1. Initialize the DMA (channels, memory pointers, size, etc.).
2. Configure the Slave mode.
3. Enable the DMA.
4. Wait for the DMA buffer transfer complete flag.
5. Disable the DMA.
6. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWI_SR.

Data Receive with the DMA

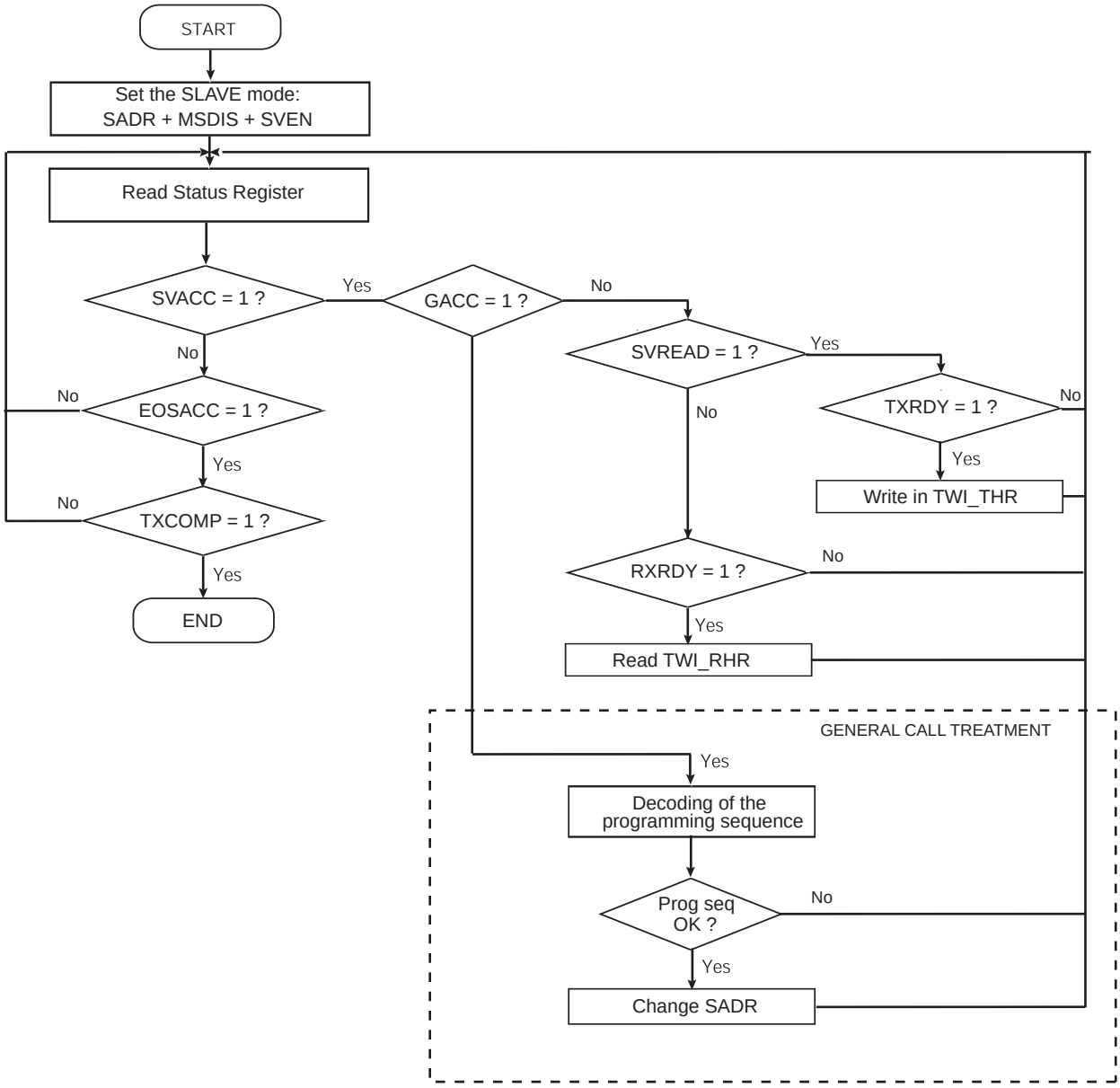
The DMA transfer size must be defined with the buffer size. In Slave mode, the number of characters to be received must be known in order to configure the DMA.

1. Initialize the DMA (channels, memory pointers, size, etc.).
2. Configure the Slave mode.
3. Enable the DMA.
4. Wait for the DMA buffer transfer complete flag.
5. Disable the DMA.
6. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWI_SR.

38.7.5.6 Read Write Flowcharts

The flowchart shown in Figure 38-29 gives an example of read and write operations in Slave mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the Interrupt Enable Register (TWI_IER) be configured first.

Figure 38-29: Read Write Flowchart in Slave Mode



38.7.6 Register Write Protection

To prevent any single software error from corrupting TWI behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the TWI Write Protection Mode Register (TWI_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the TWI Write Protection Status Register (TWI_WPSR) is set and the WPVSRC field shows the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the TWI_WPSR.

The following registers can be write-protected:

- TWI Slave Mode Register
- TWI Clock Waveform Generator Register

© 2017 Microchip Technology Inc.

38.8 Two-wire Interface (TWI) User Interface

Table 38-7: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	TWI_CR	Write-only	–
0x04	Master Mode Register	TWI_MMR	Read/Write	0x00000000
0x08	Slave Mode Register	TWI_SMR	Read/Write	0x00000000
0x0C	Internal Address Register	TWI_IADR	Read/Write	0x00000000
0x10	Clock Waveform Generator Register	TWI_CWGR	Read/Write	0x00000000
0x14–0x1C	Reserved	–	–	–
0x20	Status Register	TWI_SR	Read-only	0x0000F009
0x24	Interrupt Enable Register	TWI_IER	Write-only	–
0x28	Interrupt Disable Register	TWI_IDR	Write-only	–
0x2C	Interrupt Mask Register	TWI_IMR	Read-only	0x00000000
0x30	Receive Holding Register	TWI_RHR	Read-only	0x00000000
0x34	Transmit Holding Register	TWI_THR	Write-only	–
0x38–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	TWI_WPMR	Read/Write	0x00000000
0xE8	Write Protection Status Register	TWI_WPSR	Read-only	0x00000000
0xEC–0xFC	Reserved	–	–	–

Note: All unlisted offset values are considered as “reserved”.

38.8.1 TWI Control Register

Name: TWI_CR

Address:0xF8010000 (0), 0xF8014000 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	SVDIS	SVEN	MSDIS	MSEN	STOP	START

START: Send a START Condition

0: No effect.

1: A frame beginning with a START bit is transmitted according to the features defined in the TWI Master Mode Register (TWI_MMR).

This action is necessary for the TWI to read data from a slave. When configured in Master mode with a write operation, a frame is sent as soon as the user writes a character in the Transmit Holding Register (TWI_THR).

STOP: Send a STOP Condition

0: No effect.

1: STOP condition is sent just after completing the current byte transmission in Master read mode.

- In single data byte master read, the START and STOP must both be set.
- In multiple data bytes master read, the STOP must be set after the last data received but one.
- In Master read mode, if a NACK bit is received, the STOP is automatically performed.
- In multiple data write operation, when both THR and internal shifter are empty, a STOP condition is sent automatically.

MSEN: TWI Master Mode Enabled

0: No effect.

1: Enables the Master mode (MSDIS must be written to 0).

Note: Switching from Slave to Master mode is only permitted when TXCOMP = 1.

MSDIS: TWI Master Mode Disabled

0: No effect.

1: The Master mode is disabled, all pending data is transmitted. The shifter and holding characters (if it contains data) are transmitted in case of write operation. In read operation, the character being transferred must be completely received before disabling.

SVEN: TWI Slave Mode Enabled

0: No effect.

1: Enables the Slave mode (SVDIS must be written to 0)

Note: Switching from master to Slave mode is only permitted when TXCOMP = 1.

© 2017 Microchip Technology Inc.

SVDIS: TWI Slave Mode Disabled

0: No effect.

1: The Slave mode is disabled. The shifter and holding characters (if it contains data) are transmitted in case of read operation. In write operation, the character being transferred must be completely received before disabling.

SWRST: Software Reset

0: No effect.

1: Equivalent to a system reset.

38.8.2 TWI Master Mode Register

Name: TWI_MMR

Address:0xF8010004 (0), 0xF8014004 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	DADR						

15	14	13	12	11	10	9	8
–	–	–	MREAD	–	–	IADRSZ	

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

IADRSZ: Internal Device Address Size

Value	Name	Description
0	NONE	No internal device address
1	1_BYTE	One-byte internal device address
2	2_BYTE	Two-byte internal device address
3	3_BYTE	Three-byte internal device address

MREAD: Master Read Direction

0: Master write direction.

1: Master read direction.

DADR: Device Address

The device address is used to access slave devices in Read or Write mode. These bits are only used in Master mode.

© 2017 Microchip Technology Inc.

38.8.3 TWI Slave Mode Register

Name: TWI_SMR

Address:0xF8010008 (0), 0xF8014008 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	SADR						

15	14	13	12	11	10	9	8
–	–	–	–	–	–		

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in the TWI Write Protection Mode Register.

SADR: Slave Address

The slave device address is used in Slave mode in order to be accessed by master devices in Read or Write mode.

SADR must be programmed before enabling the Slave mode or after a general call. Writes at other times have no effect.

38.8.4 TWI Internal Address Register

Name: TWI_IADR

Address:0xF801000C (0), 0xF801400C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
IADR							

15	14	13	12	11	10	9	8
IADR							

7	6	5	4	3	2	1	0
IADR							

IADR: Internal Address

0, 1, 2 or 3 bytes depending on IADRSZ.

© 2017 Microchip Technology Inc.

38.8.5 TWI Clock Waveform Generator Register

Name: TWI_CWGR

Address:0xF8010010 (0), 0xF8014010 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	CKDIV		

15	14	13	12	11	10	9	8
CHDIV							

7	6	5	4	3	2	1	0
CLDIV							

This register can only be written if the WPEN bit is cleared in the TWI Write Protection Mode Register.

TWI_CWGR is only used in Master mode.

CLDIV: Clock Low Divider

The TWCK low period is defined as follows: $t_{low} = ((CLDIV \times 2^{CKDIV}) + 4 \times t_{peripheral\ clock}$

CHDIV: Clock High Divider

The TWCK high period is defined as follows: $t_{high} = ((CHDIV \times 2^{CKDIV}) + 4 \times t_{peripheral\ clock}$

CKDIV: Clock Divider

The CKDIV field is used to increase both TWCK high and low periods.

38.8.6 TWI Status Register

Name: TWI_SR

Address:0xF8010020 (0), 0xF8014020 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCLWS	ARBLST	NACK

7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	SVREAD	TXRDY	RXRDY	TXCOMP

TXCOMP: Transmission Completed (cleared by writing TWI_THR)

TXCOMP used in Master mode:

0: During the length of the current frame.

1: When both holding register and internal shifter are empty and STOP condition has been sent.

TXCOMP behavior in Master mode can be seen in Figure 38-8.

TXCOMP used in Slave mode:

0: As soon as a START is detected.

1: After a STOP or a REPEATED START + an address different from SADR is detected.

TXCOMP behavior in Slave mode can be seen in Figure 38-25, Figure 38-26, Figure 38-27 and Figure 38-28.

RXRDY: Receive Holding Register Ready (cleared by reading TWI_RHR)

0: No character has been received since the last TWI_RHR read operation.

1: A byte has been received in the TWI_RHR since the last read.

RXRDY behavior in Master mode can be seen in Figure 38-8.

RXRDY behavior in Slave mode can be seen in Figure 38-23, Figure 38-26, Figure 38-27 and Figure 38-28.

TXRDY: Transmit Holding Register Ready (cleared by writing TWI_THR)

TXRDY used in Master mode:

0: The transmit holding register has not been transferred into internal shifter. Set to 0 when writing into TWI_THR.

1: As soon as a data byte is transferred from TWI_THR to internal shifter or if a NACK error is detected, TXRDY is set at the same time as TXCOMP and NACK. TXRDY is also set when MSEN is set (enable TWI).

TXRDY behavior in Master mode can be seen in Figure 38.7.3.3.

TXRDY used in Slave mode:

0: As soon as data is written in the TWI_THR, until this data has been transmitted and acknowledged (ACK or NACK).

1: It indicates that the TWI_THR is empty and that data has been transmitted and acknowledged.

If TXRDY is high and if a NACK has been detected, the transmission will be stopped. Thus when TRDY = NACK = 1, the programmer must not fill TWI_THR to avoid losing it.

TXRDY behavior in Slave mode can be seen in Figure 38-22, Figure 38-25, Figure 38-27 and Figure 38-28.

© 2017 Microchip Technology Inc.

SVREAD: Slave Read

This bit is only used in Slave mode. When SVACC is low (no Slave access has been detected) SVREAD is irrelevant.

0: Indicates that a write access is performed by a Master.

1: Indicates that a read access is performed by a Master.

SVREAD behavior can be seen in Figure 38-22, Figure 38-23, Figure 38-27 and Figure 38-28.

SVACC: Slave Access

This bit is only used in Slave mode.

0: TWI is not addressed. SVACC is automatically cleared after a NACK or a STOP condition is detected.

1: Indicates that the address decoding sequence has matched (A Master has sent SADR). SVACC remains high until a NACK or a STOP condition is detected.

SVACC behavior can be seen in Figure 38-22, Figure 38-23, Figure 38-27 and Figure 38-28.

GACC: General Call Access (cleared on read)

This bit is only used in Slave mode.

0: No General Call has been detected.

1: A General Call has been detected. After the detection of General Call, if need be, the programmer may acknowledge this access and decode the following bytes and respond according to the value of the bytes.

GACC behavior can be seen in Figure 38-24.

OVRE: Overrun Error (cleared on read)

This bit is only used in Master mode.

0: TWI_RHR has not been loaded while RXRDY was set

1: TWI_RHR has been loaded while RXRDY was set. Reset by read in TWI_SR when TXCOMP is set.

NACK: Not Acknowledged (cleared on read)

NACK used in Master mode:

0: Each data byte has been correctly received by the far-end side TWI slave component.

1: A data byte or an address byte has not been acknowledged by the slave component. Set at the same time as TXCOMP.

NACK used in Slave Read mode:

0: Each data byte has been correctly received by the Master.

1: In Read mode, a data byte has not been acknowledged by the Master. When NACK is set, the programmer must not fill TWI_THR even if TXRDY is set, because that means that the Master will stop the data transfer or reinitiate it.

Note that in Slave write mode all data are acknowledged by the TWI.

ARBLST: Arbitration Lost (cleared on read)

This bit is only used in Master mode.

0: Arbitration won.

1: Arbitration lost. Another master of the TWI bus has won the multi-master arbitration. TXCOMP is set at the same time.

SCLWS: Clock Wait State

This bit is only used in Slave mode.

0: The clock is not stretched.

1: The clock is stretched. TWI_THR / TWI_RHR buffer is not filled / emptied before transmission / reception of a new character.

SCLWS behavior can be seen in Figure 38-25 and Figure 38-26.

© 2017 Microchip Technology Inc.

EOSACC: End Of Slave Access (cleared on read)

This bit is only used in Slave mode.

0: A slave access is being performed.

1: The Slave access is finished. End Of Slave Access is automatically set as soon as SVACC is reset.

EOSACC behavior can be seen in Figure 38-27 and Figure 38-28.

© 2017 Microchip Technology Inc.

38.8.7 TWI Interrupt Enable Register

Name: TWI_IER

Address:0xF8010024 (0), 0xF8014024 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

TXCOMP: Transmission Completed Interrupt Enable

RXRDY: Receive Holding Register Ready Interrupt Enable

TXRDY: Transmit Holding Register Ready Interrupt Enable

SVACC: Slave Access Interrupt Enable

GACC: General Call Access Interrupt Enable

OVRE: Overrun Error Interrupt Enable

NACK: Not Acknowledge Interrupt Enable

ARBLST: Arbitration Lost Interrupt Enable

SCL_WS: Clock Wait State Interrupt Enable

EOSACC: End Of Slave Access Interrupt Enable

© 2017 Microchip Technology Inc.

38.8.8 TWI Interrupt Disable Register

Name: TWI_IDR

Address:0xF8010028 (0), 0xF8014028 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

TXCOMP: Transmission Completed Interrupt Disable

RXRDY: Receive Holding Register Ready Interrupt Disable

TXRDY: Transmit Holding Register Ready Interrupt Disable

SVACC: Slave Access Interrupt Disable

GACC: General Call Access Interrupt Disable

OVRE: Overrun Error Interrupt Disable

NACK: Not Acknowledge Interrupt Disable

ARBLST: Arbitration Lost Interrupt Disable

SCL_WS: Clock Wait State Interrupt Disable

EOSACC: End Of Slave Access Interrupt Disable

© 2017 Microchip Technology Inc.

38.8.9 TWI Interrupt Mask Register

Name: TWI_IMR

Address:0xF801002C (0), 0xF801402C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

TXCOMP: Transmission Completed Interrupt Mask

RXRDY: Receive Holding Register Ready Interrupt Mask

TXRDY: Transmit Holding Register Ready Interrupt Mask

SVACC: Slave Access Interrupt Mask

GACC: General Call Access Interrupt Mask

OVRE: Overrun Error Interrupt Mask

NACK: Not Acknowledge Interrupt Mask

ARBLST: Arbitration Lost Interrupt Mask

SCL_WS: Clock Wait State Interrupt Mask

EOSACC: End Of Slave Access Interrupt Mask

38.8.10 TWI Receive Holding Register

Name: TWI_RHR

Address:0xF8010030 (0), 0xF8014030 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
RXDATA							

RXDATA: Master or Slave Receive Holding Data

© 2017 Microchip Technology Inc.

38.8.11 TWI Transmit Holding Register

Name: TWI_THR

Address:0xF8010034 (0), 0xF8014034 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
TXDATA							

TXDATA: Master or Slave Transmit Holding Data

38.8.12 TWI Write Protection Mode Register

Name:TWI_WPMR

Address:0xF80100E4 (0), 0xF80140E4 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

WPEN: Write Protection Enable

0: Disables the write protection if WPKEY corresponds to 0x545749 (“TWI” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x545749 (“TWI” in ASCII).

See Section 38.7.6 “Register Write Protection” for the list of registers that can be write-protected.

WPKEY: Write Protection Key

Value	Name	Description
0x545749	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0

© 2017 Microchip Technology Inc.

38.8.13 TWI Write Protection Status Register

Name:TWI_WPSR

Address:0xF80100E8 (0), 0xF80140E8 (1)

Access: Read-only

31	30	29	28	27	26	25	24
WPVSRC							
23	22	21	20	19	18	17	16
WPVSRC							
15	14	13	12	11	10	9	8
WPVSRC							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

WPVS: Write Protection Violation Status

0: No write protection violation has occurred since the last read of the TWI_WPSR.

1: A write protection violation has occurred since the last read of the TWI_WPSR. If this violation is an unauthorized attempt to write a protected register, the violation is reported into field WPVSRC.

WPVSRC: Write Protection Violation Source

When WPVS = 1, WPVSRC shows the register address offset at which a write access has been attempted.

39. Universal Synchronous Asynchronous Receiver Transmitter (USART)

39.1 Description

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multidrop communications are also supported through address bit handling in reception and transmission.

The USART features three test modes: Remote loopback, Local loopback and Automatic echo.

The USART supports specific operating modes providing interfaces on RS485, LIN, and SPI buses, with ISO7816 T = 0 or T = 1 smart card slots and infrared transceivers. The hardware handshaking feature enables an out-of-band flow control by automatic management of the pins RTS and CTS.

The USART supports the connection to the DMA Controller, which enables data transfers to the transmitter and from the receiver. The DMAC provides chained buffer management without any intervention of the processor.

39.2 Embedded Characteristics

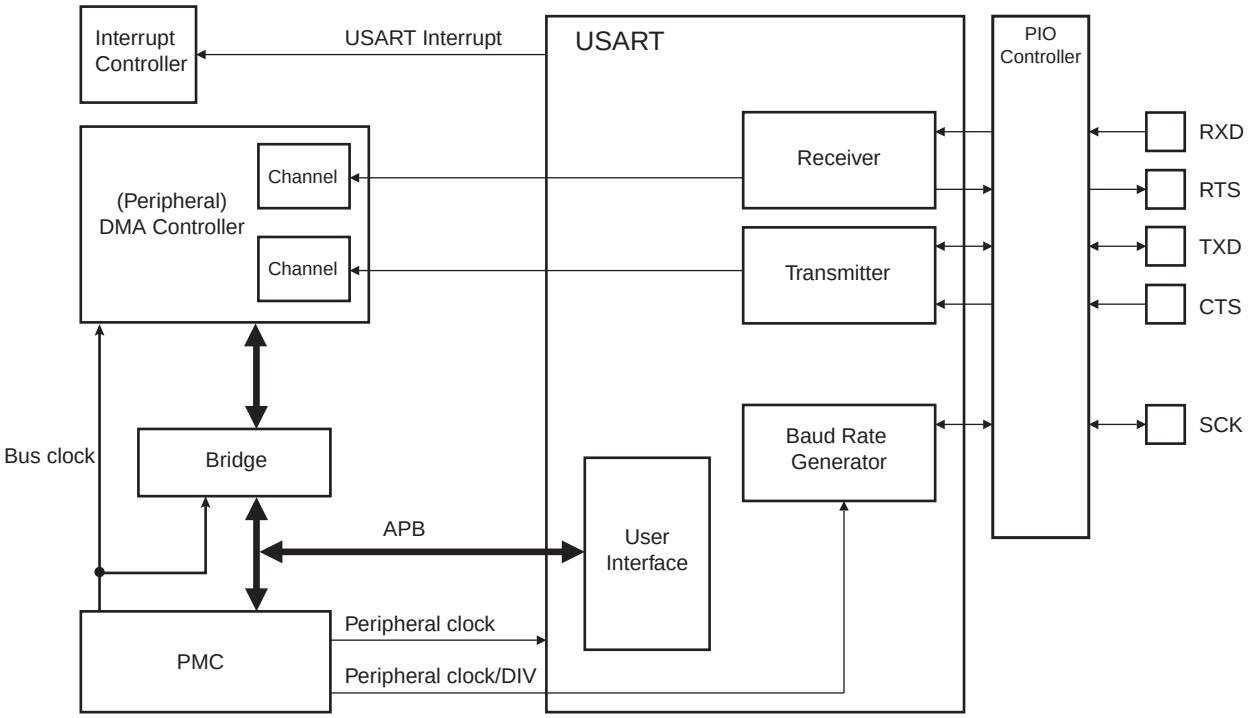
- Programmable Baud Rate Generator
- 5- to 9-bit Full-duplex Synchronous or Asynchronous Serial Communications
  - 1, 1.5 or 2 Stop Bits in Asynchronous Mode or 1 or 2 Stop Bits in Synchronous Mode
  - Parity Generation and Error Detection
  - Framing Error Detection, Overrun Error Detection
  - Digital Filter on Receive Line
  - MSB- or LSB-first
  - Optional Break Generation and Detection
  - By 8 or by 16 Oversampling Receiver Frequency
  - Optional Hardware Handshaking RTS-CTS
  - Receiver Time-out and Transmitter Timeguard
  - Optional Multidrop Mode with Address Generation and Detection
- RS485 with Driver Control Signal
- ISO7816, T = 0 or T = 1 Protocols for Interfacing with Smart Cards
  - NACK Handling, Error Counter with Repetition and Iteration Limit
- IrDA Modulation and Demodulation
  - Communication at up to 115.2 kbit/s
- SPI Mode
  - Master or Slave
  - Serial Clock Programmable Phase and Polarity
  - SPI Serial Clock (SCK) Frequency up to $f_{peripheral\ clock}/6$
- LIN Mode
  - Compliant with LIN 1.3 and LIN 2.0 SPECIFICATIONS
  - Master or Slave
  - Processing of Frames with Up to 256 Data Bytes
  - Response Data Length can be Configurable or Defined Automatically by the Identifier
  - Self-synchronization in Slave Node Configuration
  - Automatic Processing and Verification of the “Synch Break” and the “Synch Field”
  - “Synch Break” Detection Even When Partially Superimposed with a Data Byte
  - Automatic Identifier Parity Calculation/Sending and Verification
  - Parity Sending and Verification Can be Disabled
  - Automatic Checksum Calculation/sending and Verification
  - Checksum Sending and Verification Can be Disabled
  - Support Both “Classic” and “Enhanced” Checksum Types
  - Full LIN Error Checking and Reporting
  - Frame Slot Mode: Master Allocates Slots to the Scheduled Frames Automatically
  - Generation of the Wakeup Signal

© 2017 Microchip Technology Inc.

- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo
- Supports Connection of:
  - Two DMA Controller Channels (DMAC)
- Offers Buffer Transfer without Processor Intervention
- Register Write Protection

39.3 Block Diagram

Figure 39-1: USART Block Diagram



39.4 I/O Lines Description

Table 39-1: I/O Line Description

Name	Description	Type	Active Level
SCK	Serial Clock	I/O	—
TXD	Transmit Serial Data or Master Out Slave In (MOSI) in SPI master mode or Master In Slave Out (MISO) in SPI slave mode	I/O	—
RXD	Receive Serial Data or Master In Slave Out (MISO) in SPI master mode or Master Out Slave In (MOSI) in SPI slave mode	Input	—
CTS	Clear to Send or Slave Select (NSS) in SPI slave mode	Input	Low
RTS	Request to Send or Slave Select (NSS) in SPI master mode	Output	Low

© 2017 Microchip Technology Inc.

39.5 Product Dependencies

39.5.1 I/O Lines

The pins used for interfacing the USART may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

Table 39-2: I/O Lines

Instance	Signal	I/O Line	Peripheral
USART0	CTS0	PA3	A
USART0	RTS0	PA2	A
USART0	RXD0	PA1	A
USART0	SCK0	PA4	A
USART0	TXD0	PA0	A
USART1	CTS1	PC28	C
USART1	RTS1	PC27	C
USART1	RXD1	PA6	A
USART1	SCK1	PC29	C
USART1	TXD1	PA5	A
USART2	CTS2	PB1	B
USART2	RTS2	PB0	B
USART2	RXD2	PA8	A
USART2	SCK2	PB2	B
USART2	TXD2	PA7	A
USART3	CTS3	PC25	B
USART3	RTS3	PC24	B
USART3	RXD3	PC23	B
USART3	SCK3	PC26	B
USART3	TXD3	PC22	B

39.5.2 Power Management

The USART is not continuously clocked. The programmer must first enable the USART clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off.

39.5.3 Interrupt Sources

The USART interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the USART interrupt requires the Interrupt Controller to be programmed first.

Table 39-3: Peripheral IDs

Instance	ID
USART0	5

Table 39-3: Peripheral IDs

USART1	6
USART2	7
USART3	8

39.6 Functional Description

39.6.1 Baud Rate Generator

The baud rate generator provides the bit period clock, also named the baud rate clock, to both the receiver and the transmitter.

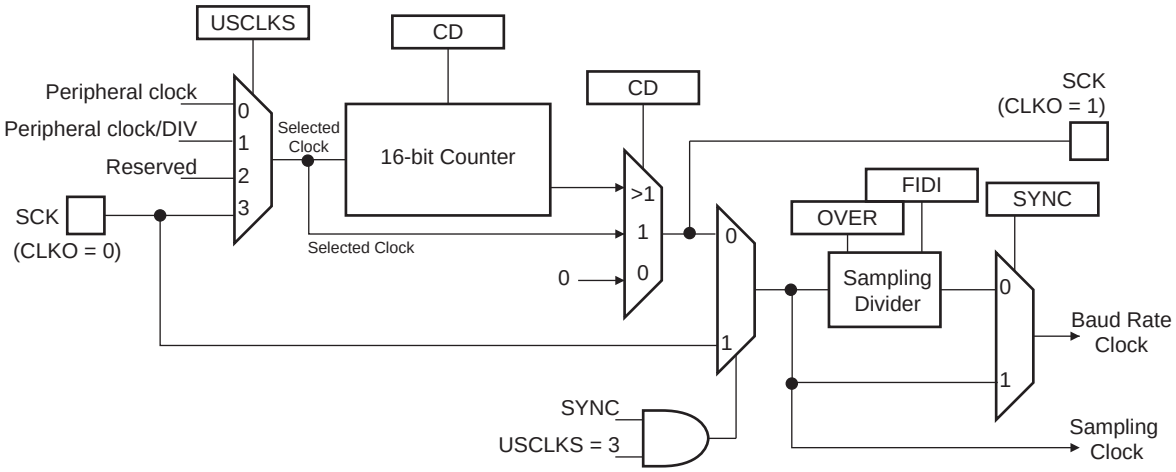
The baud rate generator clock source is selected by configuring the USCLKS field in the USART Mode Register (US_MR) to one of the following:

- The peripheral clock
- A division of the peripheral clock, where the divider is product-dependent, but generally set to 8
- The external clock, available on the SCK pin

The baud rate generator is based upon a 16-bit divider, which is programmed with the CD field of the Baud Rate Generator register (US_BRGR). If a 0 is written to CD, the baud rate generator does not generate any clock. If a 1 is written to CD, the divider is bypassed and becomes inactive.

If the external SCK clock is selected, the duration of the low and high levels of the signal provided on the SCK pin must be longer than a peripheral clock period. The frequency of the signal provided on SCK must be at least 3 times lower than the frequency provided on the peripheral clock in USART mode (field USART_MODE differs from 0xE or 0xF), or 6 times lower in SPI mode (field USART_MODE equals 0xE or 0xF).

Figure 39-2: Baud Rate Generator



39.6.1.1 Baud Rate in Asynchronous Mode

If the USART is programmed to operate in Asynchronous mode, the selected clock is first divided by CD, which is field programmed in the US_BRGR. The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on how the OVER bit in the US_MR is programmed.

If OVER is set, the receiver sampling is eight times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The baud rate is calculated as per the following formula:

$$\text{Baud Rate} = \frac{\text{Selected Clock}}{(8(2-OVER)CD)}$$

This gives a maximum baud rate of peripheral clock divided by 8, assuming that the peripheral clock is the highest possible clock and that the OVER bit is set.

© 2017 Microchip Technology Inc.

Baud Rate Calculation Example

Table 39-4 shows calculations of CD to obtain a baud rate at 38,400 bit/s for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

Table 39-4: Baud Rate Example (OVER = 0)

Source Clock (MHz)	Expected Baud Rate (bit/s)	Calculation Result	CD	Actual Baud Rate (bit/s)	Error
3,686,400	38,400	6.00	6	38,400.00	0.00%
4,915,200	38,400	8.00	8	38,400.00	0.00%
5,000,000	38,400	8.14	8	39,062.50	1.70%
7,372,800	38,400	12.00	12	38,400.00	0.00%
8,000,000	38,400	13.02	13	38,461.54	0.16%
12,000,000	38,400	19.53	20	37,500.00	2.40%
12,288,000	38,400	20.00	20	38,400.00	0.00%
14,318,180	38,400	23.30	23	38,908.10	1.31%
14,745,600	38,400	24.00	24	38,400.00	0.00%
18,432,000	38,400	30.00	30	38,400.00	0.00%
24,000,000	38,400	39.06	39	38,461.54	0.16%
24,576,000	38,400	40.00	40	38,400.00	0.00%
25,000,000	38,400	40.69	40	38,109.76	0.76%
32,000,000	38,400	52.08	52	38,461.54	0.16%
32,768,000	38,400	53.33	53	38,641.51	0.63%
33,000,000	38,400	53.71	54	38,194.44	0.54%
40,000,000	38,400	65.10	65	38,461.54	0.16%
50,000,000	38,400	81.38	81	38,580.25	0.47%

In this example, the baud rate is calculated with the following formula:

$$\text{Baud Rate} = \text{Selected Clock} / CD \times 16$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left(\frac{\text{Expected Baud Rate}}{\text{Actual Baud Rate}}\right)$$

39.6.1.2 Fractional Baud Rate in Asynchronous Mode

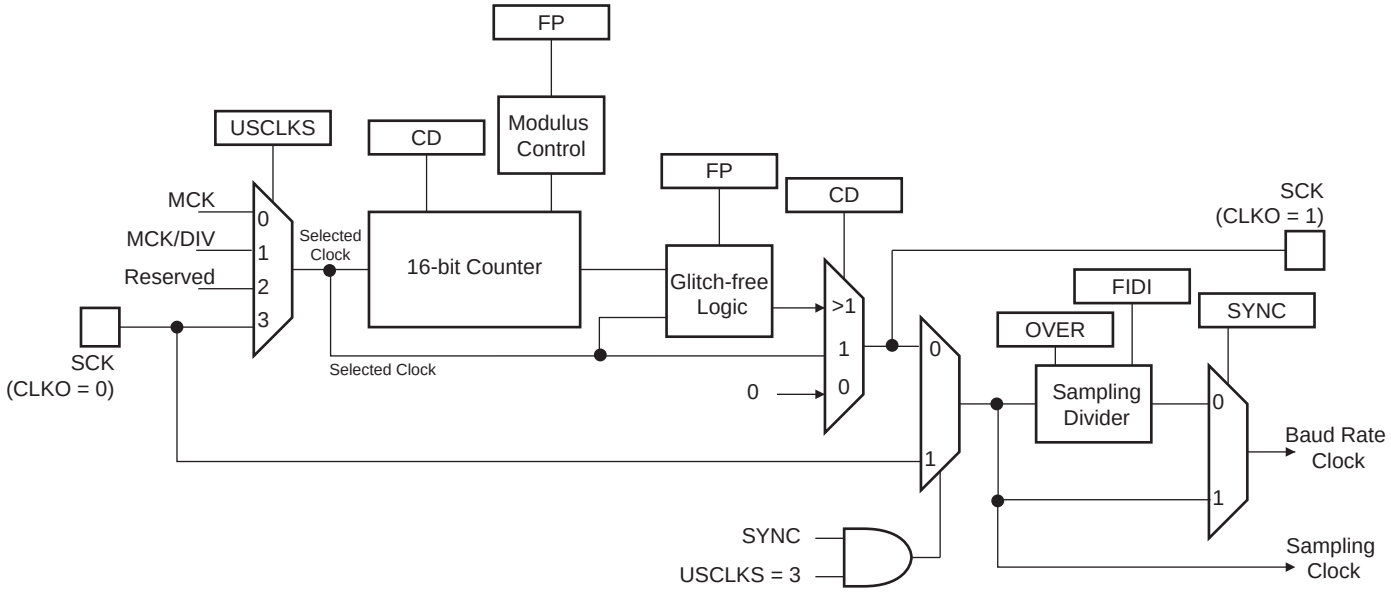
The baud rate generator is subject to the following limitation: the output frequency changes only by integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified to obtain baud rate changes by a fraction of the reference source clock. This fractional part is programmed with the FP field in the US_BRGR. If FP is not 0, the fractional part is activated. The resolution is one eighth of the clock divider. This feature is only available when using USART normal mode. The fractional baud rate is calculated using the following formula:

$$\text{Baud Rate} = \frac{\text{Selected Clock}}{\left(8(2 - OVER)\left(CD + \frac{FP}{8}\right)\right)}$$

The modified architecture is presented in the following Figure 39-3.

© 2017 Microchip Technology Inc.

Figure 39-3: Fractional Baud Rate Generator



Warning: When the value of field FP is greater than 0, the SCK (oversampling clock) generates non-constant duty cycles. The SCK high duration is increased by “selected clock” period from time to time. The duty cycle depends on the value of the CD field.

39.6.1.3 Baud Rate in Synchronous Mode or SPI Mode

If the USART is programmed to operate in Synchronous mode, the selected clock is simply divided by the field CD in the US_BRGR.

$$\text{Baud Rate} = \frac{\text{Selected Clock}}{CD}$$

In Synchronous mode, if the external clock is selected (USCLKS = 3), the clock is provided directly by the signal on the USART SCK pin. No division is active. The value written in US_BRGR has no effect. The external clock frequency must be at least 3 times lower than the system clock. In Synchronous mode master (USCLKS = 0 or 1, CLKO set to 1), the receive part limits the SCK maximum frequency to Selected Clock/3 in USART mode, or Selected Clock/6 in SPI mode.

When either the external clock SCK or the internal clock divided (peripheral clock/DIV) is selected, the value programmed in CD must be even if the user has to ensure a 50:50 mark/space ratio on the SCK pin. When the peripheral clock is selected, the baud rate generator ensures a 50:50 duty cycle on the SCK pin, even if the value programmed in CD is odd.

39.6.1.4 Baud Rate in ISO 7816 Mode

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{Di}{Fi} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in Table 39-5.

Table 39-5: Binary and Decimal Values for Di

DI field	0001	0010	0011	0100	0101	0110	1000	1001
Di (decimal)	1	2	4	8	16	32	12	20

© 2017 Microchip Technology Inc.

Fi is a binary value encoded on a 4-bit field, named FI, as represented in Table 39-6.

Table 39-6: Binary and Decimal Values for Fi

FI field	0000	0001	0010	0011	0100	0101	0110	1001	1010	1011	1100	1101
Fi (decimal)	372	372	558	744	1116	1488	1860	512	768	1024	1536	2048

Table 39-7 shows the resulting Fi/Di ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 39-7: Possible Values for the Fi/Di Ratio

Fi/Di	372	558	744	1116	1488	1806	512	768	1024	1536	2048
1	372	558	744	1116	1488	1860	512	768	1024	1536	2048
2	186	279	372	558	744	930	256	384	512	768	1024
4	93	139.5	186	279	372	465	128	192	256	384	512
8	46.5	69.75	93	139.5	186	232.5	64	96	128	192	256
16	23.25	34.87	46.5	69.75	93	116.2	32	48	64	96	128
32	11.62	17.43	23.25	34.87	46.5	58.13	16	24	32	48	64
12	31	46.5	62	93	124	155	42.66	64	85.33	128	170.6
20	18.6	27.9	37.2	55.8	74.4	93	25.6	38.4	51.2	76.8	102.4

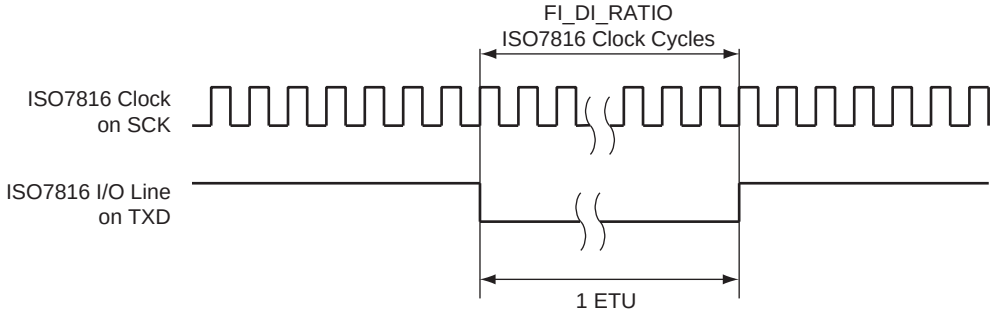
If the USART is configured in ISO7816 mode, the clock selected by the USCLKS field in US_MR is first divided by the value programmed in the field CD in the US_BRGR. The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in US_MR.

This clock is then divided by the value programmed in the FI_DI_RATIO field in the FI_DI_Ratio register (US_FIDI). This is performed by the Sampling Divider, which performs a division by up to 2047 in ISO7816 mode. The non-integer values of the Fi/Di Ratio are not supported and the user must program the FI_DI_RATIO field to a value as close as possible to the expected value.

The FI_DI_RATIO field resets to the value 0x174 (372 in decimal) and is the most common divider between the ISO7816 clock and the bit rate (Fi = 372, Di = 1).

Figure 39-4 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 39-4: Elementary Time Unit (ETU)



39.6.2 Receiver and Transmitter Control

After reset, the receiver is disabled. The user must enable the receiver by setting the RXEN bit in the Control register (US_CR). However, the receiver registers can be programmed before the receiver clock is enabled.

After reset, the transmitter is disabled. The user must enable it by setting the TXEN bit in the US_CR. However, the transmitter registers can be programmed before being enabled.

The receiver and the transmitter can be enabled together or independently.

© 2017 Microchip Technology Inc.

At any time, the software can perform a reset on the receiver or the transmitter of the USART by setting the corresponding bit, RSTRX and RSTTX respectively, in the US_CR. The software resets clear the status flag and reset internal state machines but the user interface configuration registers hold the value configured prior to software reset. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in the US_CR. If the receiver is disabled during a character reception, the USART waits until the end of reception of the current character, then the reception is stopped. If the transmitter is disabled while it is operating, the USART waits the end of transmission of both the current character and character being stored in the Transmit Holding register (US_THR). If a timeguard is programmed, it is handled normally.

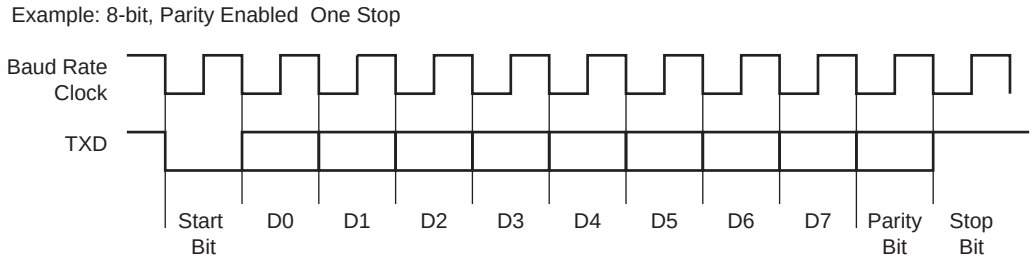
39.6.3 Synchronous and Asynchronous Modes

39.6.3.1 Transmitter Operations

The transmitter performs the same in both Synchronous and Asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, one optional parity bit and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE 9 bit in US_MR. Nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The parity bit is set according to the PAR field in US_MR. The even, odd, space, marked or none parity bit can be configured. The MSBF field in the US_MR configures which data bit is sent first. If written to 1, the most significant bit is sent first. If written to 0, the less significant bit is sent first. The number of stop bits is selected by the NBSTOP field in the US_MR. The 1.5 stop bit is supported in Asynchronous mode only.

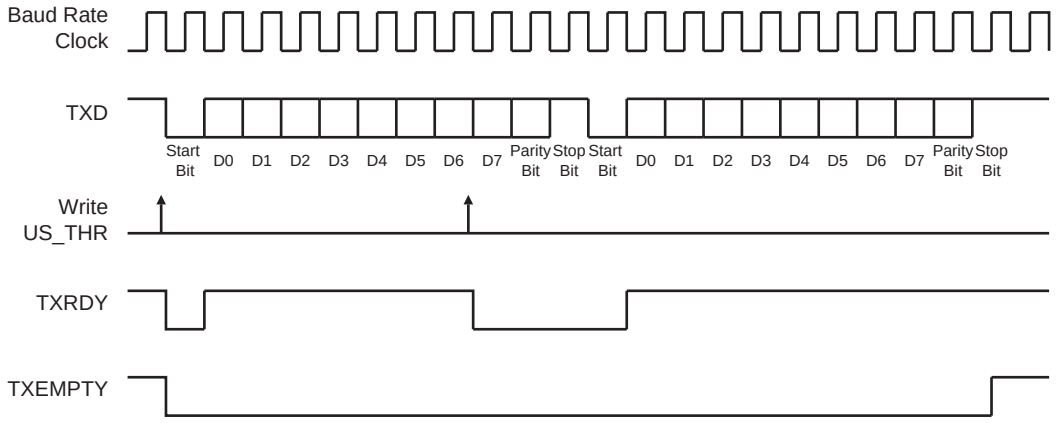
Figure 39-5: Character Transmit



The characters are sent by writing in the Transmit Holding register (US_THR). The transmitter reports two status bits in the Channel Status register (US_CSR): TXRDY (Transmitter Ready), which indicates that US_THR is empty and TXEMPTY, which indicates that all the characters written in US_THR have been processed. When the current character processing is completed, the last character written in US_THR is transferred into the Shift register of the transmitter and US_THR becomes empty, thus TXRDY rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in US_THR while TXRDY is low has no effect and the written character is lost.

Figure 39-6: Transmitter Status

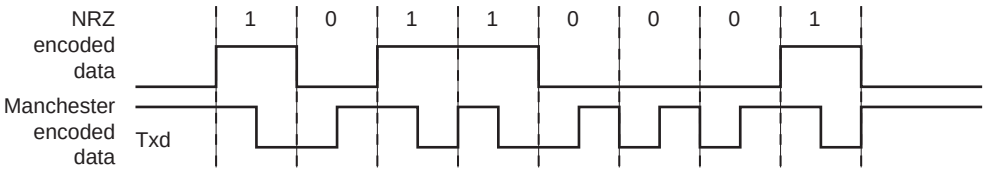


© 2017 Microchip Technology Inc.

39.6.3.2 Manchester Encoder

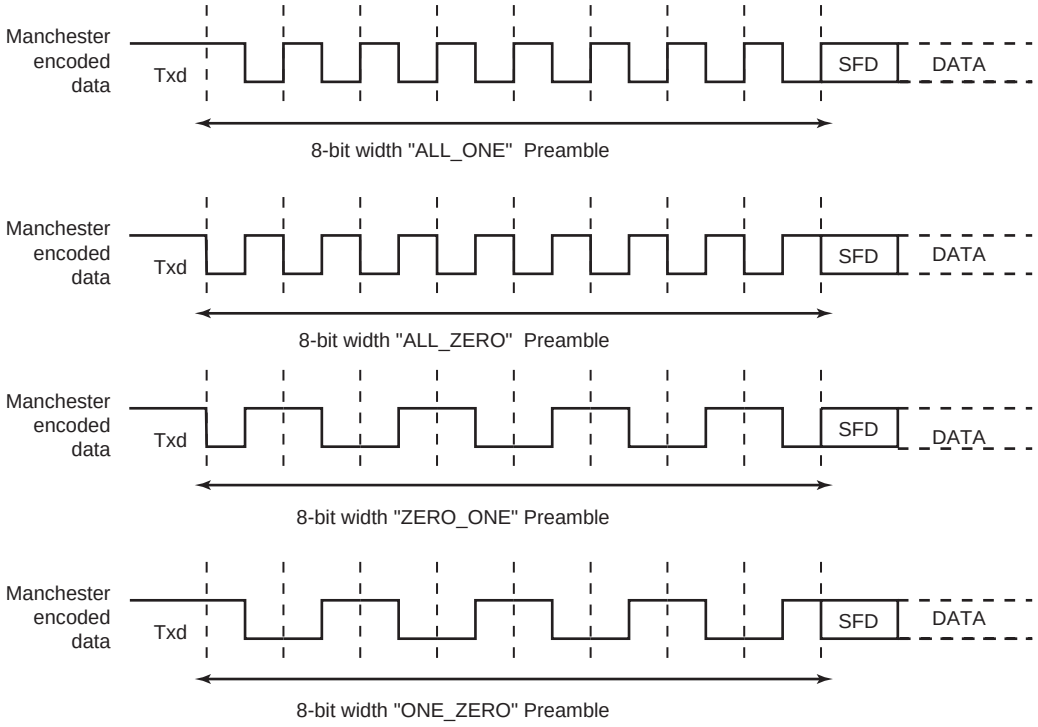
When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN bit in the US_MR to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero or zero-to-one. Thus, a transition always occurs at the midpoint of each bit time. It consumes more bandwidth than the original NRZ signal (2x) but the receiver has more error control since the expected input must show a change at the center of a bit cell. An example of Manchester encoded sequence is: the byte 0xB1 or 10110001 encodes to 10 01 10 10 01 01 01 10, assuming the default polarity of the encoder. Figure 39-7 illustrates this coding scheme.

Figure 39-7: NRZ to Manchester Encoding



The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a predefined pattern with a programmable length from 1 to 15 bit times. If the preamble length is set to 0, the preamble waveform is not generated prior to any character. The preamble pattern is chosen among the following sequences: ALL_ONE, ALL_ZERO, ONE_ZERO or ZERO_ONE, writing the field TX_PP in the US_MAN register, the field TX_PL is used to configure the preamble length. Figure 39-8 illustrates and defines the valid patterns. To improve flexibility, the encoding scheme can be configured using the TX_MPOL field in the US_MAN register. If the TX_MPOL field is set to zero (default), a logic zero is encoded with a zero-to-one transition and a logic one is encoded with a one-to-zero transition. If the TX_MPOL field is set to 1, a logic one is encoded with a one-to-zero transition and a logic zero is encoded with a zero-to-one transition.

Figure 39-8: Preamble Patterns, Default Polarity Assumed

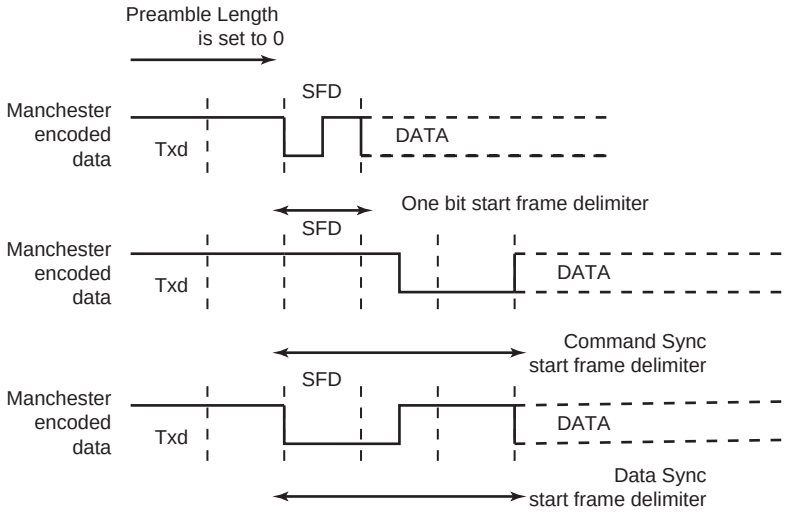


A start frame delimiter is to be configured using the ONEBIT bit in the US_MR. It consists of a user-defined pattern that indicates the beginning of a valid data. Figure 39-9 illustrates these patterns. If the start frame delimiter, also known as the start bit, is one bit, (ONEBIT = 1), a logic zero is Manchester encoded and indicates that a new character is being sent serially on the line. If the start frame delimiter is a synchronization pattern also referred to as sync (ONEBIT to 0), a sequence of three bit times is sent serially on the line to indicate the start of a new character. The sync waveform is in itself an invalid Manchester waveform as the transition occurs at the middle of the second bit time. Two distinct sync patterns are used: the command sync and the data sync. The command sync has a logic one level for one and a

© 2017 Microchip Technology Inc.

half bit times, then a transition to logic zero for the second one and a half bit times. If the MODSYNC bit in the US_MR is set to 1, the next character is a command. If it is set to 0, the next character is a data. When direct memory access is used, the MODSYNC field can be immediately updated with a modified character located in memory. To enable this mode, VAR_SYNC bit in US_MR must be set to 1. In this case, the MODSYNC bit in the US_MR is bypassed and the sync configuration is held in the TXSYNH in the US_THR. The USART character format is modified and includes sync information.

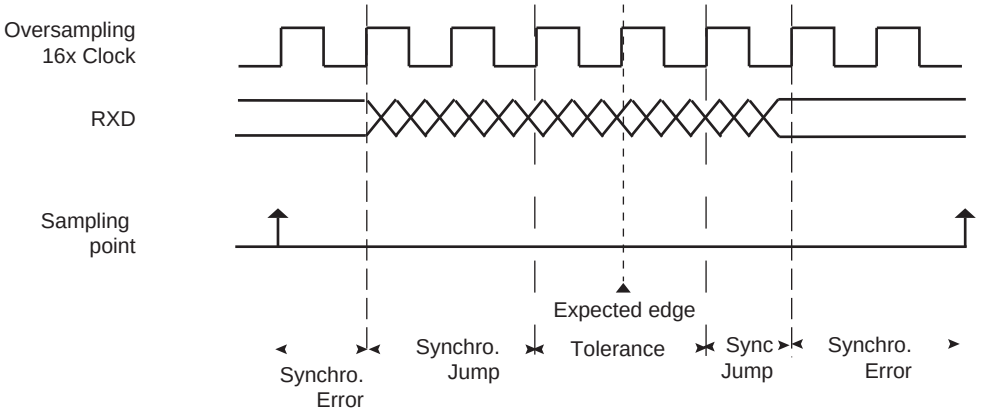
Figure 39-9: Start Frame Delimiter



Drift Compensation

Drift compensation is available only in 16X oversampling mode. An hardware recovery system allows a larger clock drift. To enable the hardware system, the bit in the USART_MAN register must be set. If the RXD edge is one 16X clock cycle from the expected edge, this is considered as normal jitter and no corrective actions is taken. If the RXD event is between 4 and 2 clock cycles before the expected edge, then the current period is shortened by one clock cycle. If the RXD event is between 2 and 3 clock cycles after the expected edge, then the current period is lengthened by one clock cycle. These intervals are considered to be drift and so corrective actions are automatically taken.

Figure 39-10: Bit Resynchronization



39.6.3.3 Asynchronous Receiver

If the USART is programmed in Asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the baud rate clock, depending on the OVER bit in the US_MR.

The receiver samples the RXD line. If the line is sampled during one half of a bit time to 0, a start bit is detected and data, parity and stop bits are successively sampled on the bit rate clock.

© 2017 Microchip Technology Inc.

If the oversampling is 16 (OVER = 0), a start is detected at the eighth sample to 0. Data bits, parity bit and stop bit are assumed to have a duration corresponding to 16 oversampling clock cycles. If the oversampling is 8 (OVER = 1), a start bit is detected at the fourth sample to 0. Data bits, parity bit and stop bit are assumed to have a duration corresponding to 8 oversampling clock cycles.

The number of data bits, first bit sent and Parity mode are selected by the same fields and bits as the transmitter, i.e., respectively CHRL, MODE9, MSBF and PAR. For the synchronization mechanism **only**, the number of stop bits has no effect on the receiver as it considers only one stop bit, regardless of the field NBSTOP, so that resynchronization between the receiver and the transmitter can occur. Moreover, as soon as the stop bit is sampled, the receiver starts looking for a new start bit so that resynchronization can also be accomplished when the transmitter is operating with one stop bit.

Figure 39-11 and Figure 39-12 illustrate start detection and character reception when USART operates in Asynchronous mode.

Figure 39-11: Asynchronous Start Detection

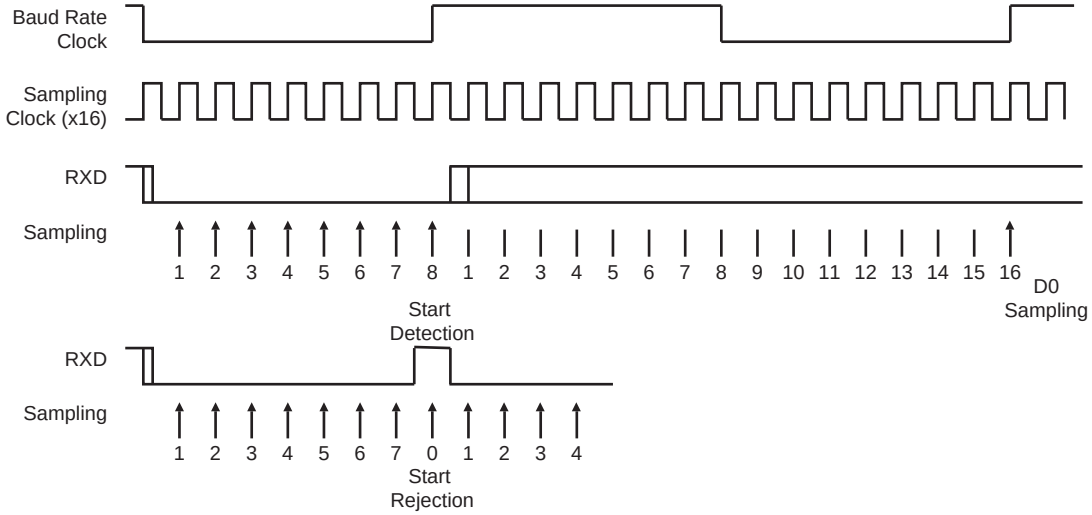
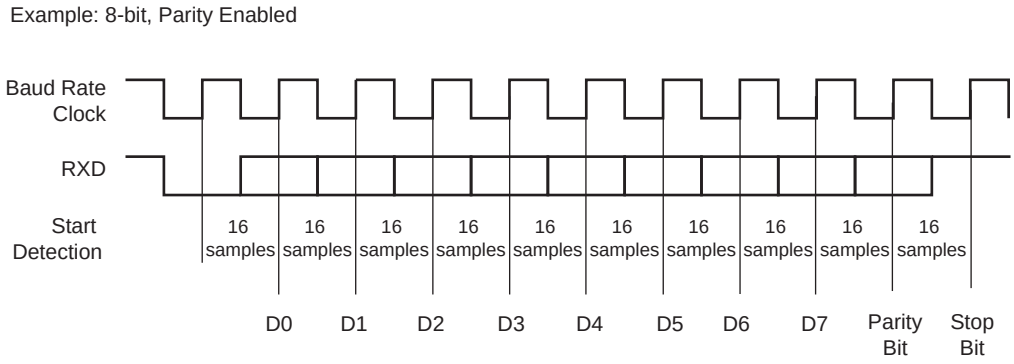


Figure 39-12: Asynchronous Character Reception



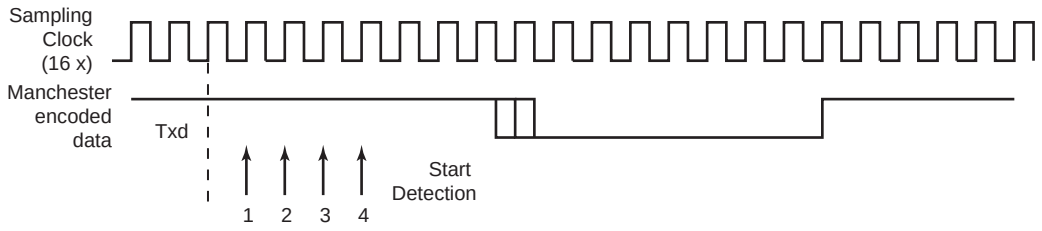
39.6.3.4 Manchester Decoder

When the MAN bit in the US_MR is set to 1, the Manchester decoder is enabled. The decoder performs both preamble and start frame delimiter detection. One input line is dedicated to Manchester encoded input data.

An optional preamble sequence can be defined, its length is user-defined and totally independent of the emitter side. Use RX_PL in US_MAN register to configure the length of the preamble sequence. If the length is set to 0, no preamble is detected and the function is disabled. In addition, the polarity of the input stream is programmable with RX_MPOL bit in US_MAN register. Depending on the desired application the preamble pattern matching is to be defined via the RX_PP field in US_MAN. See Figure 39-8 for available preamble patterns.

Unlike preamble, the start frame delimiter is shared between Manchester Encoder and Decoder. So, if ONEBIT field is set to 1, only a zero encoded Manchester can be detected as a valid start frame delimiter. If ONEBIT is set to 0, only a sync pattern is detected as a valid start frame delimiter. Decoder operates by detecting transition on incoming stream. If RXD is sampled during one quarter of a bit time to zero, a start bit is detected. See Figure 39-13. The sample pulse rejection mechanism applies.

Figure 39-13: Asynchronous Start Bit Detection



The receiver is activated and starts preamble and frame delimiter detection, sampling the data at one quarter and then three quarters. If a valid preamble pattern or start frame delimiter is detected, the receiver continues decoding with the same synchronization. If the stream does not match a valid pattern or a valid start frame delimiter, the receiver resynchronizes on the next valid edge.The minimum time threshold to estimate the bit value is three quarters of a bit time.

If a valid preamble (if used) followed with a valid start frame delimiter is detected, the incoming stream is decoded into NRZ data and passed to USART for processing. Figure 39-14 illustrates Manchester pattern mismatch. When incoming data stream is passed to the USART, the receiver is also able to detect Manchester code violation. A code violation is a lack of transition in the middle of a bit cell. In this case, MANE flag in the US_CSR is raised. It is cleared by writing a 1 to the RSTSTA in the US_CR. See Figure 39-15 for an example of Manchester error detection during data phase.

Figure 39-14: Preamble Pattern Mismatch

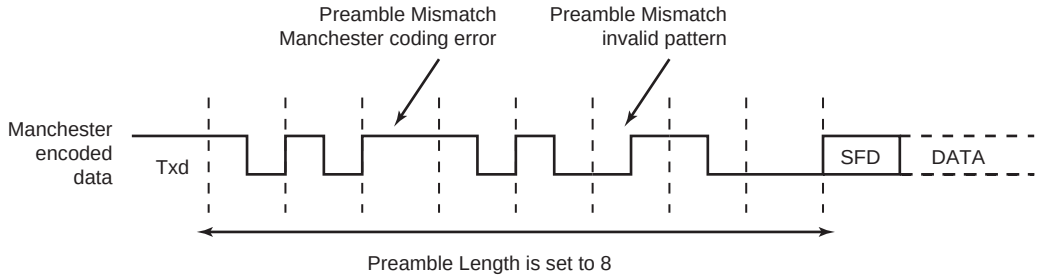
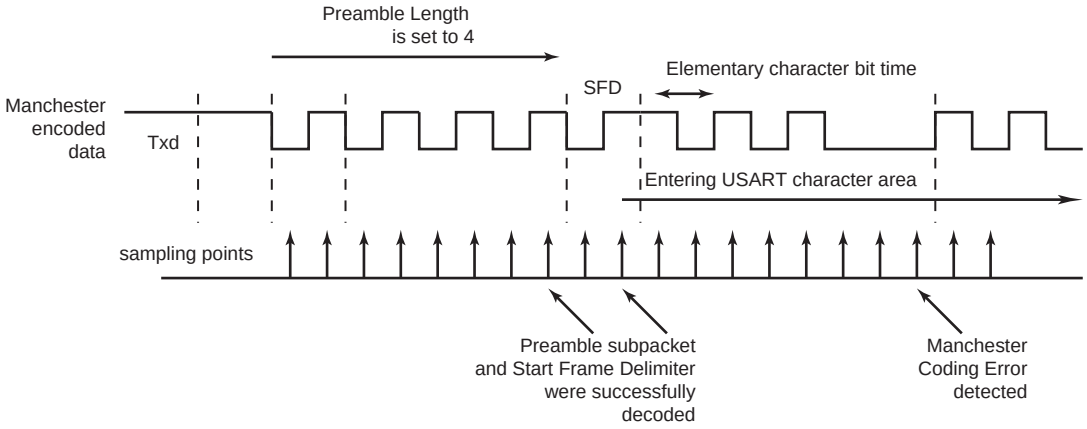


Figure 39-15: Manchester Error Flag



When the start frame delimiter is a sync pattern (ONEBIT field to 0), both command and data delimiter are supported. If a valid sync is detected, the received character is written as RXCHR field in the US_RHR and the RXSYNH is updated. RXCHR is set to 1 when the received character is a command, and it is set to 0 if the received character is a data. This mechanism alleviates and simplifies the direct memory access as the character contains its own sync field in the same register.

As the decoder is setup to be used in Unipolar mode, the first bit of the frame has to be a zero-to-one transition.

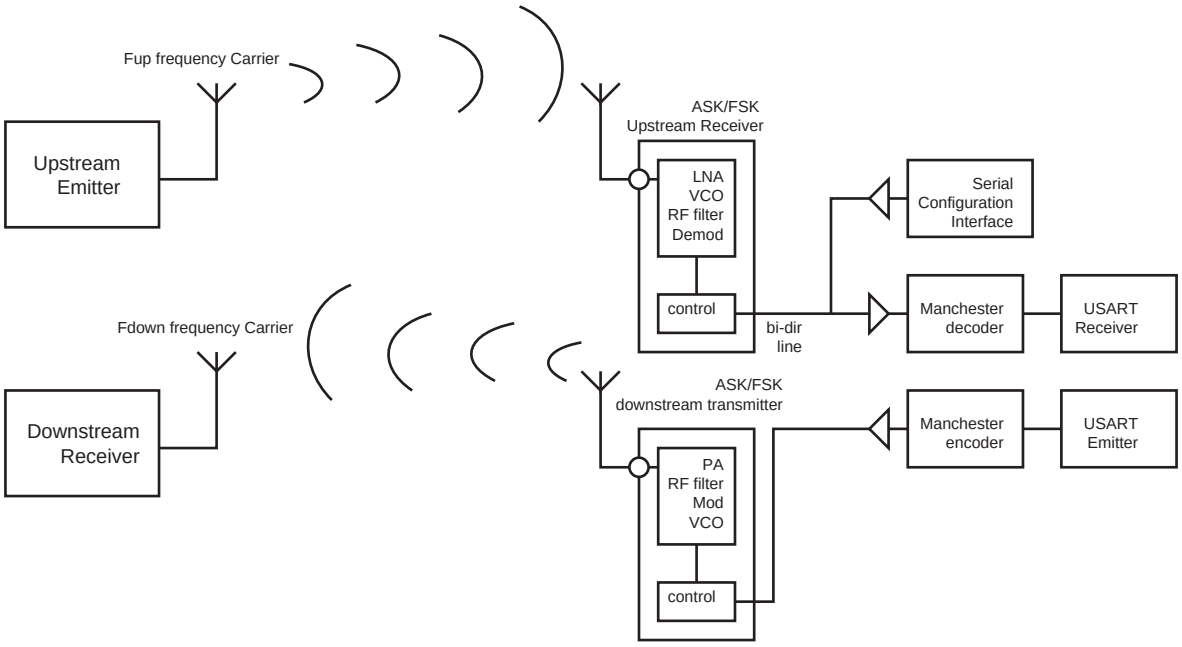
© 2017 Microchip Technology Inc.

39.6.3.5 Radio Interface: Manchester Encoded USART Application

This section describes low data rate RF transmission systems and their integration with a Manchester encoded USART. These systems are based on transmitter and receiver ICs that support ASK and FSK modulation schemes.

The goal is to perform full duplex radio transmission of characters using two different frequency carriers. See the configuration in Figure 39-16.

Figure 39-16: Manchester Encoded Characters RF Transmission



The USART peripheral is configured as a Manchester encoder/decoder. Looking at the downstream communication channel, Manchester encoded characters are serially sent to the RF emitter. This may also include a user defined preamble and a start frame delimiter. Mostly, preamble is used in the RF receiver to distinguish between a valid data from a transmitter and signals due to noise. The Manchester stream is then modulated. See Figure 39-17 for an example of ASK modulation scheme. When a logic one is sent to the ASK modulator, the power amplifier, referred to as PA, is enabled and transmits an RF signal at downstream frequency. When a logic zero is transmitted, the RF signal is turned off. If the FSK modulator is activated, two different frequencies are used to transmit data. When a logic 1 is sent, the modulator outputs an RF signal at frequency F0 and switches to F1 if the data sent is a 0. See Figure 39-18.

From the receiver side, another carrier frequency is used. The RF receiver performs a bit check operation examining demodulated data stream. If a valid pattern is detected, the receiver switches to Receiving mode. The demodulated stream is sent to the Manchester decoder. Because of bit checking inside RF IC, the data transferred to the microcontroller is reduced by a user-defined number of bits. The Manchester preamble length is to be defined in accordance with the RF IC configuration.

Figure 39-17: ASK Modulator Output

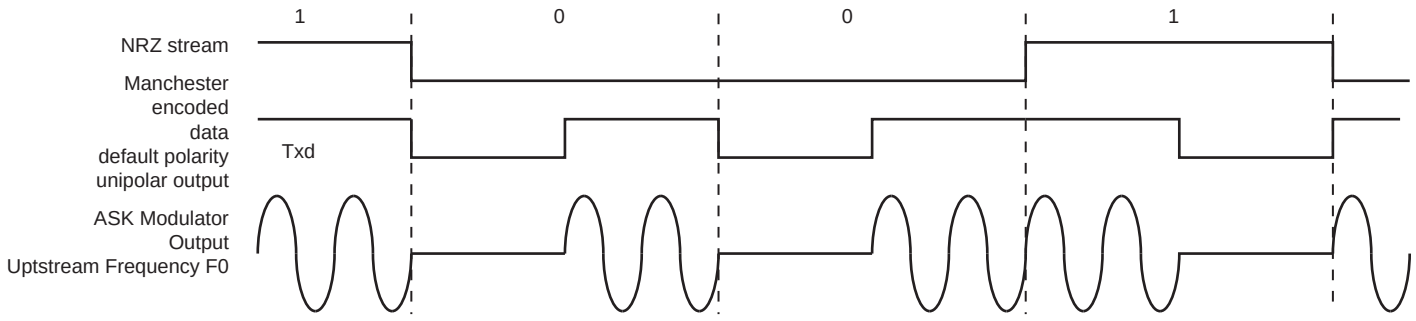
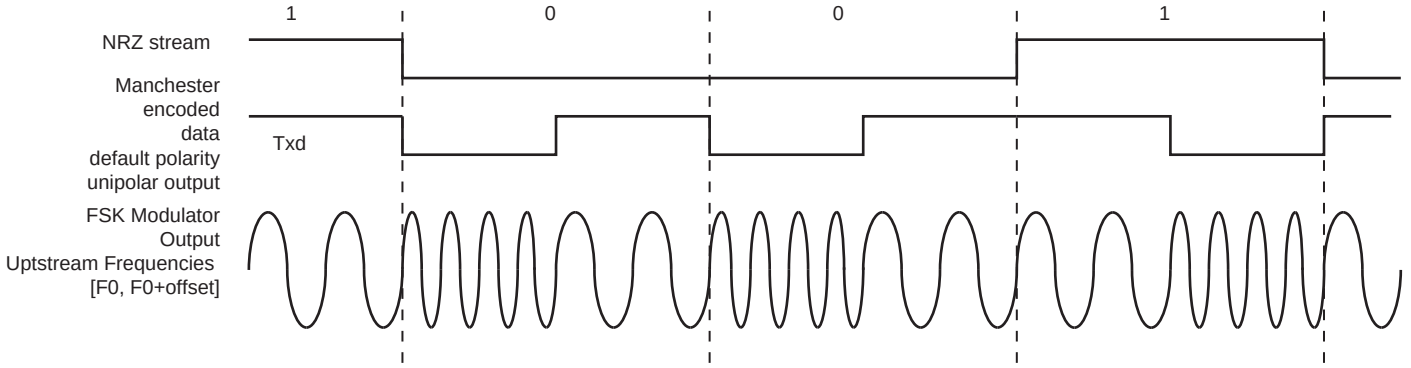


Figure 39-18: FSK Modulator Output



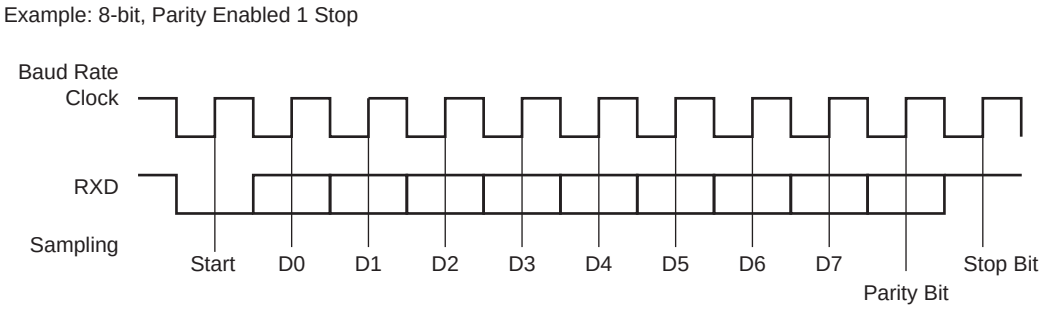
39.6.3.6 Synchronous Receiver

In Synchronous mode (SYNC = 1), the receiver samples the RXD signal on each rising edge of the baud rate clock. If a low level is detected, it is considered as a start. All data bits, the parity bit and the stop bits are sampled and the receiver waits for the next start bit. Synchronous mode operations provide a high-speed transfer capability.

Configuration fields and bits are the same as in Asynchronous mode.

Figure 39-19 illustrates a character reception in Synchronous mode.

Figure 39-19: Synchronous Mode Character Reception

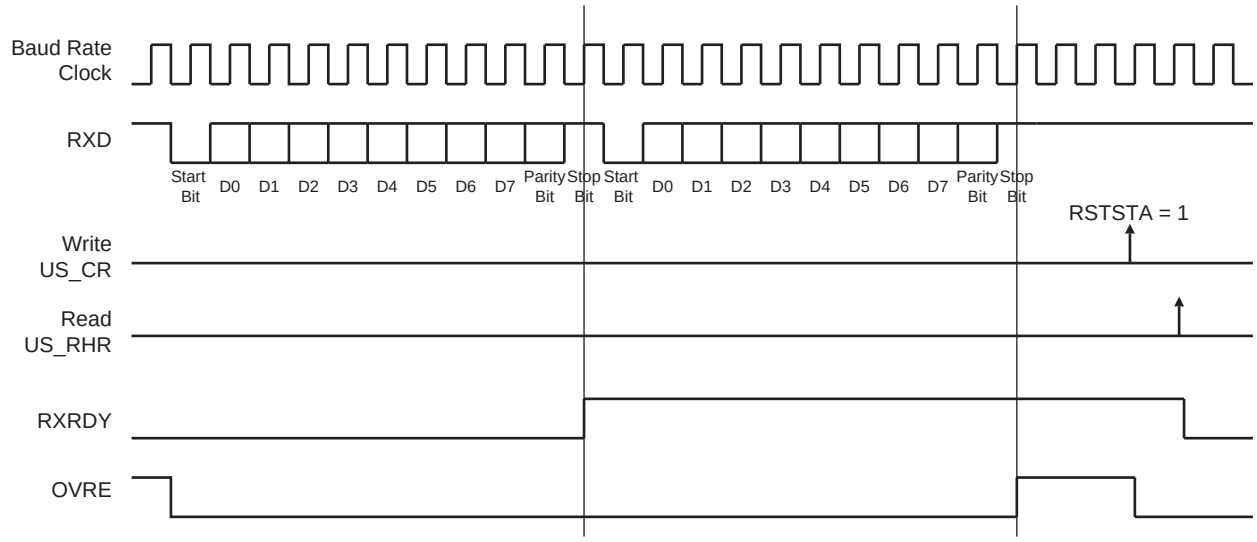


39.6.3.7 Receiver Operations

When a character reception is completed, it is transferred to the Receive Holding register (US_RHR) and the RXRDY bit in US_CSR rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing a 1 to the RSTSTA (Reset Status) bit in the US_CR.

© 2017 Microchip Technology Inc.

Figure 39-20: Receiver Status



39.6.3.8 Parity

The USART supports five Parity modes that are selected by writing to the PAR field in the US_MR. The PAR field also enables the Multi-drop mode, see Section 39.6.3.9 "Multidrop Mode". Even and odd parity bit generation and error detection are supported.

If even parity is selected, the parity generator of the transmitter drives the parity bit to 0 if a number of 1s in the character data bit is even, and to 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If odd parity is selected, the parity generator of the transmitter drives the parity bit to 1 if a number of 1s in the character data bit is even, and to 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit to 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled to 0. If the space parity is used, the parity generator of the transmitter drives the parity bit to 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled to 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

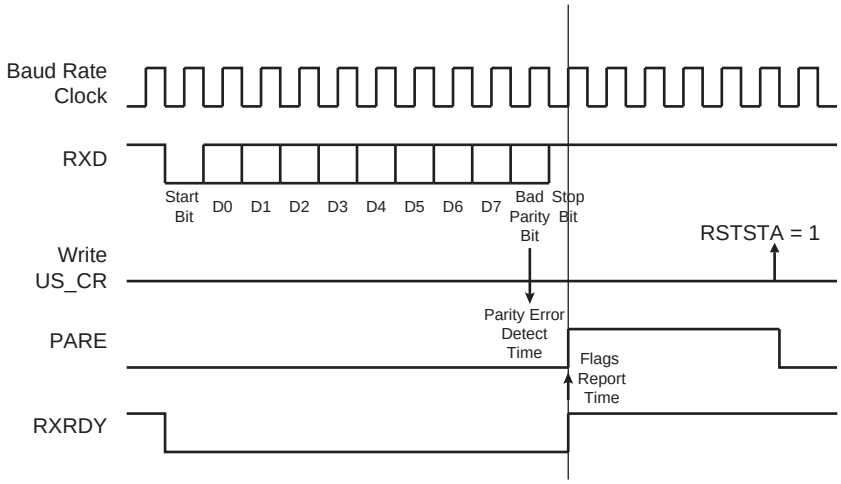
Table 39-8 shows an example of the parity bit for the character 0x41 (character ASCII "A") depending on the configuration of the USART. Because there are two bits set to 1 in the character value, the parity bit is set to 1 when the parity is odd, or configured to 0 when the parity is even.

Table 39-8: Parity Bit Examples

Character	Hexadecimal	Binary	Parity Bit	Parity Mode
A	0x41	0100 0001	1	Odd
A	0x41	0100 0001	0	Even
A	0x41	0100 0001	1	Mark
A	0x41	0100 0001	0	Space
A	0x41	0100 0001	None	None

When the receiver detects a parity error, it sets the PARE (Parity Error) bit in the US_CSR. The PARE bit can be cleared by writing a 1 to the RSTSTA bit the US_CR. Figure 39-21 illustrates the parity bit status setting and clearing.

Figure 39-21: Parity Error



39.6.3.9 Multidrop Mode

If the value 0x6 or 0x07 is written to the PAR field in the US_MR, the USART runs in Multidrop mode. This mode differentiates the data characters and the address characters. Data is transmitted with the parity bit at 0 and addresses are transmitted with the parity bit at 1.

If the USART is configured in Multidrop mode, the receiver sets the PARE parity error bit when the parity bit is high and the transmitter is able to send a character with the parity bit high when a 1 is written to the SENTA bit in the US_CR.

To handle parity error, the PARE bit is cleared when a 1 is written to the RSTSTA bit in the US_CR.

The transmitter sends an address byte (parity bit set) when SENDA is written to in the US_CR. In this case, the next byte written to the US_THR is transmitted as an address. Any character written in the US_THR without having written the command SENDA is transmitted normally with the parity at 0.

39.6.3.10 Transmitter Timeguard

The timeguard feature enables the USART interface with slow remote devices.

The timeguard function enables the transmitter to insert an idle state on the TXD line between two characters. This idle state actually acts as a long stop bit.

The duration of the idle state is programmed in the TG field of the Transmitter Timeguard register (US_TTGR). When this field is written to zero no timeguard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in TG in addition to the number of stop bits.

As illustrated in Figure 39-22, the behavior of TXRDY and TXEMPTY status bits is modified by the programming of a timeguard. TXRDY rises only when the start bit of the next character is sent, and thus remains to 0 during the timeguard transmission if a character has been written in US_THR. TXEMPTY remains low until the timeguard transmission is completed as the timeguard is part of the current character being transmitted.

© 2017 Microchip Technology Inc.

Figure 39-22: Timeguard Operations

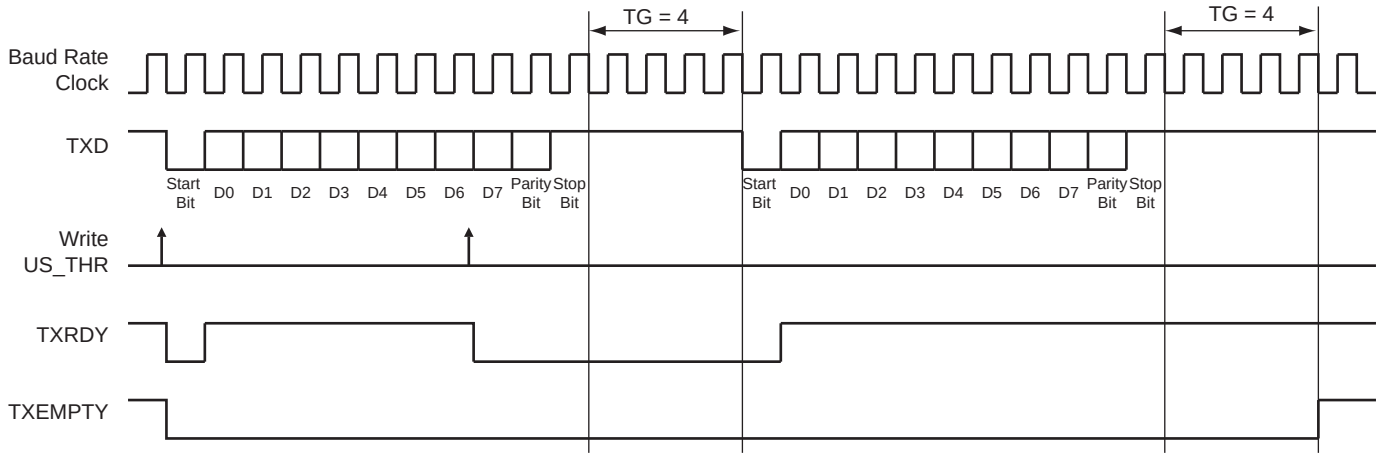


Table 39-9 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the baud rate.

Table 39-9: Maximum Timeguard Length Depending on Baud Rate

Baud Rate (bit/s)	Bit Time (µs)	Timeguard (ms)
1,200	833	212.50
9,600	104	26.56
14,400	69.4	17.71
19,200	52.1	13.28
28,800	34.7	8.85
38,400	26	6.63
56,000	17.9	4.55
57,600	17.4	4.43
115,200	8.7	2.21

39.6.3.11 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the US_CSR rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out register (US_RTOR). If the TO field is written to 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in the US_CSR remains at 0. Otherwise, the receiver loads a 16-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in US_CSR rises. Then, the user can either:

- Stop the counter clock until a new character is received. This is performed by writing a 1 to the STTTO (Start Time-out) bit in the US_CR. In this case, the idle state on RXD before a new character is received will not provide a time-out. This prevents having to handle an interrupt before a character is received and allows waiting for the next idle state on RXD after a frame is received.
- Obtain an interrupt while no character is received. This is performed by writing a 1 to the RETTO (Reload and Start Time-out) bit in the US_CR. If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

If STTTO is performed, the counter clock is stopped until a first character is received. The idle state on RXD before the start of the frame does not provide a time-out. This prevents having to obtain a periodic interrupt and enables a wait of the end of frame when the idle state on RXD is detected.

If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

© 2017 Microchip Technology Inc.

Figure 39-23 shows the block diagram of the Receiver Time-out feature.

Figure 39-23: Receiver Time-out Block Diagram

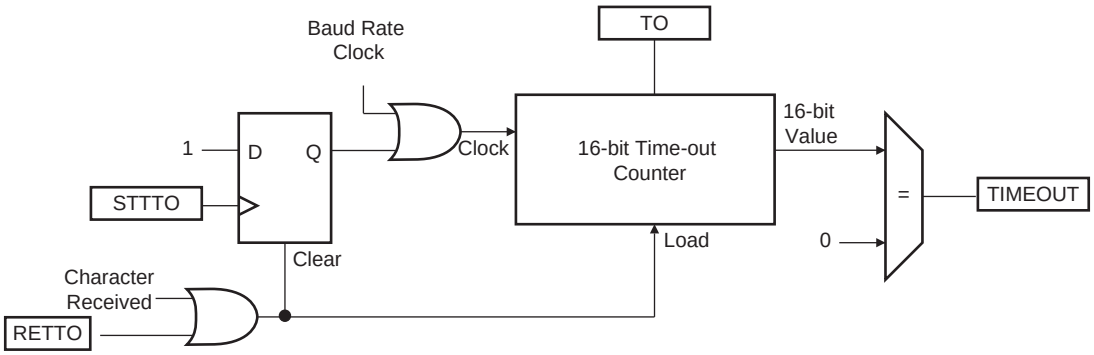


Table 39-10 gives the maximum time-out period for some standard baud rates.

Table 39-10: Maximum Time-out Period

Baud Rate (bit/s)	Bit Time (µs)	Time-out (ms)
600	1,667	109,225
1,200	833	54,613
2,400	417	27,306
4,800	208	13,653
9,600	104	6,827
14,400	69	4,551
19,200	52	3,413
28,800	35	2,276
38,400	26	1,704
56,000	18	1,170
57,600	17	1,138
200,000	5	328

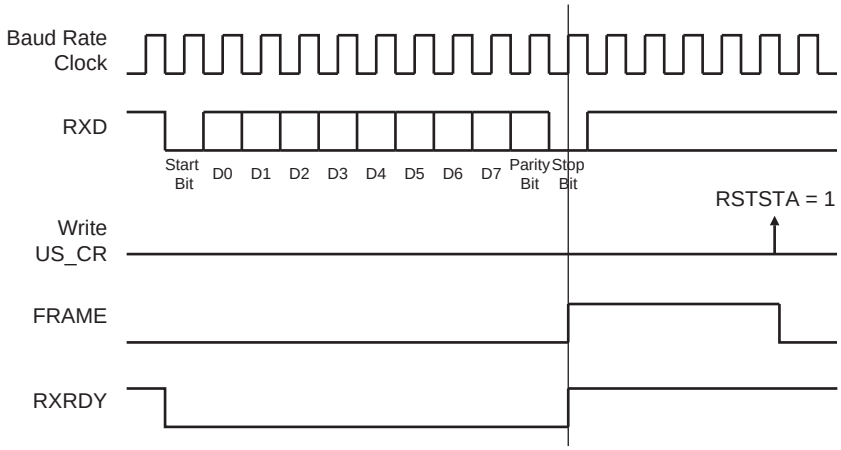
39.6.3.12 Framing Error

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of US_CSR. The FRAME bit is asserted in the middle of the stop bit as soon as the framing error is detected. It is cleared by writing a 1 to the RSTSTA bit in the US_CR.

© 2017 Microchip Technology Inc.

Figure 39-24: Framing Error Status



39.6.3.13 Transmit Break

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing a 1 to the STTBRK bit in the US_CR. This can be performed at any time, either while the transmitter is empty (no character in either the Shift register or in US_THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBRK command is requested further STTBRK commands are ignored until the end of the break is completed.

The break condition is removed by writing a 1 to the STPBRK bit in the US_CR. If the STPBRK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e., the STTBRK and STPBRK commands are processed only if the TXRDY bit in US_CSR is to 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

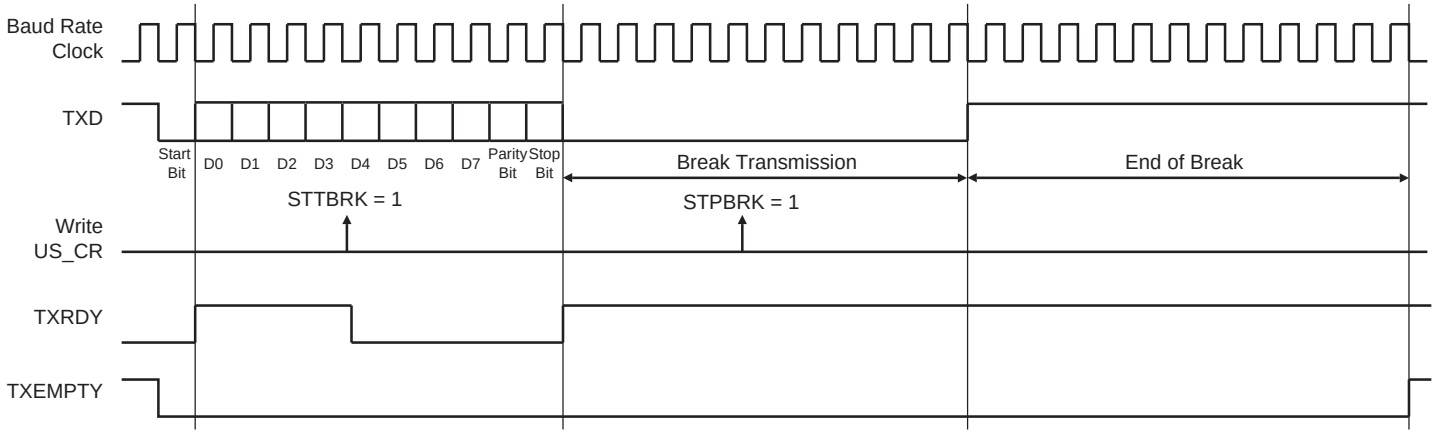
Writing US_CR with both STTBRK and STPBRK bits to 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding register while a break is pending, but not started, is ignored.

After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 bit times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 39-25 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 39-25: Break Transmission



39.6.3.14 Receive Break

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data to 0x00, but FRAME remains low.

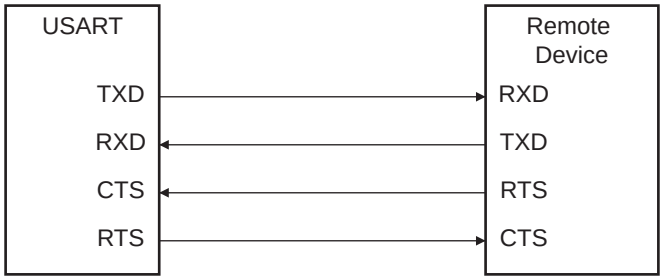
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing a 1 to the RSTSTA bit in the US_CR.

An end of receive break is detected by a high level for at least 2/16 of a bit period in Asynchronous operating mode or one sample at high level in Synchronous operating mode. The end of break detection also asserts the RXBRK bit.

39.6.3.15 Hardware Handshaking

The USART features a hardware handshaking out-of-band flow control. The RTS and CTS pins are used to connect with the remote device, as shown in Figure 39-26.

Figure 39-26: Connection with a Remote Device for Hardware Handshaking



Setting the USART to operate with hardware handshaking is performed by writing the USART_MODE field in US_MR to the value 0x2.

The USART behavior when hardware handshaking is enabled is the same as the behavior in standard Synchronous or Asynchronous mode, except that the receiver drives the RTS pin as described below and the level on the CTS pin modifies the behavior of the transmitter as described below. Using this mode requires using the DMA channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 39-27 shows how the receiver operates if hardware handshaking is enabled. The RTS pin is driven high if the receiver is disabled or if the DMA status flag indicates that the buffer transfer is completed. Normally, the remote device does not start transmitting while its CTS pin (driven by RTS) is high. As soon as the receiver is enabled, the RTS falls, indicating to the remote device that it can start transmitting. Defining a new transfer descriptor in the DMA clears the status flag and, as a result, asserts the pin RTS low.

Figure 39-27: Receiver Behavior when Operating with Hardware Handshaking

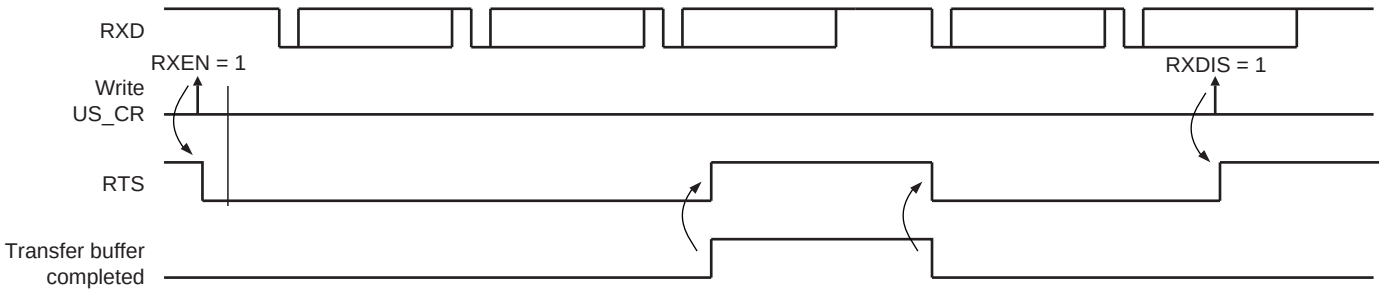
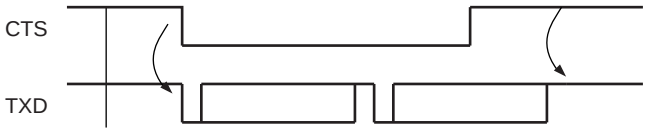


Figure 39-28 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processed, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 39-28: Transmitter Behavior when Operating with Hardware Handshaking



© 2017 Microchip Technology Inc.

39.6.4 ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

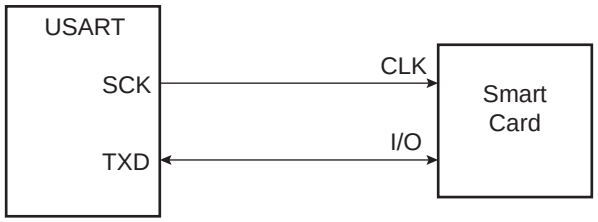
Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in US_MR to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

39.6.4.1 ISO7816 Mode Overview

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see Section 39-2 ”Baud Rate Generator”).

The USART connects to a smart card as shown in Figure 39-29. The TXD line becomes bidirectional and the baud rate generator feeds the ISO7816 clock on the SCK pin. As the TXD pin becomes bidirectional, its output remains driven by the output of the transmitter but only when the transmitter is active while its input is directed to the input of the receiver. The USART is considered as the master of the communication as it generates the clock.

Figure 39-29: Connection of a Smart Card to the USART



When operating in ISO7816, either in T = 0 or T = 1 modes, the character format is fixed. The configuration is 8 data bits, even parity and 1 or 2 stop bits, regardless of the values programmed in the CHRL, MODE9, PAR and CHMODE fields. MSBF can be used to transmit LSB or MSB first. Parity Bit (PAR) can be used to transmit in normal or inverse mode. Refer to Section 39.7.3 ”USART Mode Register” and “PAR: Parity Type” .

The USART cannot operate concurrently in both Receiver and Transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value.

39.6.4.2 Protocol T = 0

In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 39-30.

If a parity error is detected by the receiver, it drives the I/O line to 0 during the guard time, as shown in Figure 39-31. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time which lasts 1 bit time.

When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding register (US_RHR). It appropriately sets the PARE bit in the Status register (US_SR) so that the software can handle the error.

Figure 39-30: T = 0 Protocol without Parity Error

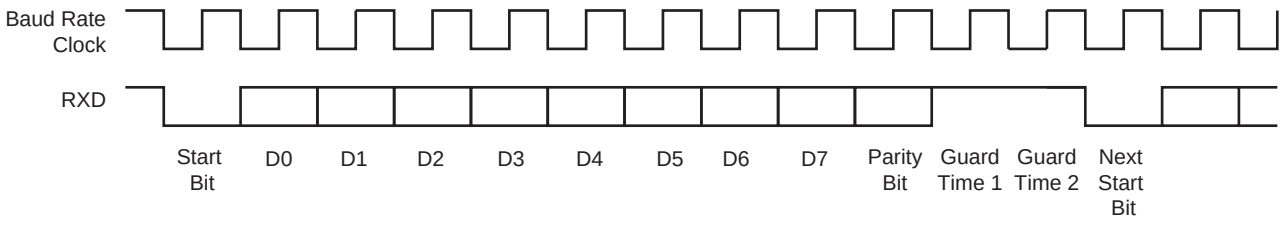
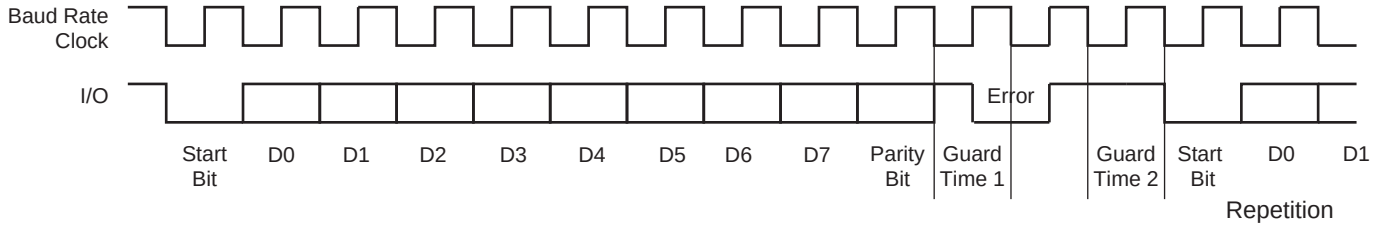


Figure 39-31: T = 0 Protocol with Parity Error



Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (US_NER) register. The NB_ERRORS field can record up to 255 errors. Reading US_NER automatically clears the NB_ERRORS field.

Receive NACK Inhibit

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in US_MR. If INACK is to 1, no error signal is driven on the I/O line even if a parity bit is detected.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding register, as if no error occurred and the RXRDY bit does rise.

Transmit Character Repetition

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next one. Repetition is enabled by writing the MAX_ITERATION field in the US_MR at a value higher than 0. Each character can be transmitted up to eight times; the first transmission plus seven repetitions.

If MAX_ITERATION does not equal zero, the USART repeats the character as many times as the value loaded in MAX_ITERATION.

When the USART repetition number reaches MAX_ITERATION and the last repeated character is not acknowledged, the ITER bit is set in US_CSR. If the repetition of the character is acknowledged by the receiver, the repetitions are stopped and the iteration counter is cleared.

The ITER bit in US_CSR can be cleared by writing a 1 to the RSTIT bit in the US_CR.

Disable Successive Receive NACK

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the bit DSNACK in the US_MR. The maximum number of NACKs transmitted is programmed in the MAX_ITERATION field. As soon as MAX_ITERATION is reached, no error signal is driven on the I/O line and the ITER bit in the US_CSR is set.

39.6.4.3 Protocol T = 1

When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the US_CSR.

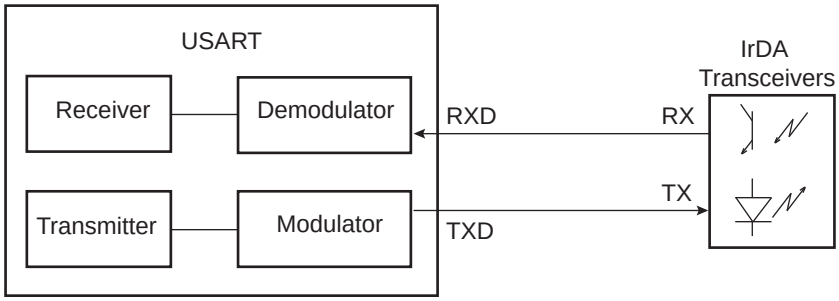
39.6.5 IrDA Mode

The USART features an IrDA mode supplying half-duplex point-to-point wireless communication. It embeds the modulator and demodulator which allows a glueless connection to the infrared transceivers, as shown in Figure 39-32. The modulator and demodulator are compliant with the IrDA specification version 1.1 and support data transfer speeds ranging from 2.4 kbit/s to 115.2 kbit/s.

The IrDA mode is enabled by setting the USART_MODE field in US_MR to the value 0x8. The IrDA Filter register (US_IF) is used to configure the demodulator filter. The USART transmitter and receiver operate in a normal Asynchronous mode and all parameters are accessible. Note that the modulator and the demodulator are activated.

© 2017 Microchip Technology Inc.

Figure 39-32: Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled depending on the direction of the transmission to be managed.

To receive IrDA signals, the following needs to be done:

- Disable TX and Enable RX
- Configure the TXD pin as PIO and set it as an output to 0 (to avoid LED emission). Disable the internal pull-up (better for power consumption).
- Receive data

39.6.5.1 IrDA Modulation

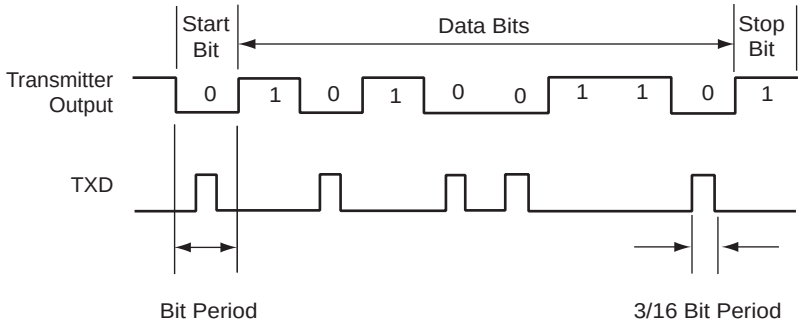
For baud rates up to and including 115.2 kbit/s, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 39-11.

Table 39-11: IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
2.4 kbit/s	78.13 µs
9.6 kbit/s	19.53 µs
19.2 kbit/s	9.77 µs
38.4 kbit/s	4.88 µs
57.6 kbit/s	3.26 µs
115.2 kbit/s	1.63 µs

Figure 39-33 shows an example of character transmission.

Figure 39-33: IrDA Modulation



39.6.5.2 IrDA Baud Rate

Table 39-12 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

Table 39-12: IrDA Baud Rate Error

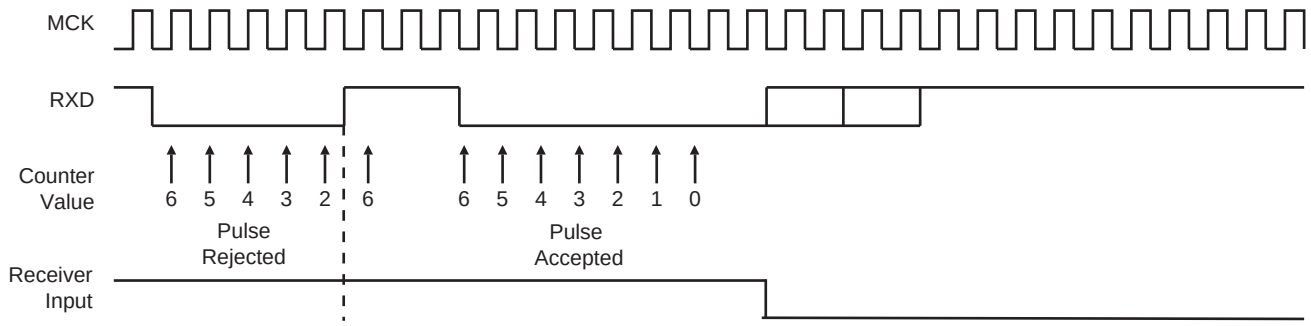
Peripheral Clock	Baud Rate (bit/s)	CD	Baud Rate Error	Pulse Time (µs)
3,686,400	115,200	2	0.00%	1.63
20,000,000	115,200	11	1.38%	1.63
32,768,000	115,200	18	1.25%	1.63
40,000,000	115,200	22	1.38%	1.63
3,686,400	57,600	4	0.00%	3.26
20,000,000	57,600	22	1.38%	3.26
32,768,000	57,600	36	1.25%	3.26
40,000,000	57,600	43	0.93%	3.26
3,686,400	38,400	6	0.00%	4.88
20,000,000	38,400	33	1.38%	4.88
32,768,000	38,400	53	0.63%	4.88
40,000,000	38,400	65	0.16%	4.88
3,686,400	19,200	12	0.00%	9.77
20,000,000	19,200	65	0.16%	9.77
32,768,000	19,200	107	0.31%	9.77
40,000,000	19,200	130	0.16%	9.77
3,686,400	9,600	24	0.00%	19.53
20,000,000	9,600	130	0.16%	19.53
32,768,000	9,600	213	0.16%	19.53
40,000,000	9,600	260	0.16%	19.53
3,686,400	2,400	96	0.00%	78.13
20,000,000	2,400	521	0.03%	78.13
32,768,000	2,400	853	0.04%	78.13

39.6.5.3 IrDA Demodulator

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in US_IF. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the peripheral clock speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with US_IF. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 39-34 illustrates the operations of the IrDA demodulator.

Figure 39-34: IrDA Demodulator Operations



© 2017 Microchip Technology Inc.

The programmed value in the US_IF register must always meet the following criteria:

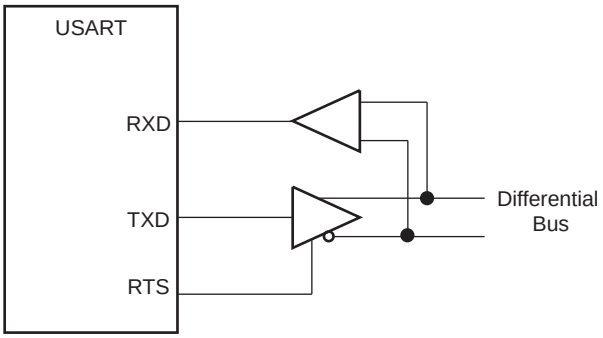
$t_{peripheral\ clock} \times (IRDA\_FILTER + 3) < 1.41\ \mu s$

As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in US_FIDI must be set to a value higher than 0 in order to make sure IrDA communications operate correctly.

39.6.6 RS485 Mode

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in Asynchronous or Synchronous mode and configuration of all the parameters is possible. The difference is that the RTS pin is driven high when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to an RS485 bus is shown in Figure 39-35.

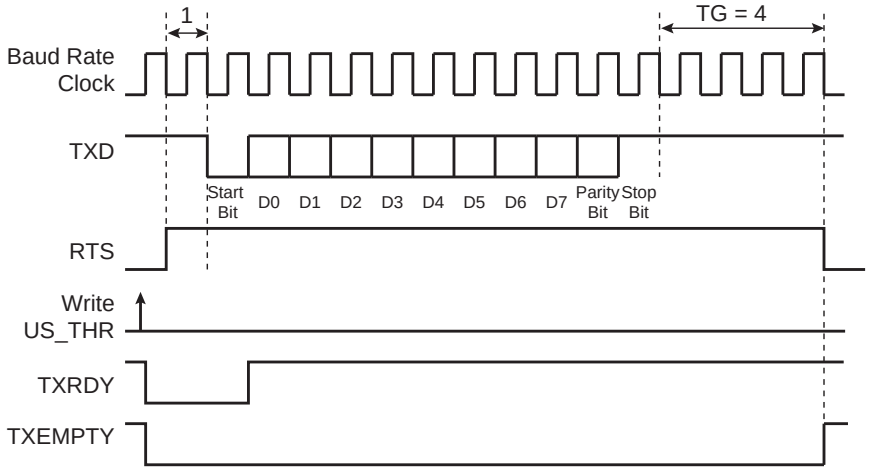
Figure 39-35: Typical Connection to a RS485 Bus



The USART is set in RS485 mode by writing the value 0x1 to the USART_MODE field in US_MR.

The RTS pin is at a level inverse to the TXEMPTY bit. Significantly, the RTS pin remains high when a timeguard is programmed so that the line can remain driven after the last character completion. Figure 39-36 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 39-36: Example of RTS Drive with Timeguard



39.6.7 SPI Mode

The Serial Peripheral Interface (SPI) mode is a synchronous serial data link that provides communication with external devices in Master or Slave mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master” which controls the data flow, while the other devices act as “slaves" which have data shifted into and out by the master. Different CPUs can take turns being masters and one master may simultaneously shift data into multiple slaves. (Multiple master protocol is the opposite of single master protocol, where one CPU is always the master while all of the others are always slaves.) However, only one slave may drive its output to write data back to the master at any given time.

A slave device is selected when its NSS signal is asserted by the master. The USART in SPI Master mode can address only one SPI slave because it can generate only one NSS signal.

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input of the slave.
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master.
- Serial Clock (SCK): This control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates. The SCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows the master to select or deselect the slave.

39.6.7.1 Modes of Operation

The USART can operate in SPI Master mode or in SPI Slave mode.

Operation in SPI Master mode is programmed by writing 0xE to the USART_MODE field in US_MR. In this case the SPI lines must be connected as described below:

- The MOSI line is driven by the output pin TXD
- The MISO line drives the input pin RXD
- The SCK line is driven by the output pin SCK
- The NSS line is driven by the output pin RTS

Operation in SPI Slave mode is programmed by writing to 0xF the USART_MODE field in US_MR. In this case the SPI lines must be connected as described below:

- The MOSI line drives the input pin RXD
- The MISO line is driven by the output pin TXD
- The SCK line drives the input pin SCK
- The NSS line drives the input pin CTS

In order to avoid unpredictable behavior, any change of the SPI mode must be followed by a software reset of the transmitter and of the receiver (except the initial configuration after a hardware reset). (See **Section 39.6.7.4 “Receiver and Transmitter Control”**).

39.6.7.2 Baud Rate

In SPI mode, the baud rate generator operates in the same way as in USART Synchronous mode. See Section 39.6.1.3 ”Baud Rate in Synchronous Mode or SPI Mode”. However, there are some restrictions:

In SPI Master mode:

- The external clock SCK must not be selected (USCLKS ≠ 0x3), and the bit CLKO must be set to 1 in the US_MR, in order to generate correctly the serial clock on the SCK pin.
- To obtain correct behavior of the receiver and the transmitter, the value programmed in CD must be superior or equal to 6.
- If the divided peripheral clock is selected, the value programmed in CD must be even to ensure a 50:50 mark/space ratio on the SCK pin, this value can be odd if the peripheral clock is selected.

In SPI Slave mode:

- The external clock (SCK) selection is forced regardless of the value of the USCLKS field in the US_MR. Likewise, the value written in US_BRGR has no effect, because the clock is provided directly by the signal on the USART SCK pin.
- To obtain correct behavior of the receiver and the transmitter, the external clock (SCK) frequency must be at least 6 times lower than the system clock.

39.6.7.3 Data Transfer

Up to nine data bits are successively shifted out on the TXD pin at each rising or falling edge (depending of CPOL and CPHA) of the programmed serial clock. There is no Start bit, no Parity bit and no Stop bit.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the US_MR. The nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The MSB data bit is always sent first in SPI mode (Master or Slave).

© 2017 Microchip Technology Inc.

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the US_MR. The clock phase is programmed with the CPHA bit. These two parameters determine the edges of the clock signal upon which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

Table 39-13: SPI Bus Protocol Mode

SPI Bus Protocol Mode	CPOL	CPHA
0	0	1
1	0	0
2	1	1
3	1	0

Figure 39-37: SPI Transfer Format (CPHA = 1, 8 bits per transfer)

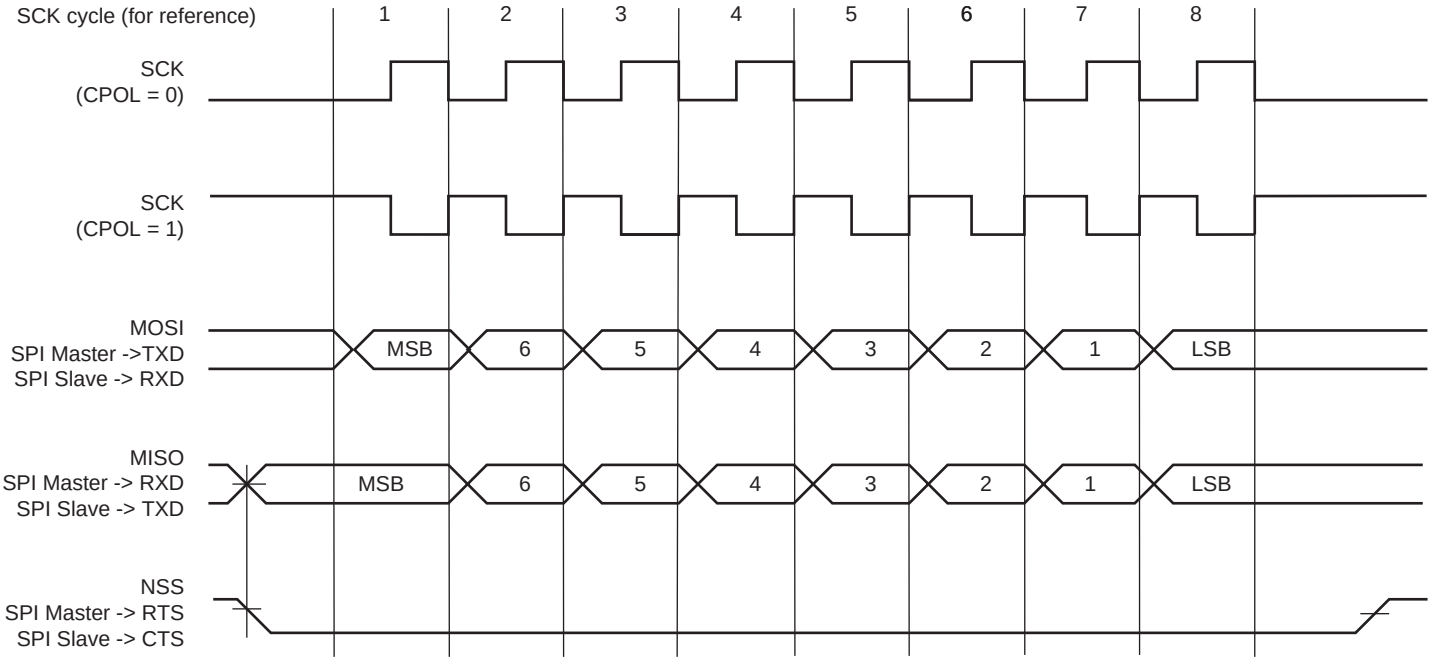
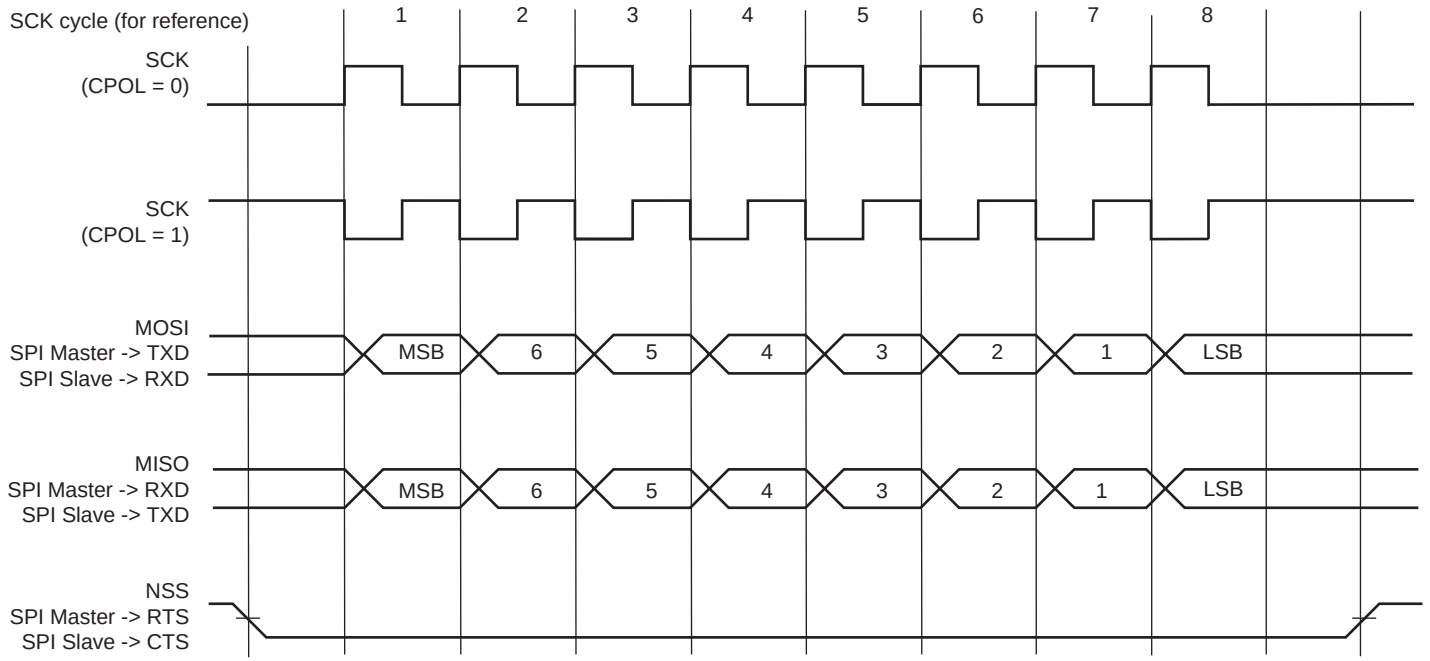


Figure 39-38: SPI Transfer Format (CPHA = 0, 8 bits per transfer)



39.6.7.4 Receiver and Transmitter Control

See Section 39.6.2 "Receiver and Transmitter Control"

39.6.7.5 Character Transmission

The characters are sent by writing in the Transmit Holding register (US_THR). An additional condition for transmitting a character can be added when the USART is configured in SPI Master mode. In the USART Mode Register (SPI_MODE) (USART_MR), the value configured on the bit WRDBT can prevent any character transmission (even if US_THR has been written) while the receiver side is not ready (character not read). When WRDBT equals 0, the character is transmitted whatever the receiver status. If WRDBT is set to 1, the transmitter waits for the Receive Holding register (US_RHR) to be read before transmitting the character (RXRDY flag cleared), thus preventing any overflow (character loss) on the receiver side.

The chip select line is de-asserted for a period equivalent to three bits between the transmission of two data.

The transmitter reports two status bits in US_CSR: TXRDY (Transmitter Ready), which indicates that US_THR is empty and TXEMPTY, which indicates that all the characters written in US_THR have been processed. When the current character processing is completed, the last character written in US_THR is transferred into the Shift register of the transmitter and US_THR becomes empty, thus TXRDY rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in US_THR while TXRDY is low has no effect and the written character is lost.

If the USART is in SPI Slave mode and if a character must be sent while the US_THR is empty, the UNRE (Underrun Error) bit is set. The TXD transmission line stays at high level during all this time. The UNRE bit is cleared by writing a 1 to the RSTSTA (Reset Status) bit in US_CR.

In SPI Master mode, the slave select line (NSS) is asserted at low level one t_{bit} (t_{bit} being the nominal time required to transmit a bit) before the transmission of the MSB bit and released at high level one t_{bit} after the transmission of the LSB bit. So, the slave select line (NSS) is always released between each character transmission and a minimum delay of three t_{bit} always inserted. However, in order to address slave devices supporting the CSAAT mode (Chip Select Active After Transfer), the slave select line (NSS) can be forced at low level by writing a 1 to the RTSEN bit in the US_CR. The slave select line (NSS) can be released at high level only by writing a 1 to the RTSDIS bit in the US_CR (for example, when all data have been transferred to the slave device).

In SPI Slave mode, the transmitter does not require a falling edge of the slave select line (NSS) to initiate a character transmission but only a low level. However, this low level must be present on the slave select line (NSS) at least one t_{bit} before the first serial clock cycle corresponding to the MSB bit.

© 2017 Microchip Technology Inc.

39.6.7.6 Character Reception

When a character reception is completed, it is transferred to the Receive Holding register (US_RHR) and the RXRDY bit in the Status register (US_CSR) rises. If a character is completed while RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing a 1 to the RSTSTA (Reset Status) bit in the US_CR.

To ensure correct behavior of the receiver in SPI Slave mode, the master device sending the frame must ensure a minimum delay of one t_{bit} between each character transmission. The receiver does not require a falling edge of the slave select line (NSS) to initiate a character reception but only a low level. However, this low level must be present on the slave select line (NSS) at least one t_{bit} before the first serial clock cycle corresponding to the MSB bit.

39.6.7.7 Receiver Timeout

Because the receiver baud rate clock is active only during data transfers in SPI mode, a receiver timeout is impossible in this mode, whatever the time-out value is (field TO) in the US_RTOR.

39.6.8 LIN Mode

The LIN mode provides master node and slave node connectivity on a LIN bus.

The LIN (Local Interconnect Network) is a serial communication protocol which efficiently supports the control of mechatronic nodes in distributed automotive applications.

The main properties of the LIN bus are:

- Single master/multiple slaves concept
- Low-cost silicon implementation based on common UART/SCI interface hardware, an equivalent in software, or as a pure state machine.
- Self synchronization without quartz or ceramic resonator in the slave nodes
- Deterministic signal transmission
- Low cost single-wire implementation
- Speed up to 20 kbit/s

LIN provides cost efficient bus communication where the bandwidth and versatility of CAN are not required.

The LIN mode enables processing LIN frames with a minimum of action from the microprocessor.

39.6.8.1 Modes of Operation

The USART can act either as a LIN master node or as a LIN slave node.

The node configuration is chosen by setting the USART_MODE field in the USART Mode register (US_MR):

- LIN master node (USART_MODE = 0xA)
- LIN slave node (USART_MODE = 0xB)

In order to avoid unpredictable behavior, any change of the LIN node configuration must be followed by a software reset of the transmitter and of the receiver (except the initial node configuration after a hardware reset). (See **Section 39.6.8.3 “Receiver and Transmitter Control”**.)

39.6.8.2 Baud Rate Configuration

See Section 39.6.1.1 ”Baud Rate in Asynchronous Mode”

The baud rate is configured in US_BRGR.

39.6.8.3 Receiver and Transmitter Control

See Section 39.6.2 ”Receiver and Transmitter Control”

39.6.8.4 Character Transmission

See Section 39.6.3.1 ”Transmitter Operations”.

39.6.8.5 Character Reception

See Section 39.6.3.7 ”Receiver Operations”.

39.6.8.6 Header Transmission (Master Node Configuration)

All the LIN frames start with a header which is sent by the master node and consists of a Synch Break Field, Synch Field and Identifier Field.

So in master node configuration, the frame handling starts with the sending of the header.

The header is transmitted as soon as the identifier is written in the LIN Identifier register (US_LINIR). At this moment the flag TXRDY falls.

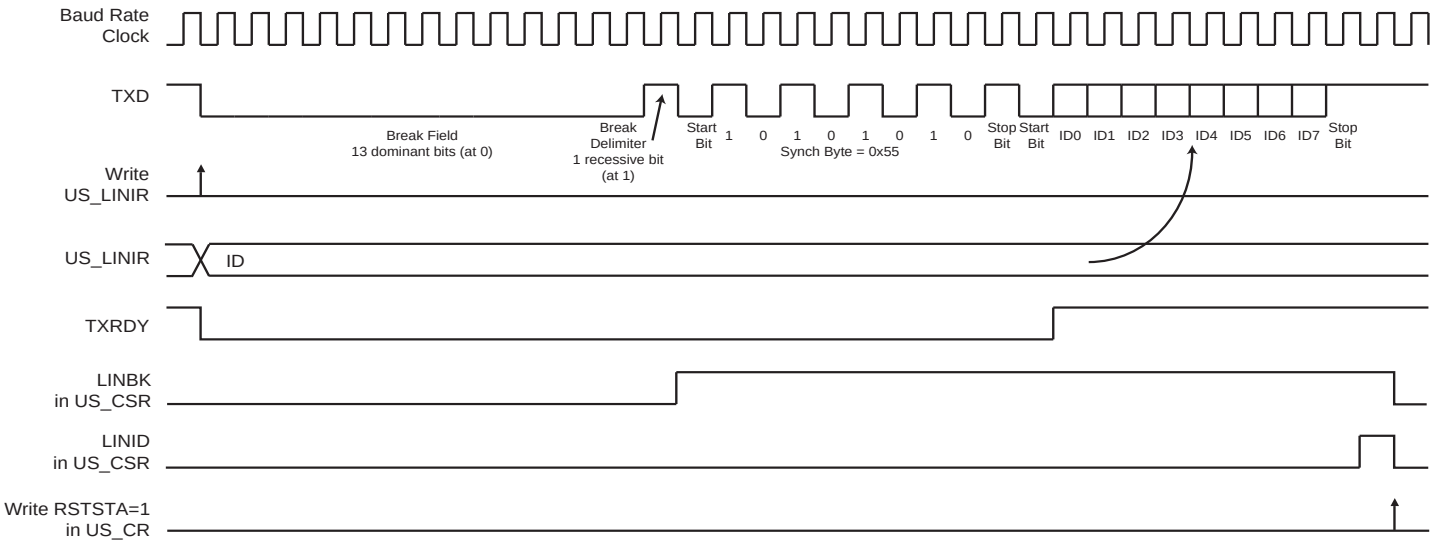
The Break Field, the Synch Field and the Identifier Field are sent automatically one after the other.

The Break Field consists of 13 dominant bits and 1 recessive bit, the Synch Field is the character 0x55 and the Identifier corresponds to the character written in the LIN Identifier register (US_LINIR). The Identifier parity bits can be automatically computed and sent (see Section 39.6.8.9 "Identifier Parity").

The flag TXRDY rises when the identifier character is transferred into the Shift register of the transmitter.

As soon as the Synch Break Field is transmitted, the flag LINBK in US_CSR is set to 1. Likewise, as soon as the Identifier Field is sent, the flag bit LINID in the US_CSR is set to 1. These flags are reset by writing a 1 to the bit RSTSTA in US_CR.

Figure 39-39: Header Transmission



39.6.8.7 Header Reception (Slave Node Configuration)

All the LIN frames start with a header which is sent by the master node and consists of a Synch Break Field, Synch Field and Identifier Field.

In slave node configuration, the frame handling starts with the reception of the header.

The USART uses a break detection threshold of 11 nominal bit times at the actual baud rate. At any time, if 11 consecutive recessive bits are detected on the bus, the USART detects a Break Field. As long as a Break Field has not been detected, the USART stays idle and the received data are not taken in account.

When a Break Field has been detected, the flag LINBK in US_CSR is set to 1 and the USART expects the Synch Field character to be 0x55. This field is used to update the actual baud rate in order to stay synchronized (see Section 39.6.8.8 "Slave Node Synchronization"). If the received Synch character is not 0x55, an Inconsistent Synch Field error is generated (see Section 39.6.8.14 "LIN Errors").

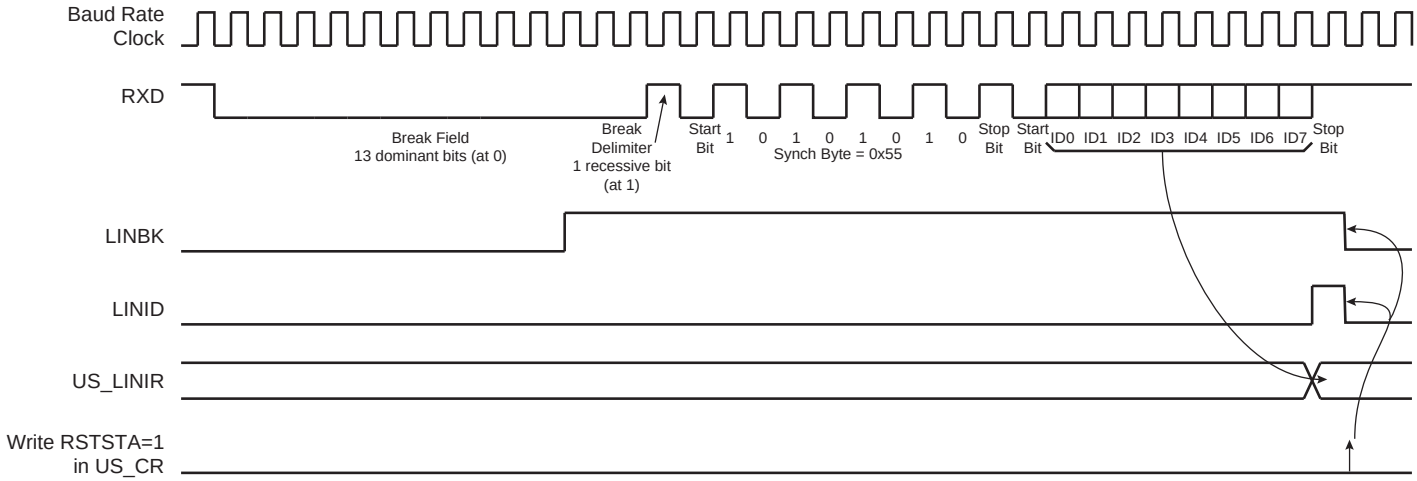
After receiving the Synch Field, the USART expects to receive the Identifier Field.

When the Identifier Field has been received, the flag bit LINID in the US_CSR is set to 1. At this moment the field IDCHR in the LIN Identifier register (US_LINIR) is updated with the received character. The Identifier parity bits can be automatically computed and checked (see Section 39.6.8.9 "Identifier Parity").

The flag bits LINID and LINBK are reset by writing a 1 to the bit RSTSTA in US_CR.

© 2017 Microchip Technology Inc.

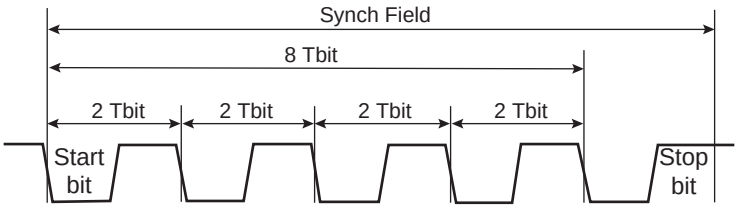
Figure 39-40: Header Reception



39.6.8.8 Slave Node Synchronization

The synchronization is done only in slave node configuration. The procedure is based on time measurement between falling edges of the Synch Field. The falling edges are available in distances of 2, 4, 6 and 8 bit times.

Figure 39-41: Synch Field



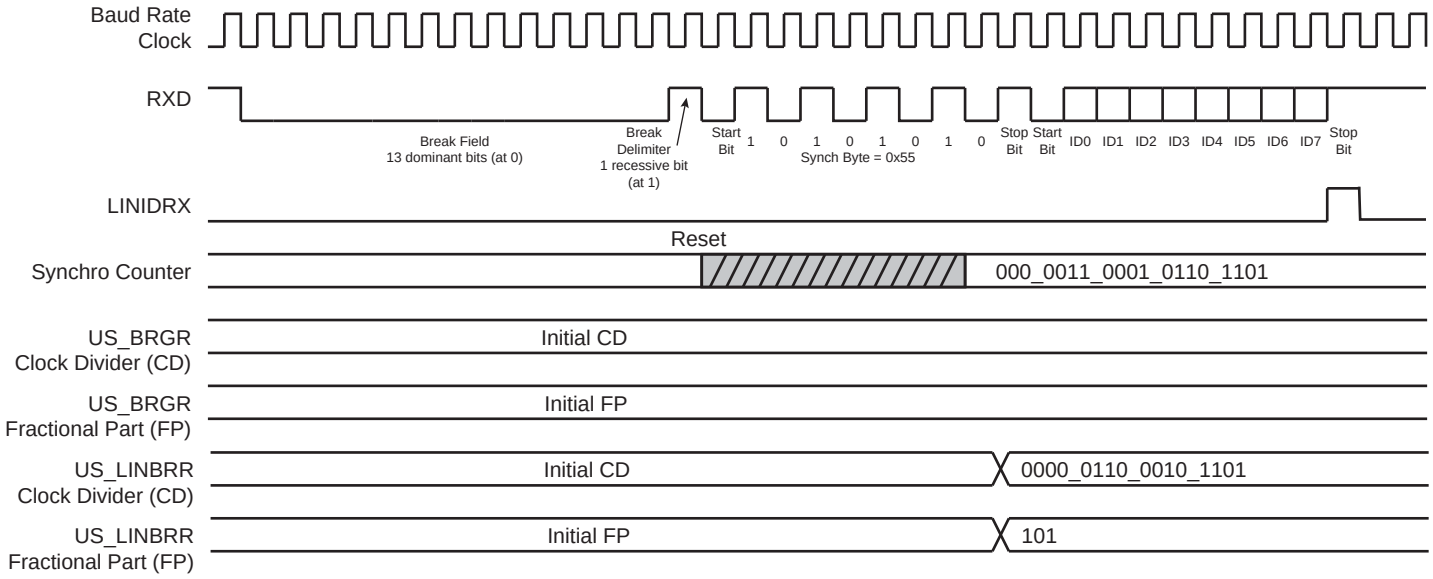
The time measurement is made by a 19-bit counter clocked by the sampling clock (see Section 39.6.1 "Baud Rate Generator").

When the start bit of the Synch Field is detected, the counter is reset. Then during the next eight t_{bit} of the Synch Field, the counter is incremented. At the end of these eight t_{bit}, the counter is stopped. At this moment, the 16 most significant bits of the counter (value divided by 8) give the new clock divider (LINCD) and the three least significant bits of this value (the remainder) give the new fractional part (LINFP).

When the Synch Field has been received, the clock divider (CD) and the fractional part (FP) are updated in US_BRGR.

If it appears that the sampled Synch character is not equal to 0x55, then the error flag LINISFE in US_CSR is set to 1. It is reset by writing bit RSTSTA to 1 in US_CR.

Figure 39-42: Slave Node Synchronization



The accuracy of the synchronization depends on several parameters:

- Nominal clock frequency (f_{Nom}) (the theoretical slave node clock frequency)
- Baud Rate
- Oversampling (OVER = 0 → 16X or OVER = 0 → 8X)

The following formula is used to compute the deviation of the slave bit rate relative to the master bit rate after synchronization (f_{SLAVE} is the real slave node clock frequency):

$$\text{Baudrate\_deviation} = \left(100 \times \frac{[\alpha \times 8 \times (2 - \text{OVER}) + \beta] \times \text{Baudrate}}{8 \times f_{SLAVE}}\right)\%$$

$$\text{Baudrate\_deviation} = \left(100 \times \frac{[\alpha \times 8 \times (2 - \text{OVER}) + \beta] \times \text{Baudrate}}{8 \times \left(\frac{f_{TOL\_UNSYNCH}}{100}\right) \times f_{Nom}}\right)\%$$

$$-0.5 \le \alpha \le +0.5 \quad -1 < \beta < +1$$

$f_{TOL\_UNSYNCH}$ is the deviation of the real slave node clock from the nominal clock frequency. The LIN Standard imposes that it must not exceed ±15%. The LIN Standard imposes also that for communication between two nodes, their bit rate must not differ by more than ±2%. This means that the baudrate_deviation must not exceed ±1%.

It follows from that, a minimum value for the nominal clock frequency:

$$f_{Nom}(\text{min}) = \left(100 \times \frac{[0.5 \times 8 \times (2 - \text{OVER}) + 1] \times \text{Baudrate}}{8 \times \left(\frac{-15}{100} + 1\right) \times 1\%}\right)\text{Hz}$$

Examples:

- Baud rate = 20 kbit/s, OVER = 0 (Oversampling 16X) → f_{Nom}(min) = 2.64 MHz
- Baud rate = 20 kbit/s, OVER = 1 (Oversampling 8X) → f_{Nom}(min) = 1.47 MHz
- Baud rate = 1 kbit/s, OVER = 0 (Oversampling 16X) → f_{Nom}(min) = 132 kHz
- Baud rate = 1 kbit/s, OVER = 1 (Oversampling 8X) → f_{Nom}(min) = 74 kHz

© 2017 Microchip Technology Inc.

39.6.8.9 Identifier Parity

A protected identifier consists of two subfields: the identifier and the identifier parity. Bits 0 to 5 are assigned to the identifier and bits 6 and 7 are assigned to the parity.

The USART interface can generate/check these parity bits, but this feature can also be disabled. The user can choose between two modes by the PARDIS bit of US_LINMR:

- PARDIS = 0:
  - During header transmission, the parity bits are computed and sent with the six least significant bits of the IDCHR field of the LIN Identifier register (US_LINIR). The bits 6 and 7 of this register are discarded.
  - During header reception, the parity bits of the identifier are checked. If the parity bits are wrong, an Identifier Parity error occurs (see **Section 39.6.3.8 “Parity”**). Only the six least significant bits of the IDCHR field are updated with the received Identifier. The bits 6 and 7 are stuck to 0.
- PARDIS = 1:
  - During header transmission, all the bits of the IDCHR field of the LIN Identifier register (US_LINIR) are sent on the bus.
  - During header reception, all the bits of the IDCHR field are updated with the received Identifier.

39.6.8.10 Node Action

Depending on the identifier, the node is affected – or not – by the LIN response. Consequently, after sending or receiving the identifier, the USART must be configured. There are three possible configurations:

- PUBLISH: the node sends the response.
- SUBSCRIBE: the node receives the response.
- IGNORE: the node is not concerned by the response, it does not send and does not receive the response.

This configuration is made by the field Node Action (NACT) in the US_LINMR (see **Section 39.7.26 “USART LIN Mode Register”**).

Example: a LIN cluster that contains a master and two slaves:

- Data transfer from the master to the slave1 and to the slave2:

NACT(master)=PUBLISH

NACT(slave1)=SUBSCRIBE

NACT(slave2)=SUBSCRIBE

- Data transfer from the master to the slave1 only:

NACT(master)=PUBLISH

NACT(slave1)=SUBSCRIBE

NACT(slave2)=IGNORE

- Data transfer from the slave1 to the master:

NACT(master)=SUBSCRIBE

NACT(slave1)=PUBLISH

NACT(slave2)=IGNORE

- Data transfer from the slave1 to the slave2:

NACT(master)=IGNORE

NACT(slave1)=PUBLISH

NACT(slave2)=SUBSCRIBE

- Data transfer from the slave2 to the master and to the slave1:

NACT(master)=SUBSCRIBE

NACT(slave1)=SUBSCRIBE

NACT(slave2)=PUBLISH

39.6.8.11 Response Data Length

The LIN response data length is the number of data fields (bytes) of the response excluding the checksum.

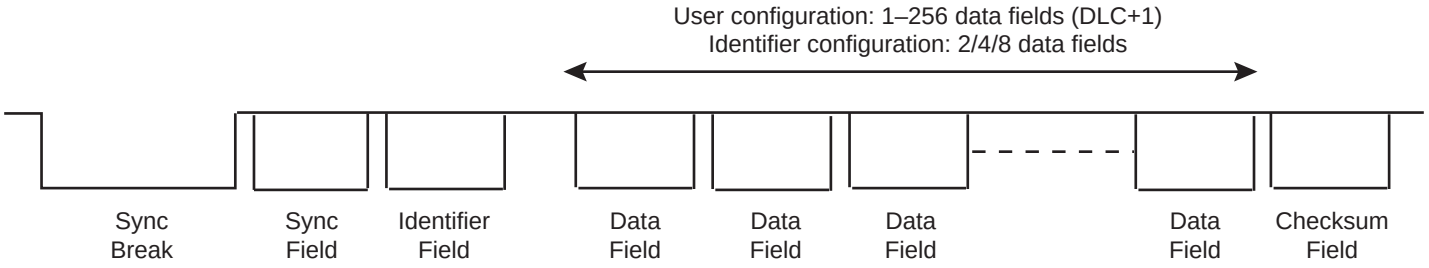
The response data length can either be configured by the user or be defined automatically by bits 4 and 5 of the Identifier (compatibility to LIN Specification 1.1). The user can choose between these two modes by the DLM bit of US_LINMR:

- DLM = 0: The response data length is configured by the user via the DLC field of US_LINMR. The response data length is equal to (DLC + 1) bytes. DLC can be programmed from 0 to 255, so the response can contain from 1 data byte up to 256 data bytes.
- DLM = 1: The response data length is defined by the Identifier (IDCHR in US_LINIR) according to the table below. The DLC field of US_LINMR is discarded. The response can contain 2 or 4 or 8 data bytes.

Table 39-14: Response Data Length if DLM = 1

IDCHR[5]	IDCHR[4]	Response Data Length [Bytes]
0	0	2
0	1	2
1	0	4
1	1	8

Figure 39-43: Response Data Length



39.6.8.12 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8-bit sum with carry, over all data bytes or all data bytes and the protected identifier. Checksum calculation over the data bytes only is called classic checksum and it is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum and it is used for communication with LIN 2.0 slaves.

The USART can be configured to:

- Send/Check an Enhanced checksum automatically (CHKDIS = 0 & CHKTYP = 0)
- Send/Check a Classic checksum automatically (CHKDIS = 0 & CHKTYP = 1)
- Not send/check a checksum (CHKDIS = 1)

This configuration is made by the Checksum Type (CHKTYP) and Checksum Disable (CHKDIS) fields of US_LINMR.

If the checksum feature is disabled, the user can send it manually all the same, by considering the checksum as a normal data byte and by adding 1 to the response data length (see **Section 39.6.8.11 “Response Data Length”**).

39.6.8.13 Frame Slot Mode

This mode is useful only for master nodes. It complies with the following rule: each frame slot should be longer than or equal to $t_{Frame\_Maximum}$.

If the Frame slot mode is enabled (FSDIS = 0) and a frame transfer has been completed, the TXRDY flag is set again only after $t_{Frame\_Maximum}$ delay, from the start of frame. So the master node cannot send a new header if the frame slot duration of the previous frame is inferior to $t_{Frame\_Maximum}$.

If the Frame slot mode is disabled (FSDIS = 1) and a frame transfer has been completed, the TXRDY flag is set again immediately.

© 2017 Microchip Technology Inc.

The $t_{Frame\_Maximum}$ is calculated as shown below:

If the Checksum is sent (CHKDIS = 0):

$t_{Header\_Nominal} = 34 \times t_{bit}$

$t_{Response\_Nominal} = 10 \times (NData + 1) \times t_{bit}$

$t_{Frame\_Maximum} = 1.4 \times (t_{Header\_Nominal} + t_{Response\_Nominal} + 1)$(1)

$t_{Frame\_Maximum} = 1.4 \times (34 + 10 \times (DLC + 1 + 1) + 1) \times t_{bit}$

$t_{Frame\_Maximum} = (77 + 14 \times DLC) \times t_{bit}$

If the Checksum is not sent (CHKDIS = 1):

$t_{Header\_Nominal} = 34 \times t_{bit}$

$t_{Response\_Nominal} = 10 \times NData \times t_{bit}$

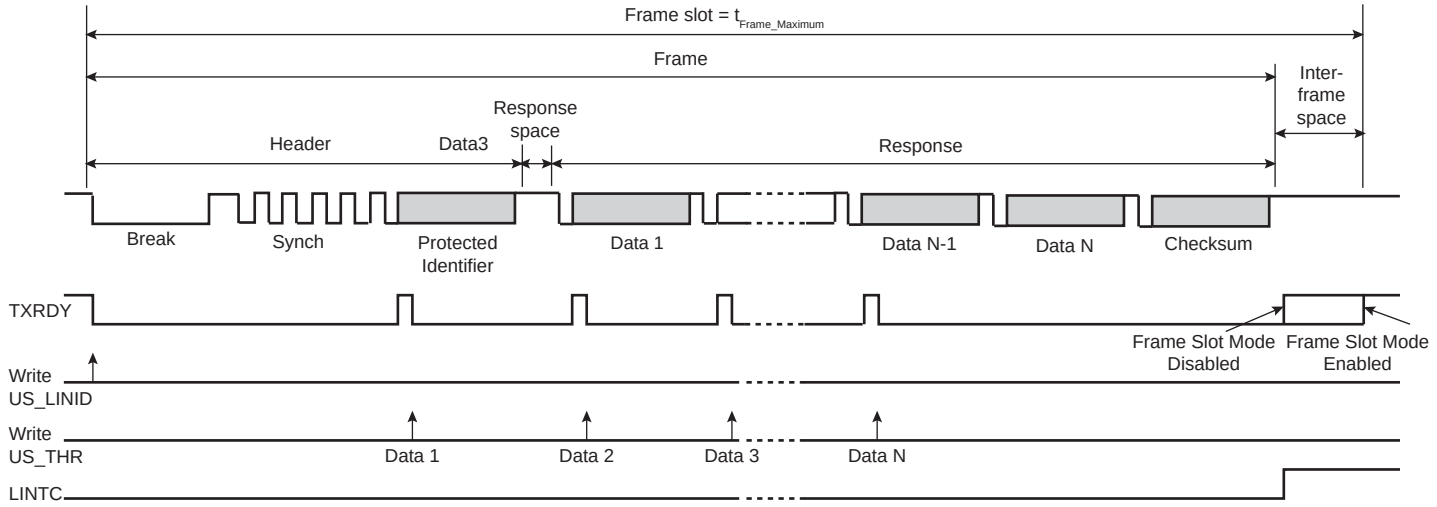
$t_{Frame\_Maximum} = 1.4 \times (t_{Header\_Nominal} + t_{Response\_Nominal} + 1)$(1)

$t_{Frame\_Maximum} = 1.4 \times (34 + 10 \times (DLC + 1) + 1) \times t_{bit}$

$t_{Frame\_Maximum} = (63 + 14 \times DLC) \times t_{bit}$

Note 1: The term “+1” leads to an integer result for $t_{Frame\_Maximum}$ (LIN Specification 1.3).

Figure 39-44: Frame Slot Mode



39.6.8.14 LIN Errors

Bit Error

This error is generated in master of slave node configuration, when the USART is transmitting and if the transmitted value on the Tx line is different from the value sampled on the Rx line. If a bit error is detected, the transmission is aborted at the next byte border.

This error is reported by flag LINBE in US_CSR.

Inconsistent Synch Field Error

This error is generated in slave node configuration, if the Synch Field character received is other than 0x55.

This error is reported by flag LINISFE in the US_CSR.

Identifier Parity Error

This error is generated in slave node configuration, if the parity of the identifier is wrong. This error can be generated only if the parity feature is enabled (PARDIS = 0).

This error is reported by flag LINIPE in the US_CSR.

Checksum Error

This error is generated in master of slave node configuration, if the received checksum is wrong. This flag can be set to 1 only if the checksum feature is enabled (CHKDIS = 0).

This error is reported by flag LINCE in the US_CSR.

Slave Not Responding Error

This error is generated in master of slave node configuration, when the USART expects a response from another node (NACT = SUBSCRIBE) but no valid message appears on the bus within the time given by the maximum length of the message frame, $t_{Frame\_Maximum}$ (see **Section 39.6.8.13 “Frame Slot Mode”**). This error is disabled if the USART does not expect any message (NACT = PUBLISH or NACT = IGNORE).

This error is reported by flag LINSNRE in the US_CSR.

39.6.8.15 LIN Frame Handling

Master Node Configuration

- Write TXEN and RXEN in US_CR to enable both the transmitter and the receiver.
- Write USART_MODE in US_MR to select the LIN mode and the master node configuration.
- Write CD and FP in US_BRGR to configure the baud rate.
- Write NACT, PARDIS, CHKDIS, CHKTYPE, DLCM, FSDIS and DLC in US_LINMR to configure the frame transfer.
- Check that TXRDY in US_CSR is set to 1
- Write IDCHR in US_LINIR to send the header

What comes next depends on the NACT configuration:

- Case 1: NACT = PUBLISH, the USART sends the response
  - Wait until TXRDY in US_CSR rises
  - Write TCHR in US_THR to send a byte
  - If all the data have not been written, redo the two previous steps
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors
- Case 2: NACT = SUBSCRIBE, the USART receives the response
  - Wait until RXRDY in US_CSR rises
  - Read RCHR in US_RHR
  - If all the data have not been read, redo the two previous steps
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors
- Case 3: NACT = IGNORE, the USART is not concerned by the response
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors

© 2017 Microchip Technology Inc.

Figure 39-45: Master Node Configuration, NACT = PUBLISH

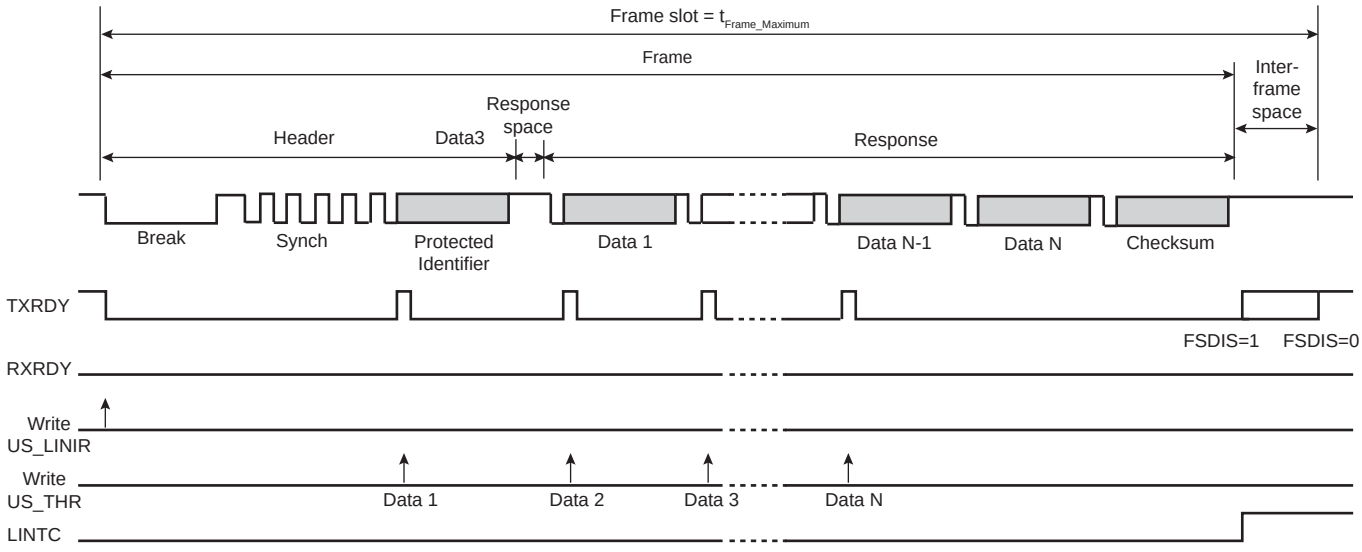


Figure 39-46: Master Node Configuration, NACT = SUBSCRIBE

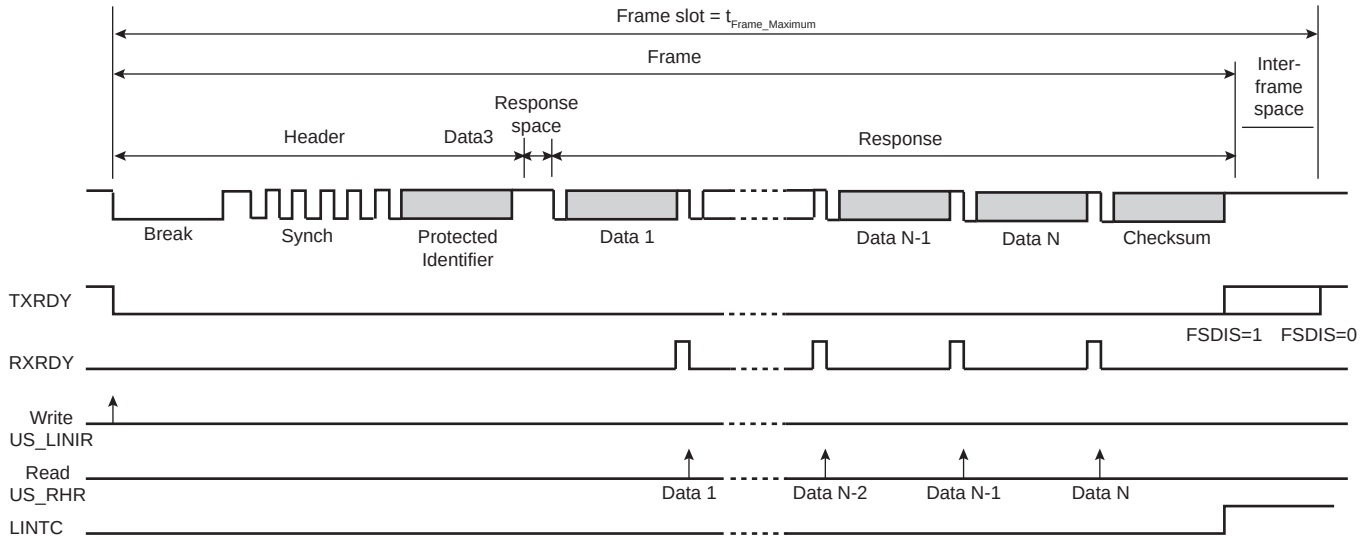
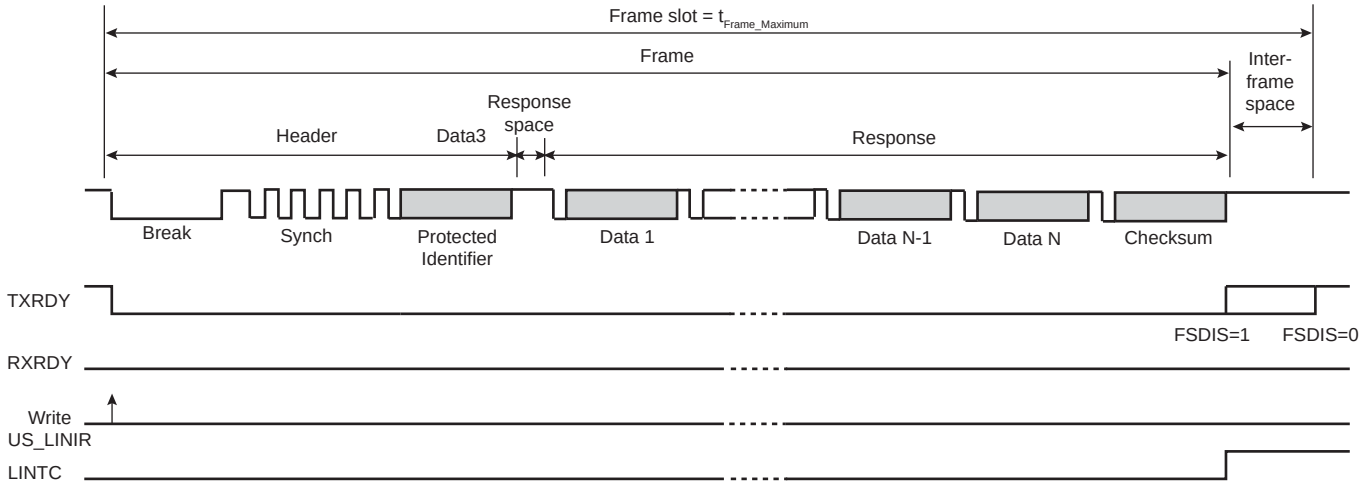


Figure 39-47: Master Node Configuration, NACT = IGNORE



Slave Node Configuration

- Write TXEN and RXEN in US_CR to enable both the transmitter and the receiver.
- Write USART_MODE in US_MR to select the LIN mode and the slave node configuration.
- Write CD and FP in US_BRGR to configure the baud rate.
- Wait until LINID in US_CSR rises
- Check LINISFE and LINPE errors
- Read IDCHR in US_RHR
- Write NACT, PARDIS, CHKDIS, CHKTYPE, DLCM and DLC in US_LINMR to configure the frame transfer.

IMPORTANT: If the NACT configuration for this frame is PUBLISH, the US_LINMR must be written with NACT = PUBLISH even if this field is already correctly configured, in order to set the TXREADY flag and the corresponding write transfer request.

What comes next depends on the NACT configuration:

- Case 1: NACT = PUBLISH, the LIN controller sends the response
  - Wait until TXRDY in US_CSR rises
  - Write TCHR in US_THR to send a byte
  - If all the data have not been written, redo the two previous steps
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors
- Case 2: NACT = SUBSCRIBE, the USART receives the response
  - Wait until RXRDY in US_CSR rises
  - Read RCHR in US_RHR
  - If all the data have not been read, redo the two previous steps
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors
- Case 3: NACT = IGNORE, the USART is not concerned by the response
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors

© 2017 Microchip Technology Inc.

Figure 39-48: Slave Node Configuration, NACT = PUBLISH

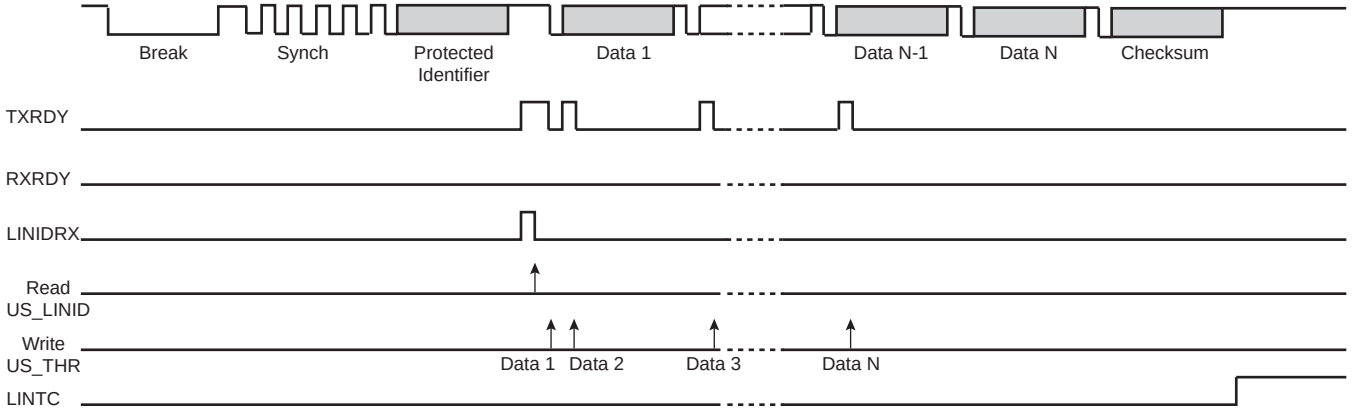


Figure 39-49: Slave Node Configuration, NACT = SUBSCRIBE

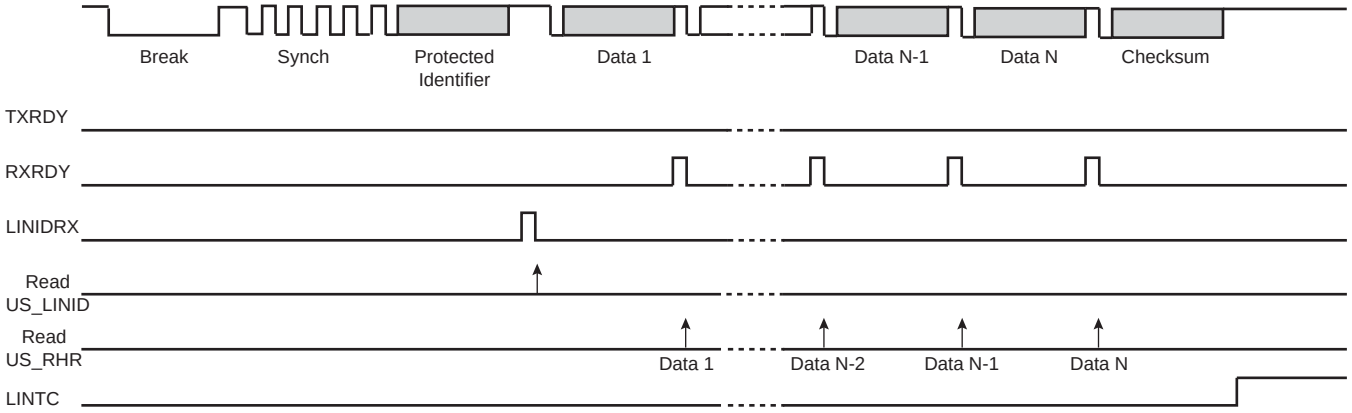
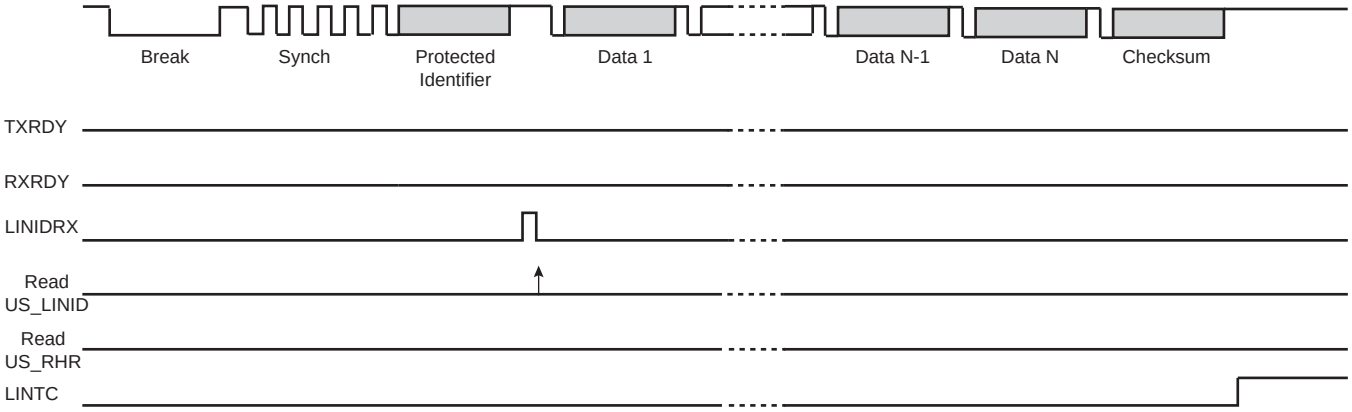


Figure 39-50: Slave Node Configuration, NACT = IGNORE



39.6.8.16 LIN Frame Handling with the DMAC

The USART can be used in association with the DMAC in order to transfer data directly into/from the on- and off-chip memories without any processor intervention.

The DMAC uses the trigger flags, TXRDY and RXRDY, to write or read into the USART. The DMAC always writes in the Transmit Holding register (US_THR) and it always reads in the Receive Holding register (US_RHR). The size of the data written or read by the DMAC in the USART is always a byte.

Master Node Configuration

The user can choose between two DMAC modes by the PDCM bit in the US_LINMR:

- PDCM = 1: the LIN configuration is stored in the WRITE buffer and it is written by the DMAC in the Transmit Holding register US_THR (instead of the LIN Mode register US_LINMR). Because the DMAC transfer size is limited to a byte, the transfer is split into two accesses. During the first access the bits, NACT, PARDIS, CHKDIS, CHKTYP, DLM and FSDIS are written. During the second access the 8-bit DLC field is written.
- PDCM = 0: the LIN configuration is not stored in the WRITE buffer and it must be written by the user in US_LINMR.

The WRITE buffer also contains the Identifier and the DATA, if the USART sends the response (NACT = PUBLISH).

The READ buffer contains the DATA if the USART receives the response (NACT = SUBSCRIBE).

Figure 39-51: Master Node with DMAC (PDCM = 1)

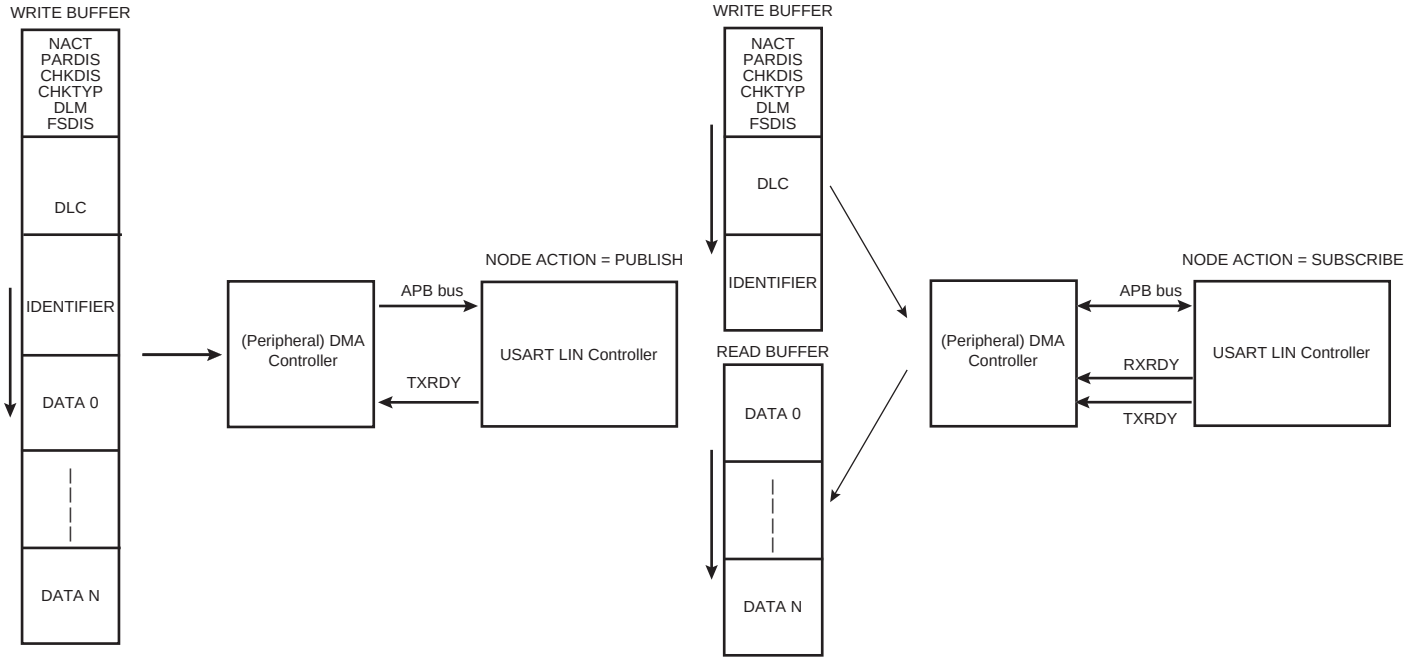
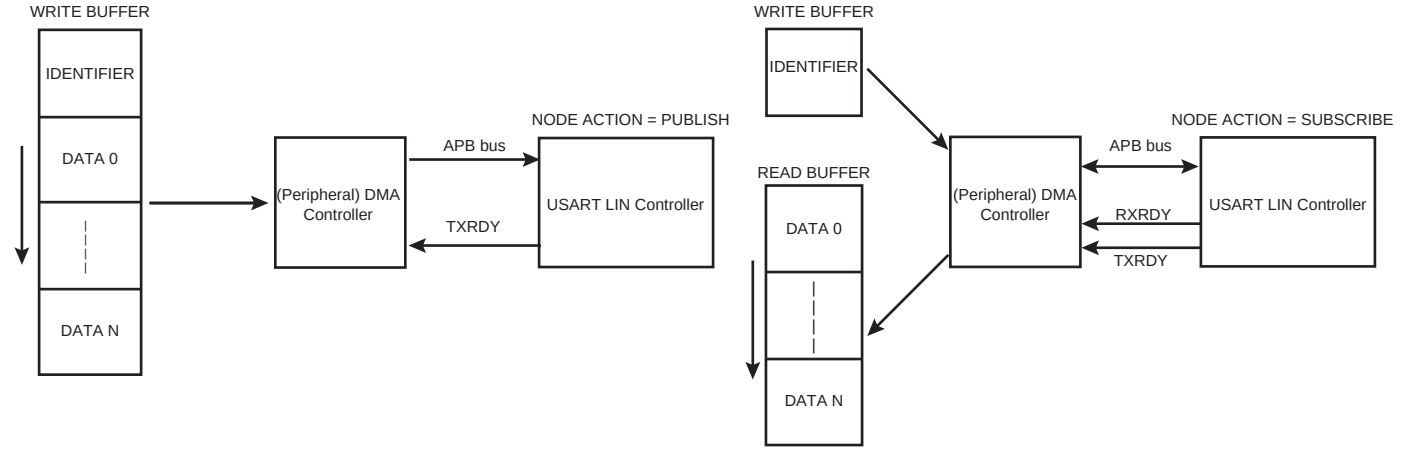


Figure 39-52: Master Node with DMAC (PDCM = 0)



© 2017 Microchip Technology Inc.

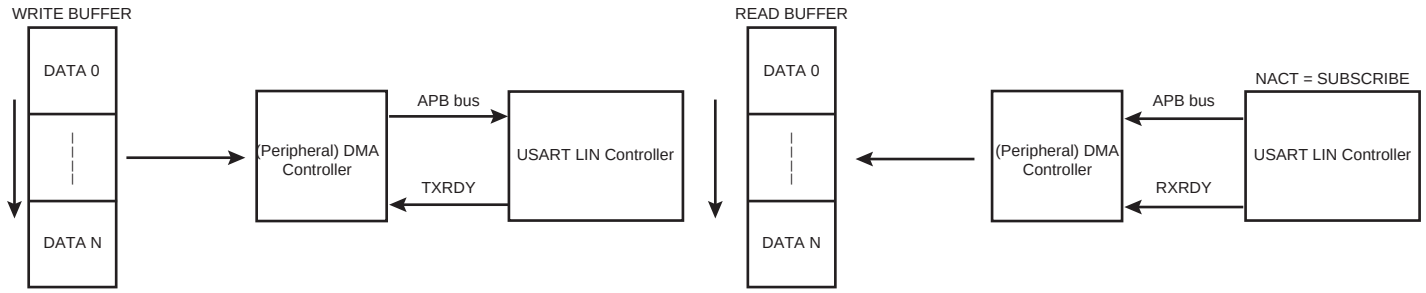
Slave Node Configuration

In this configuration, the DMAC transfers only the DATA. The Identifier must be read by the user in the LIN Identifier register (US_LINIR). The LIN mode must be written by the user in US_LINMR.

The WRITE buffer contains the DATA if the USART sends the response (NACT = PUBLISH).

The READ buffer contains the DATA if the USART receives the response (NACT = SUBSCRIBE).

Figure 39-53: Slave Node with DMAC



39.6.8.17 Wake-up Request

Any node in a sleeping LIN cluster may request a wake-up.

In the LIN 2.0 specification, the wakeup request is issued by forcing the bus to the dominant state from 250 µs to 5 ms. For this, it is necessary to send the character 0xF0 in order to impose five successive dominant bits. Whatever the baud rate is, this character complies with the specified timings.

- Baud rate min = 1 kbit/s $\rightarrow$ t_{bit} = 1 ms $\rightarrow$ 5 t_{bit} = 5 ms
- Baud rate max = 20 kbit/s $\rightarrow$ t_{bit} = 50 µs $\rightarrow$ 5 t_{bit} = 250 µs

In the LIN 1.3 specification, the wakeup request should be generated with the character 0x80 in order to impose eight successive dominant bits.

The user can choose by the WKUPTYP bit in US_LINMR either to send a LIN 2.0 wakeup request (WKUPTYP = 0) or to send a LIN 1.3 wakeup request (WKUPTYP = 1).

A wake-up request is transmitted by writing a 1 to the LINWKUP bit in the US_CR. Once the transfer is completed, the LINTC flag is asserted in the Status register (US_SR). It is cleared by writing a 1 to the RSTSTA bit in the US_CR.

39.6.8.18 Bus Idle Time-out

If the LIN bus is inactive for a certain duration, the slave nodes shall automatically enter in Sleep mode. In the LIN 2.0 specification, this time-out is fixed at 4 seconds. In the LIN 1.3 specification, it is fixed at 25,000 t_{bit}.

In slave Node configuration, the receiver time-out detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in US_CSR rises and can generate an interrupt, thus indicating to the driver to go into Sleep mode.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of US_RTOR. If a 0 is written to the TO field, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in US_CSR remains at 0. Otherwise, the receiver loads a 17-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the US_CSR rises.

If STTTO is performed, the counter clock is stopped until a first character is received.

If RETTO is performed, the counter starts counting down immediately from the value TO.

Table 39-15: Receiver Time-out Programming

LIN Specification	Baud Rate	Time-out period	US_RTOR.TO
2.0	1,000 bit/s	4 s	4,000
	2,400 bit/s		9,600
	9,600 bit/s		38,400
	19,200 bit/s		76,800
	20,000 bit/s		80,000
1.3	–	25,000 t_{bit}	25,000

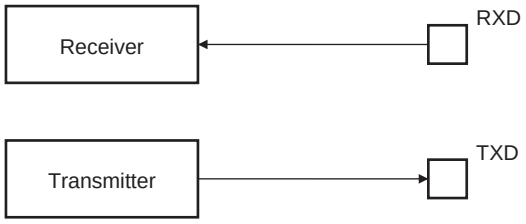
39.6.9 Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In Loopback mode, the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

39.6.9.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

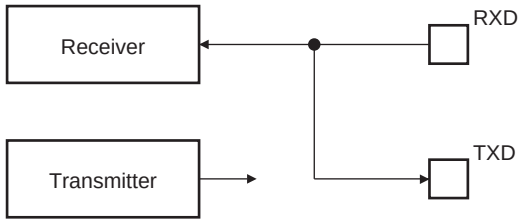
Figure 39-54: Normal Mode Configuration



39.6.9.2 Automatic Echo Mode

Automatic echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is sent to the TXD pin, as shown in Figure 39-55. Programming the transmitter has no effect on the TXD pin. The RXD pin is still connected to the receiver input, thus the receiver remains active.

Figure 39-55: Automatic Echo Mode Configuration

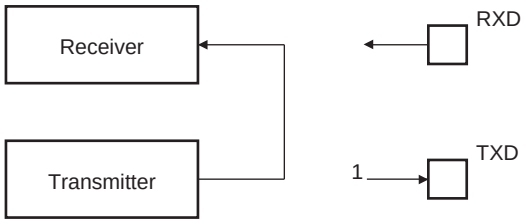


39.6.9.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 39-56. The TXD and RXD pins are not used. The RXD pin has no effect on the receiver and the TXD pin is continuously driven high, as in idle state.

© 2017 Microchip Technology Inc.

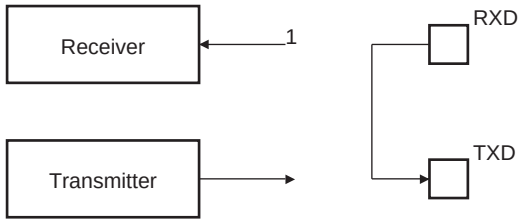
Figure 39-56: Local Loopback Mode Configuration



39.6.9.4 Remote Loopback Mode

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 39-57. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 39-57: Remote Loopback Mode Configuration



39.6.10 Register Write Protection

To prevent any single software error from corrupting USART behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the USART Write Protection Mode Register (US_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the USART Write Protection Status Register (US_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the US_WPSR.

The following registers can be write-protected:

- USART Mode Register
- USART Baud Rate Generator Register
- USART Receiver Time-out Register
- USART Transmitter Timeguard Register
- USART FI DI RATIO Register
- USART IrDA Filter Register
- USART Manchester Configuration Register

39.7 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface

Table 39-16: Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Control Register	US_CR	Write-only	–
0x0004	Mode Register	US_MR	Read/Write	0x0
0x0008	Interrupt Enable Register	US_IER	Write-only	–
0x000C	Interrupt Disable Register	US_IDR	Write-only	–
0x0010	Interrupt Mask Register	US_IMR	Read-only	0x0
0x0014	Channel Status Register	US_CSR	Read-only	0x0
0x0018	Receive Holding Register	US_RHR	Read-only	0x0
0x001C	Transmit Holding Register	US_THR	Write-only	–
0x0020	Baud Rate Generator Register	US_BRGR	Read/Write	0x0
0x0024	Receiver Time-out Register	US_RTOR	Read/Write	0x0
0x0028	Transmitter Timeguard Register	US_TTGR	Read/Write	0x0
0x002C–0x003C	Reserved	–	–	–
0x0040	FI DI Ratio Register	US_FIDI	Read/Write	0x174
0x0044	Number of Errors Register	US_NER	Read-only	0x0
0x0048	Reserved	–	–	–
0x004C	IrDA Filter Register	US_IF	Read/Write	0x0
0x0050	Manchester Configuration Register	US_MAN	Read/Write	0x30011004
0x0054	LIN Mode Register	US_LINMR	Read/Write	0x0
0x0058	LIN Identifier Register	US_LINIR	Read/Write(1)	0x0
0x005C	LIN Baud Rate Register	US_LINBRR	Read-only	0x0
0x0060–0x00E0	Reserved	–	–	–
0x00E4	Write Protection Mode Register	US_WPMR	Read/Write	0x0
0x00E8	Write Protection Status Register	US_WPSR	Read-only	0x0
0x00EC–0x00FC	Reserved	–	–	–

Note 1: Write is possible only in LIN master node configuration.

© 2017 Microchip Technology Inc.

39.7.1 USART Control Register

Name:US_CR

Address:0xF801C000 (0), 0xF8020000 (1), 0xF8024000 (2), 0xF8028000 (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	LINWKUP	LINABT	RTSDIS	RTSEN	–	–
15	14	13	12	11	10	9	8
RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA
7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

For SPI control, see Section 39.7.2 "USART Control Register (SPI_MODE)".

RSTRX: Reset Receiver

0: No effect.

1: Resets the receiver.

RSTTX: Reset Transmitter

0: No effect.

1: Resets the transmitter.

RXEN: Receiver Enable

0: No effect.

1: Enables the receiver, if RXDIS is 0.

RXDIS: Receiver Disable

0: No effect.

1: Disables the receiver.

TXEN: Transmitter Enable

0: No effect.

1: Enables the transmitter if TXDIS is 0.

TXDIS: Transmitter Disable

0: No effect.

1: Disables the transmitter.

RSTSTA: Reset Status Bits

0: No effect.

1: Resets the status bits PARE, FRAME, OVRE, MANERR, LINBE, LINISFE, LINIPE, LINCE, LINSNRE, LINID, LINTC, LINBK and RXBRK in US_CSR.

STTBRK: Start Break

0: No effect.

1: Starts transmission of a break after the characters present in US_THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.

STPBRK: Stop Break

0: No effect.

1: Stops transmission of the break after a minimum of one character length and transmits a high level during 12-bit periods. No effect if no break is being transmitted.

STTTO: Clear TIMEOUT Flag and Start Time-out After Next Character Received

0: No effect.

1: Starts waiting for a character before enabling the time-out counter. Immediately disables a time-out period in progress. Resets the status bit TIMEOUT in US_CSR.

SENDA: Send Address

0: No effect.

1: In Multidrop mode only, the next character written to the US_THR is sent with the address bit set.

RSTIT: Reset Iterations

0: No effect.

1: Resets ITER in US_CSR. No effect if the ISO7816 is not enabled.

RSTNACK: Reset Non Acknowledge

0: No effect

1: Resets NACK in US_CSR.

RETTO: Start Time-out Immediately

0: No effect

1: Immediately restarts time-out period.

RTSEN: Request to Send Pin Control

0: No effect.

1: Drives RTS pin to 0 if US_MR.USART_MODE field = 0.

RTSDIS: Request to Send Pin Control

0: No effect.

1: Drives RTS pin to 1 if US_MR.USART_MODE field = 0.

LINABT: Abort LIN Transmission

0: No effect.

1: Abort the current LIN transmission.

LINWKUP: Send LIN Wakeup Signal

0: No effect.

1: Sends a wakeup signal on the LIN bus.

© 2017 Microchip Technology Inc.

39.7.2 USART Control Register (SPI_MODE)

Name:US_CR (SPI_MODE)

Address:0xF801C000 (0), 0xF8020000 (1), 0xF8024000 (2), 0xF8028000 (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RCS	FCS	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RSTSTA
7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

RSTRX: Reset Receiver

0: No effect.

1: Resets the receiver.

RSTTX: Reset Transmitter

0: No effect.

1: Resets the transmitter.

RXEN: Receiver Enable

0: No effect.

1: Enables the receiver, if RXDIS is 0.

RXDIS: Receiver Disable

0: No effect.

1: Disables the receiver.

TXEN: Transmitter Enable

0: No effect.

1: Enables the transmitter if TXDIS is 0.

TXDIS: Transmitter Disable

0: No effect.

1: Disables the transmitter.

RSTSTA: Reset Status Bits

0: No effect.

1: Resets the status bits OVRE, UNRE in US_CSR.

FCS: Force SPI Chip Select

Applicable if USART operates in SPI master mode (USART_MODE = 0xE):

0: No effect.

1: Forces the Slave Select Line NSS (RTS pin) to 0, even if USART is not transmitting, in order to address SPI slave devices supporting the CSAAT mode (Chip Select Active After Transfer).

© 2017 Microchip Technology Inc.

RCS: Release SPI Chip Select

Applicable if USART operates in SPI master mode (USART_MODE = 0xE):

0: No effect.

1: Releases the Slave Select Line NSS (RTS pin).

© 2017 Microchip Technology Inc.

39.7.3 USART Mode Register

Name:US_MR

Address:0xF801C004 (0), 0xF8020004 (1), 0xF8024004 (2), 0xF8028004 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
ONEBIT	MODSYNC	MAN	FILTER	–	MAX_ITERATION		

23	22	21	20	19	18	17	16
INVDATA	VAR_SYNC	DSNACK	INACK	OVER	CLKO	MODE9	MSBF

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC

7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

For SPI configuration, see Section 39.7.4 ”USART Mode Register (SPI_MODE)”.

USART_MODE: USART Mode of Operation

Value	Name	Description
0x0	NORMAL	Normal mode
0x1	RS485	RS485
0x2	HW_HANDSHAKING	Hardware Handshaking
0x3	—	Reserved
0x4	IS07816_T_0	IS07816 Protocol: T = 0
0x6	IS07816_T_1	IS07816 Protocol: T = 1
0x8	IRDA	IrDA
0xA	LIN_MASTER	LIN master
0xB	LIN_SLAVE	LIN Slave
0xE	SPI_MASTER	SPI master
0xF	SPI_SLAVE	SPI Slave

USCLKS: Clock Selection

Value	Name	Description
0	MCK	Peripheral clock is selected
1	DIV	Peripheral clock divided (DIV = 8) is selected
2	—	Reserved
3	SCK	Serial clock (SCK) is selected

© 2017 Microchip Technology Inc.

CHRL: Character Length

Value	Name	Description
0	5_BIT	Character length is 5 bits
1	6_BIT	Character length is 6 bits
2	7_BIT	Character length is 7 bits
3	8_BIT	Character length is 8 bits

SYNC: Synchronous Mode Select

0: USART operates in Asynchronous mode.

1: USART operates in Synchronous mode.

PAR: Parity Type

Value	Name	Description
0	EVEN	Even parity
1	ODD	Odd parity
2	SPACE	Parity forced to 0 (Space)
3	MARK	Parity forced to 1 (Mark)
4	NO	No parity
6	MULTIDROP	Multidrop mode

NBSTOP: Number of Stop Bits

Value	Name	Description
0	1_BIT	1 stop bit
1	1_5_BIT	1.5 stop bit (SYNC = 0) or reserved (SYNC = 1)
2	2_BIT	2 stop bits

CHMODE: Channel Mode

Value	Name	Description
0	NORMAL	Normal mode
1	AUTOMATIC	Automatic Echo. Receiver input is connected to the TXD pin.
2	LOCAL_LOOPBACK	Local Loopback. Transmitter output is connected to the Receiver Input.
3	REMOTE_LOOPBACK	Remote Loopback. RXD pin is internally connected to the TXD pin.

MSBF: Bit Order

0: Least significant bit is sent/received first.

1: Most significant bit is sent/received first.

MODE9: 9-bit Character Length

0: CHRL defines character length

1: 9-bit character length

CLKO: Clock Output Select

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if USCLKS does not select the external clock SCK.

© 2017 Microchip Technology Inc.

OVER: Oversampling Mode

0: 16 × Oversampling

1: 8 × Oversampling

INACK: Inhibit Non Acknowledge

0: The NACK is generated.

1: The NACK is not generated.

DSNACK: Disable Successive NACK

0: NACK is sent on the ISO line as soon as a parity error occurs in the received character (unless INACK is set).

1: Successive parity errors are counted up to the value specified in the MAX_ITERATION field. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line. The flag ITER is asserted.

Note: MAX_ITERATION field must be set to 0 if DSNACK is cleared.

INVDATA: Inverted Data

0: The data field transmitted on TXD line is the same as the one written in US_THR or the content read in US_RHR is the same as RXD line. Normal mode of operation.

1: The data field transmitted on TXD line is inverted (voltage polarity only) compared to the value written on US_THR or the content read in US_RHR is inverted compared to what is received on RXD line (or ISO7816 IO line). Inverted mode of operation, useful for contactless card application. To be used with configuration bit MSBF.

VAR_SYNC: Variable Synchronization of Command/Data Sync Start Frame Delimiter

0: User defined configuration of command or data sync field depending on MODSYNC value.

1: The sync field is updated when a character is written into US_THR.

MAX_ITERATION: Maximum Number of Automatic Iteration

0–7: Defines the maximum number of iterations in mode ISO7816, protocol T = 0.

FILTER: Receive Line Filter

0: The USART does not filter the receive line.

1: The USART filters the receive line using a three-sample filter (1/16-bit clock) (2 over 3 majority).

MAN: Manchester Encoder/Decoder Enable

0: Manchester encoder/decoder are disabled.

1: Manchester encoder/decoder are enabled.

MODSYNC: Manchester Synchronization Mode

0:The Manchester start bit is a 0 to 1 transition

1: The Manchester start bit is a 1 to 0 transition.

ONEBIT: Start Frame Delimiter Selector

0: Start frame delimiter is COMMAND or DATA SYNC.

1: Start frame delimiter is one bit.

39.7.4 USART Mode Register (SPI_MODE)

Name:US_MR (SPI_MODE)

Address:0xF801C004 (0), 0xF8020004 (1), 0xF8024004 (2), 0xF8028004 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	WRDBT	–	CLKO	–	CPOL
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	CPHA
7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

USART_MODE: USART Mode of Operation

Value	Name	Description
0xE	SPI_MASTER	SPI master
0xF	SPI_SLAVE	SPI Slave

USCLKS: Clock Selection

Value	Name	Description
0	MCK	Peripheral clock is selected
1	DIV	Peripheral clock divided (DIV = 8) is selected
3	SCK	Serial Clock SLK is selected

CHRL: Character Length

Value	Name	Description
3	8_BIT	Character length is 8 bits

CPHA: SPI Clock Phase

– Applicable if USART operates in SPI mode (USART_MODE = 0xE or 0xF):

0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

CPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

CPOL: SPI Clock Polarity

Applicable if USART operates in SPI mode (slave or master, USART_MODE = 0xE or 0xF):

0: The inactive state value of SPCK is logic level zero.

1: The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with CPHA to produce the required clock/data relationship between master and slave devices.

© 2017 Microchip Technology Inc.

CLKO: Clock Output Select

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if USCLKS does not select the external clock SCK.

WRDBT: Wait Read Data Before Transfer

0: The character transmission starts as soon as a character is written into US_THR (assuming TXRDY was set).

1: The character transmission starts when a character is written and only if RXRDY flag is cleared (Receive Holding Register has been read).

39.7.5 USART Interrupt Enable Register

Name:US_IER

Address:0xF801C008 (0), 0xF8020008 (1), 0xF8024008 (2), 0xF8028008 (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	MANE
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	–	–	–
15	14	13	12	11	10	9	8
–	–	NACK	–	–	ITER	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

For SPI specific configuration, see Section 39.7.6 ”USART Interrupt Enable Register (SPI_MODE)”.

For LIN specific configuration, see Section 39.7.7 ”USART Interrupt Enable Register (LIN_MODE)”.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt.

RXRDY: RXRDY Interrupt Enable

TXRDY: TXRDY Interrupt Enable

RXBRK: Receiver Break Interrupt Enable

OVRE: Overrun Error Interrupt Enable

FRAME: Framing Error Interrupt Enable

PARE: Parity Error Interrupt Enable

TIMEOUT: Time-out Interrupt Enable

TXEMPTY: TXEMPTY Interrupt Enable

ITER: Max number of Repetitions Reached Interrupt Enable

NACK: Non Acknowledge Interrupt Enable

CTSIC: Clear to Send Input Change Interrupt Enable

MANE: Manchester Error Interrupt Enable

© 2017 Microchip Technology Inc.

39.7.6 USART Interrupt Enable Register (SPI_MODE)

Name:US_IER (SPI_MODE)

Address:0xF801C008 (0), 0xF8020008 (1), 0xF8024008 (2), 0xF8028008 (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	NSSE	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
–	–	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt.

RXRDY: RXRDY Interrupt Enable

TXRDY: TXRDY Interrupt Enable

OVRE: Overrun Error Interrupt Enable

TXEMPTY: TXEMPTY Interrupt Enable

UNRE: SPI Underrun Error Interrupt Enable

NSSE: NSS Line (Driving CTS Pin) Rising or Falling Edge Event Interrupt Enable

© 2017 Microchip Technology Inc.

39.7.7 USART Interrupt Enable Register (LIN_MODE)

Name:US_IER (LIN_MODE)

Address:0xF801C008 (0), 0xF8020008 (1), 0xF8024008 (2), 0xF8028008 (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt.

RXRDY: RXRDY Interrupt Enable

TXRDY: TXRDY Interrupt Enable

OVRE: Overrun Error Interrupt Enable

FRAME: Framing Error Interrupt Enable

PARE: Parity Error Interrupt Enable

TIMEOUT: Time-out Interrupt Enable

TXEMPTY: TXEMPTY Interrupt Enable

LINBK: LIN Break Sent or LIN Break Received Interrupt Enable

LINID: LIN Identifier Sent or LIN Identifier Received Interrupt Enable

LINTC: LIN Transfer Completed Interrupt Enable

LINBE: LIN Bus Error Interrupt Enable

LINISFE: LIN Inconsistent Synch Field Error Interrupt Enable

LINIPE: LIN Identifier Parity Interrupt Enable

LINCE: LIN Checksum Error Interrupt Enable

LINSNRE: LIN Slave Not Responding Error Interrupt Enable

© 2017 Microchip Technology Inc.

39.7.8 USART Interrupt Disable Register

Name:US_IDR

Address:0xF801C00C (0), 0xF802000C (1), 0xF802400C (2), 0xF802800C (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	MANE
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	–	–	–
15	14	13	12	11	10	9	8
–	–	NACK	–	–	ITER	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

For SPI specific configuration, see Section 39.7.9 ”USART Interrupt Disable Register (SPI_MODE)”.

For LIN specific configuration, see Section 39.7.10 ”USART Interrupt Disable Register (LIN_MODE)”.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt.

RXRDY: RXRDY Interrupt Disable

TXRDY: TXRDY Interrupt Disable

RXBRK: Receiver Break Interrupt Disable

OVRE: Overrun Error Interrupt Enable

FRAME: Framing Error Interrupt Disable

PARE: Parity Error Interrupt Disable

TIMEOUT: Time-out Interrupt Disable

TXEMPTY: TXEMPTY Interrupt Disable

ITER: Max Number of Repetitions Reached Interrupt Disable

NACK: Non Acknowledge Interrupt Disable

CTSIC: Clear to Send Input Change Interrupt Disable

MANE: Manchester Error Interrupt Disable

© 2017 Microchip Technology Inc.

39.7.9 USART Interrupt Disable Register (SPI_MODE)

Name:US_IDR (SPI_MODE)

Address:0xF801C00C (0), 0xF802000C (1), 0xF802400C (2), 0xF802800C (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	NSSE	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
–	–	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt.

RXRDY: RXRDY Interrupt Disable

TXRDY: TXRDY Interrupt Disable

OVRE: Overrun Error Interrupt Disable

TXEMPTY: TXEMPTY Interrupt Disable

UNRE: SPI Underrun Error Interrupt Disable

NSSE: NSS Line (Driving CTS Pin) Rising or Falling Edge Event Interrupt Disable

© 2017 Microchip Technology Inc.

39.7.10 USART Interrupt Disable Register (LIN_MODE)

Name:US_IDR (LIN_MODE)

Address:0xF801C00C (0), 0xF802000C (1), 0xF802400C (2), 0xF802800C (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt.

RXRDY: RXRDY Interrupt Disable

TXRDY: TXRDY Interrupt Disable

OVRE: Overrun Error Interrupt Disable

FRAME: Framing Error Interrupt Disable

PARE: Parity Error Interrupt Disable

TIMEOUT: Time-out Interrupt Disable

TXEMPTY: TXEMPTY Interrupt Disable

LINBK: LIN Break Sent or LIN Break Received Interrupt Disable

LINID: LIN Identifier Sent or LIN Identifier Received Interrupt Disable

LINTC: LIN Transfer Completed Interrupt Disable

LINBE: LIN Bus Error Interrupt Disable

LINISFE: LIN Inconsistent Synch Field Error Interrupt Disable

LINIPE: LIN Identifier Parity Interrupt Disable

LINCE: LIN Checksum Error Interrupt Disable

LINSNRE: LIN Slave Not Responding Error Interrupt Disable

© 2017 Microchip Technology Inc.

39.7.11 USART Interrupt Mask Register

Name:US_IMR

Address:0xF801C010 (0), 0xF8020010 (1), 0xF8024010 (2), 0xF8028010 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	MANE
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	–	–	–
15	14	13	12	11	10	9	8
–	–	NACK	–	–	ITER	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

For SPI specific configuration, see Section 39.7.12 ”USART Interrupt Mask Register (SPI_MODE)”.

For LIN specific configuration, see Section 39.7.13 ”USART Interrupt Mask Register (LIN_MODE)”.

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

RXRDY: RXRDY Interrupt Mask

TXRDY: TXRDY Interrupt Mask

RXBRK: Receiver Break Interrupt Mask

OVRE: Overrun Error Interrupt Mask

FRAME: Framing Error Interrupt Mask

PARE: Parity Error Interrupt Mask

TIMEOUT: Time-out Interrupt Mask

TXEMPTY: TXEMPTY Interrupt Mask

ITER: Max Number of Repetitions Reached Interrupt Mask

NACK: Non Acknowledge Interrupt Mask

CTSIC: Clear to Send Input Change Interrupt Mask

MANE: Manchester Error Interrupt Mask

© 2017 Microchip Technology Inc.

39.7.12 USART Interrupt Mask Register (SPI_MODE)

Name:US_IMR (SPI_MODE)

Address:0xF801C010 (0), 0xF8020010 (1), 0xF8024010 (2), 0xF8028010 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	NSSE	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
–	–	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

RXRDY: RXRDY Interrupt Mask

TXRDY: TXRDY Interrupt Mask

OVRE: Overrun Error Interrupt Mask

TXEMPTY: TXEMPTY Interrupt Mask

UNRE: SPI Underrun Error Interrupt Mask

NSSE: NSS Line (Driving CTS Pin) Rising or Falling Edge Event Interrupt Mask

© 2017 Microchip Technology Inc.

39.7.13 USART Interrupt Mask Register (LIN_MODE)

Name:US_IMR (LIN_MODE)

Address:0xF801C010 (0), 0xF8020010 (1), 0xF8024010 (2), 0xF8028010 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

RXRDY: RXRDY Interrupt Mask

TXRDY: TXRDY Interrupt Mask

OVRE: Overrun Error Interrupt Mask

FRAME: Framing Error Interrupt Mask

PARE: Parity Error Interrupt Mask

TIMEOUT: Time-out Interrupt Mask

TXEMPTY: TXEMPTY Interrupt Mask

LINBK: LIN Break Sent or LIN Break Received Interrupt Mask

LINID: LIN Identifier Sent or LIN Identifier Received Interrupt Mask

LINTC: LIN Transfer Completed Interrupt Mask

LINBE: LIN Bus Error Interrupt Mask

LINISFE: LIN Inconsistent Synch Field Error Interrupt Mask

LINIPE: LIN Identifier Parity Interrupt Mask

LINCE: LIN Checksum Error Interrupt Mask

LINSNRE: LIN Slave Not Responding Error Interrupt Mask

© 2017 Microchip Technology Inc.

39.7.14 USART Channel Status Register

Name:US_CSR

Address:0xF801C014 (0), 0xF8020014 (1), 0xF8024014 (2), 0xF8028014 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	MANERR
23	22	21	20	19	18	17	16
CTS	–	–	–	CTSIC	–	–	–
15	14	13	12	11	10	9	8
–	–	NACK	–	–	ITER	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

For SPI specific configuration, see Section 39.7.15 ”USART Channel Status Register (SPI_MODE)”.

For LIN specific configuration, see Section 39.7.16 ”USART Channel Status Register (LIN_MODE)”.

RXRDY: Receiver Ready (cleared by reading US_RHR)

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

TXRDY: Transmitter Ready (cleared by writing US_THR)

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested, or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

RXBRK: Break Received/End of Break (cleared by writing a one to bit US_CR.RSTSTA)

0: No break received or end of break detected since the last RSTSTA.

1: Break received or end of break detected since the last RSTSTA.

OVRE: Overrun Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

FRAME: Framing Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No stop bit has been detected low since the last RSTSTA.

1: At least one stop bit has been detected low since the last RSTSTA.

PARE: Parity Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No parity error has been detected since the last RSTSTA.

1: At least one parity error has been detected since the last RSTSTA.

TIMEOUT: Receiver Time-out (cleared by writing a one to bit US_CR.STTTO)

0: There has not been a time-out since the last Start Time-out command (STTTO in US_CR) or the Time-out Register is 0.

1: There has been a time-out since the last Start Time-out command (STTTO in US_CR).

TXEMPTY: Transmitter Empty (cleared by writing US_THR)

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There are no characters in US_THR, nor in the Transmit Shift Register.

© 2017 Microchip Technology Inc.

ITER: Max Number of Repetitions Reached (cleared by writing a one to bit US_CR.RSTIT)

0: Maximum number of repetitions has not been reached since the last RSTIT.

1: Maximum number of repetitions has been reached since the last RSTIT.

NACK: Non Acknowledge Interrupt (cleared by writing a one to bit US_CR.RSTNACK)

0: Non acknowledge has not been detected since the last RSTNACK.

1: At least one non acknowledge has been detected since the last RSTNACK.

CTSIC: Clear to Send Input Change Flag (cleared on read)

0: No input change has been detected on the CTS pin since the last read of US_CSR.

1: At least one input change has been detected on the CTS pin since the last read of US_CSR.

CTS: Image of CTS Input

0: CTS input is driven low.

1: CTS input is driven high.

MANERR: Manchester Error (cleared by writing a one to the bit US_CR.RSTSTA)

0: No Manchester error has been detected since the last RSTSTA.

1: At least one Manchester error has been detected since the last RSTSTA.

© 2017 Microchip Technology Inc.

39.7.15 USART Channel Status Register (SPI_MODE)

Name:US_CSR (SPI_MODE)

Address:0xF801C014 (0), 0xF8020014 (1), 0xF8024014 (2), 0xF8028014 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NSS	–	–	–	NSSE	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
–	–	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

RXRDY: Receiver Ready (cleared by reading US_RHR)

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

TXRDY: Transmitter Ready (cleared by writing US_THR)

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

OVRE: Overrun Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

TXEMPTY: Transmitter Empty (cleared by writing US_THR)

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There are no characters in US_THR, nor in the Transmit Shift Register.

UNRE: Underrun Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No SPI underrun error has occurred since the last RSTSTA.

1: At least one SPI underrun error has occurred since the last RSTSTA.

NSSE: NSS Line (Driving CTS Pin) Rising or Falling Edge Event (cleared on read)

0: No NSS line event has been detected since the last read of US_CSR.

1: A rising or falling edge event has been detected on NSS line since the last read of US_CSR .

NSS: Image of NSS Line

0: NSS line is driven low (if NSSE = 1, falling edge occurred on NSS line).

1: NSS line is driven high (if NSSE = 1, rising edge occurred on NSS line).

39.7.16 USART Channel Status Register (LIN_MODE)

Name:US_CSR (LIN_MODE)

Address:0xF801C014 (0), 0xF8020014 (1), 0xF8024014 (2), 0xF8028014 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
LINBLS	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

RXRDY: Receiver Ready (cleared by reading US_THR)

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

TXRDY: Transmitter Ready (cleared by writing US_THR)

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

OVRE: Overrun Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

FRAME: Framing Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No stop bit has been detected low since the last RSTSTA.

1: At least one stop bit has been detected low since the last RSTSTA.

PARE: Parity Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No parity error has been detected since the last RSTSTA.

1: At least one parity error has been detected since the last RSTSTA.

TIMEOUT: Receiver Time-out (cleared by writing a one to bit US_CR.RSTSTA)

0: There has not been a time-out since the last start time-out command (STTTO in US_CR) or the Time-out Register is 0.

1: There has been a time-out since the last start time-out command (STTTO in US_CR).

© 2017 Microchip Technology Inc.

TXEMPTY: Transmitter Empty (cleared by writing US_THR)

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There are no characters in US_THR, nor in the Transmit Shift Register.

LINBK: LIN Break Sent or LIN Break Received (cleared by writing a one to bit US_CR.RSTSTA)

Applicable if USART operates in LIN master mode (USART_MODE = 0xA):

0: No LIN break has been sent since the last RSTSTA.

1:At least one LIN break has been sent since the last RSTSTA

If USART operates in LIN slave mode (USART_MODE = 0xB):

0: No LIN break has received sent since the last RSTSTA.

1:At least one LIN break has been received since the last RSTSTA.

LINID: LIN Identifier Sent or LIN Identifier Received (cleared by writing a one to bit US_CR.RSTSTA)

If USART operates in LIN master mode (USART_MODE = 0xA):

0: No LIN identifier has been sent since the last RSTSTA.

1:At least one LIN identifier has been sent since the last RSTSTA.

If USART operates in LIN slave mode (USART_MODE = 0xB):

0: No LIN identifier has been received since the last RSTSTA.

1:At least one LIN identifier has been received since the last RSTSTA

LINTC: LIN Transfer Completed (cleared by writing a one to bit US_CR.RSTSTA)

0: The USART is idle or a LIN transfer is ongoing.

1: A LIN transfer has been completed since the last RSTSTA.

LINBLS: LIN Bus Line Status

0: LIN bus line is set to 0.

1: LIN bus line is set to 1.

LINBE: LIN Bit Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No bit error has been detected since the last RSTSTA.

1: A bit error has been detected since the last RSTSTA.

LINISFE: LIN Inconsistent Synch Field Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No LIN inconsistent synch field error has been detected since the last RSTSTA

1: The USART is configured as a slave node and a LIN Inconsistent synch field error has been detected since the last RSTSTA.

LINIPE: LIN Identifier Parity Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No LIN identifier parity error has been detected since the last RSTSTA.

1: A LIN identifier parity error has been detected since the last RSTSTA.

LINCE: LIN Checksum Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No LIN checksum error has been detected since the last RSTSTA.

1: A LIN checksum error has been detected since the last RSTSTA.

LINSNRE: LIN Slave Not Responding Error (cleared by writing a one to bit US_CR.RSTSTA)

0: No LIN slave not responding error has been detected since the last RSTSTA.

1: A LIN slave not responding error has been detected since the last RSTSTA.

© 2017 Microchip Technology Inc.

39.7.17 USART Receive Holding Register

Name:US_RHR

Address:0xF801C018 (0), 0xF8020018 (1), 0xF8024018 (2), 0xF8028018 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RXSYNH	–	–	–	–	–	–	RXCHR
7	6	5	4	3	2	1	0
RXCHR							

RXCHR: Received Character

Last character received if RXRDY is set.

RXSYNH: Received Sync

0: Last character received is a data.

1: Last character received is a command.

© 2017 Microchip Technology Inc.

39.7.18 USART Transmit Holding Register

Name:US_THR

Address:0xF801C01C (0), 0xF802001C (1), 0xF802401C (2), 0xF802801C (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
TXSYNH	–	–	–	–	–	–	TXCHR
7	6	5	4	3	2	1	0
TXCHR							

TXCHR: Character to be Transmitted

Next character to be transmitted after the current character if TXRDY is not set.

TXSYNH: Sync Field to be Transmitted

0: The next character sent is encoded as a data. Start frame delimiter is DATA SYNC.

1: The next character sent is encoded as a command. Start frame delimiter is COMMAND SYNC.

39.7.19 USART Baud Rate Generator Register

Name:US_BRGR

Address:0xF801C020 (0), 0xF8020020 (1), 0xF8024020 (2), 0xF8028020 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	FP		

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

CD: Clock Divider

	USART_MODE ≠ ISO7816			
	SYNC = 0		**SYNC = 1 or USART_MODE = SPI (Master or Slave)**	**USART_MODE = ISO7816**
CD	**OVER = 0**	**OVER = 1**		
0	Baud Rate Clock Disabled			
1 to 65535	CD = Selected Clock / (16 × Baud Rate)	CD = Selected Clock / (8 × Baud Rate)	CD = Selected Clock / Baud Rate	CD = Selected Clock / (FI_DI_RATIO × Baud Rate)

FP: Fractional Part

0: Fractional divider is disabled.

1–7: Baud rate resolution, defined by FP × 1/8.

Warning: When the value of field FP is greater than 0, the SCK (oversampling clock) generates non-constant duty cycles. The SCK high duration is increased by “selected clock” period from time to time. The duty cycle depends on the value of the CD field.

© 2017 Microchip Technology Inc.

39.7.20 USART Receiver Time-out Register

Name:US_RTOR

Address:0xF801C024 (0), 0xF8020024 (1), 0xF8024024 (2), 0xF8028024 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	TO
15	14	13	12	11	10	9	8
TO							
7	6	5	4	3	2	1	0
TO							

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

TO: Time-out Value

0: The receiver time-out is disabled.

1–131071: The receiver time-out is enabled and TO is Time-out Delay / Bit Period.

39.7.21 USART Transmitter Timeguard Register

Name:US_TTGR

Address:0xF801C028 (0), 0xF8020028 (1), 0xF8024028 (2), 0xF8028028 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
TG							

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

TG: Timeguard Value

0: The transmitter timeguard is disabled.

1–255: The transmitter timeguard is enabled and TG is Timeguard Delay / Bit Period.

© 2017 Microchip Technology Inc.

39.7.22 USART FI DI RATIO Register

Name:US_FIDI

Address:0xF801C040 (0), 0xF8020040 (1), 0xF8024040 (2), 0xF8028040 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	FI_DI_RATIO		

7	6	5	4	3	2	1	0
FI_DI_RATIO							

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

FI_DI_RATIO: FI Over DI Ratio Value

0: If ISO7816 mode is selected, the baud rate generator generates no signal.

1–2: Do not use.

3–2047: If ISO7816 mode is selected, the baud rate is the clock provided on SCK divided by FI_DI_RATIO.

39.7.23 USART Number of Errors Register

Name:US_NER

Address:0xF801C044 (0), 0xF8020044 (1), 0xF8024044 (2), 0xF8028044 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
NB_ERRORS							

This register is relevant only if USART_MODE = 0x4 or 0x6 in the USART Mode Register.

NB_ERRORS: Number of Errors

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

© 2017 Microchip Technology Inc.

39.7.24 USART IrDA Filter Register

Name:US_IF

Address:0xF801C04C (0), 0xF802004C (1), 0xF802404C (2), 0xF802804C (3)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
IRDA_FILTER							

This register is relevant only if USART_MODE = 0x8 in the USART Mode Register.

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

IRDA_FILTER: IrDA Filter

The IRDA_FILTER value must be defined to meet the following criteria:

$t_{peripheral\ clock} \times (IRDA\_FILTER + 3) < 1.41\ \mu s$

39.7.25 USART Manchester Configuration Register

Name:US_MAN

Address:0xF801C050 (0), 0xF8020050 (1), 0xF8024050 (2), 0xF8028050 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
–	DRIFT	ONE	RX_MPOL	–	–	RX_PP	

23	22	21	20	19	18	17	16
–	–	–	–	RX_PL			

15	14	13	12	11	10	9	8
–	–	–	TX_MPOL	–	–	TX_PP	

7	6	5	4	3	2	1	0
–	–	–	–	TX_PL			

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

TX_PL: Transmitter Preamble Length

0: The transmitter preamble pattern generation is disabled

1–15: The preamble length is TX_PL × Bit Period

TX_PP: Transmitter Preamble Pattern

The following values assume that TX_MPOL field is not set:

Value	Name	Description
0	ALL_ONE	The preamble is composed of ‘1’s
1	ALL_ZERO	The preamble is composed of ‘0’s
2	ZERO_ONE	The preamble is composed of ‘01’s
3	ONE_ZERO	The preamble is composed of ‘10’s

TX_MPOL: Transmitter Manchester Polarity

0: Logic zero is coded as a zero-to-one transition, Logic one is coded as a one-to-zero transition.

1: Logic zero is coded as a one-to-zero transition, Logic one is coded as a zero-to-one transition.

RX_PL: Receiver Preamble Length

0: The receiver preamble pattern detection is disabled

1–15: The detected preamble length is RX_PL × Bit Period

RX_PP: Receiver Preamble Pattern detected

The following values assume that RX_MPOL field is not set:

Value	Name	Description
00	ALL_ONE	The preamble is composed of ‘1’s
01	ALL_ZERO	The preamble is composed of ‘0’s
10	ZERO_ONE	The preamble is composed of ‘01’s
11	ONE_ZERO	The preamble is composed of ‘10’s

© 2017 Microchip Technology Inc.

RX_MPOL: Receiver Manchester Polarity

0: Logic zero is coded as a zero-to-one transition, Logic one is coded as a one-to-zero transition.

1: Logic zero is coded as a one-to-zero transition, Logic one is coded as a zero-to-one transition.

ONE: Must Be Set to 1

Bit 29 must always be set to 1 when programming the US_MAN register.

DRIFT: Drift Compensation

0: The USART cannot recover from an important clock drift

1: The USART can recover from clock drift. The 16X clock mode must be enabled.

© 2017 Microchip Technology Inc.

39.7.26 USART LIN Mode Register

Name:US_LINMR

Address:0xF801C054 (0), 0xF8020054 (1), 0xF8024054 (2), 0xF8028054 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	PDCM

15	14	13	12	11	10	9	8
DLC							

7	6	5	4	3	2	1	0
WKUPTYP	FSDIS	DLM	CHKTYP	CHKDIS	PARDIS	NACT	

This register is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

NACT: LIN Node Action

Value	Name	Description
00	PUBLISH	The USART transmits the response.
01	SUBSCRIBE	The USART receives the response.
10	IGNORE	The USART does not transmit and does not receive the response.

Values which are not listed in the table must be considered as “reserved”.

PARDIS: Parity Disable

0: In master node configuration, the identifier parity is computed and sent automatically. In master node and slave node configuration, the parity is checked automatically.

1: Whatever the node configuration is, the Identifier parity is not computed/sent and it is not checked.

CHKDIS: Checksum Disable

0: In master node configuration, the checksum is computed and sent automatically. In slave node configuration, the checksum is checked automatically.

1: Whatever the node configuration is, the checksum is not computed/sent and it is not checked.

CHKTYP: Checksum Type

0: LIN 2.0 “enhanced” checksum

1: LIN 1.3 “classic” checksum

DLM: Data Length Mode

0: The response data length is defined by field DLC of this register.

1: The response data length is defined by bits 5 and 6 of the identifier (IDCHR in US_LINIR).

FSDIS: Frame Slot Mode Disable

0: The Frame slot mode is enabled.

1: The Frame slot mode is disabled.

© 2017 Microchip Technology Inc.

WKUPTYP: Wakeup Signal Type

0: Setting the bit LINWKUP in the control register sends a LIN 2.0 wakeup signal.

1: Setting the bit LINWKUP in the control register sends a LIN 1.3 wakeup signal.

DLC: Data Length Control

0–255: Defines the response data length if DLM = 0,in that case the response data length is equal to DLC+1 bytes.

PDCM: DMAC Mode

0: The LIN mode register US_LINMR is not written by the DMAC.

1: The LIN mode register US_LINMR (excepting that flag) is written by the DMAC.

39.7.27 USART LIN Identifier Register

Name:US_LINIR

Address:0xF801C058 (0), 0xF8020058 (1), 0xF8024058 (2), 0xF8028058 (3)

Access: Read/Write or Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
IDCHR							

This register is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

IDCHR: Identifier Character

If USART_MODE = 0xA (master node configuration):

IDCHR is Read/Write and its value is the identifier character to be transmitted.

If USART_MODE = 0xB (slave node configuration):

IDCHR is Read-only and its value is the last identifier character that has been received.

© 2017 Microchip Technology Inc.

39.7.28 USART LIN Baud Rate Register

Name:US_LINBRR

Address:0xF801C05C (0), 0xF802005C (1), 0xF802405C (2), 0xF802805C (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	LINFP		

15	14	13	12	11	10	9	8
LINCD							

7	6	5	4	3	2	1	0
LINCD							

This register is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

Returns the baud rate value after the synchronization process completion.

LINCD: Clock Divider after Synchronization

LINFP: Fractional Part after Synchronization

39.7.29 USART Write Protection Mode Register

Name:US_WPMR

Address:0xF801C0E4 (0), 0xF80200E4 (1), 0xF80240E4 (2), 0xF80280E4 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

WPEN: Write Protection Enable

0: Disables the write protection if WPKEY corresponds to 0x555341 (“USA” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x555341 (“USA” in ASCII).

See Section 39.6.10 ”Register Write Protection” for the list of registers that can be write-protected.

WPKEY: Write Protection Key

Value	Name	Description
0x555341	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

© 2017 Microchip Technology Inc.

39.7.30 USART Write Protection Status Register

Name:US_WPSR

Address:0xF801C0E8 (0), 0xF80200E8 (1), 0xF80240E8 (2), 0xF80280E8 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

WPVS: Write Protection Violation Status

0: No write protection violation has occurred since the last read of the US_WPSR.

1: A write protection violation has occurred since the last read of the US_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

WPVSRC: Write Protection Violation Source

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

40. Universal Asynchronous Receiver Transmitter (UART)

40.1 Description

The Universal Asynchronous Receiver Transmitter (UART) features a two-pin UART that can be used for communication and trace purposes and offers an ideal medium for in-situ programming solutions.

Moreover, the association with a DMA controller permits packet handling for these tasks with processor time reduced to a minimum.

40.2 Embedded Characteristics

- Two-pin UART
  - Independent Receiver and Transmitter with a Common Programmable Baud Rate Generator
  - Even, Odd, Mark or Space Parity Generation
  - Parity, Framing and Overrun Error Detection
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
  - Interrupt Generation
  - Support for Two DMA Channels with Connection to Receiver and Transmitter

40.3 Block Diagram

Figure 40-1: UART Block Diagram

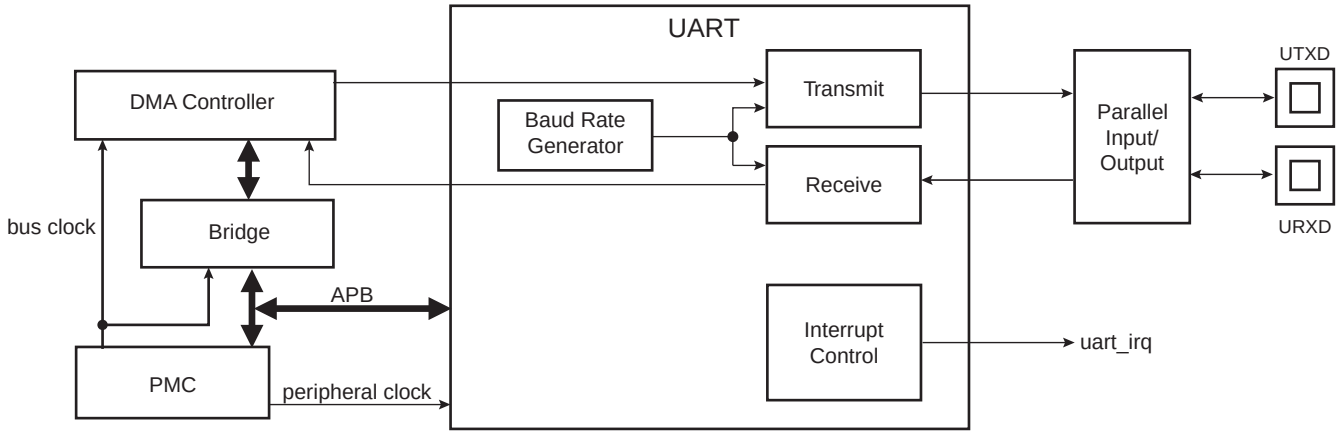


Table 40-1: UART Pin Description

Pin Name	Description	Type
URXD	UART Receive Data	Input
UTXD	UART Transmit Data	Output

© 2017 Microchip Technology Inc.

40.4 Product Dependencies

40.4.1 I/O Lines

The UART pins are multiplexed with PIO lines. The user must first configure the corresponding PIO Controller to enable I/O line operations of the UART.

Table 40-2: I/O Lines

Instance	Signal	I/O Line	Peripheral
UART0	URXD0	PC9	C
UART0	UTXD0	PC8	C
UART1	URXD1	PC17	C
UART1	UTXD1	PC16	C

40.4.2 Power Management

The UART clock can be controlled through the Power Management Controller (PMC). In this case, the user must first configure the PMC to enable the UART clock. Usually, the peripheral identifier used for this purpose is 1.

40.4.3 Interrupt Sources

The UART interrupt line is connected to one of the interrupt sources of the Interrupt Controller. Interrupt handling requires programming of the Interrupt Controller before configuring the UART.

Table 40-3: Peripheral IDs

Instance	ID
UART0	15
UART1	16

40.5 Functional Description

The UART operates in Asynchronous mode only and supports only 8-bit character handling (with parity). It has no clock pin.

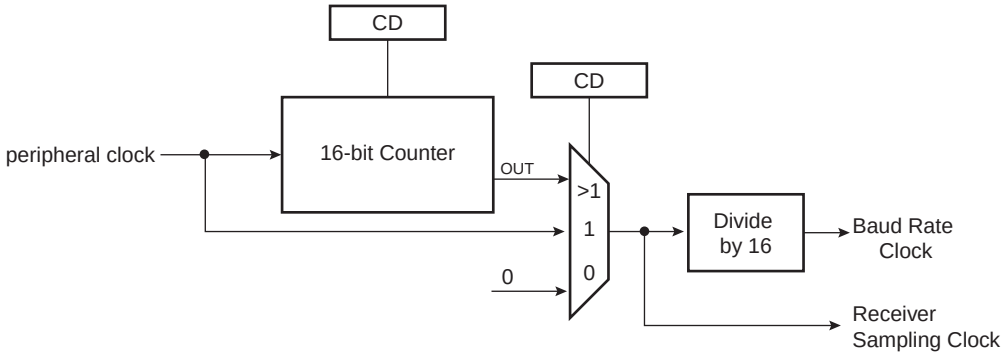
The UART is made up of a receiver and a transmitter that operate independently, and a common baud rate generator. Receiver timeout and transmitter time guard are not implemented. However, all the implemented features are compatible with those of a standard USART.

40.5.1 Baud Rate Generator

The baud rate generator provides the bit period clock named baud rate clock to both the receiver and the transmitter.

The baud rate clock is the peripheral clock divided by 16 times the clock divisor (CD) value written in the Baud Rate Generator register (UART_BRGR). If UART_BRGR is set to 0, the baud rate clock is disabled and the UART remains inactive. The maximum allowable baud rate is peripheral clock divided by 16. The minimum allowable baud rate is peripheral clock divided by (16 x 65536).

Figure 40-2: Baud Rate Generator



40.5.2 Receiver

40.5.2.1 Receiver Reset, Enable and Disable

After device reset, the UART receiver is disabled and must be enabled before being used. The receiver can be enabled by writing the Control Register (UART_CR) with the bit RXEN at 1. At this command, the receiver starts looking for a start bit.

The programmer can disable the receiver by writing UART_CR with the bit RXDIS at 1. If the receiver is waiting for a start bit, it is immediately stopped. However, if the receiver has already detected a start bit and is receiving the data, it waits for the stop bit before actually stopping its operation.

The receiver can be put in reset state by writing UART_CR with the bit RSTRX at 1. In this case, the receiver immediately stops its current operations and is disabled, whatever its current state. If RSTRX is applied when data is being processed, this data is lost.

40.5.2.2 Start Detection and Data Sampling

The UART only supports asynchronous operations, and this affects only its receiver. The UART receiver detects the start of a received character by sampling the URXD signal until it detects a valid start bit. A low level (space) on URXD is interpreted as a valid start bit if it is detected for more than seven cycles of the sampling clock, which is 16 times the baud rate. Hence, a space that is longer than 7/16 of the bit period is detected as a valid start bit. A space which is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

When a valid start bit has been detected, the receiver samples the URXD at the theoretical midpoint of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock (1-bit period) so the bit sampling point is eight cycles (0.5-bit period) after the start of the bit. The first sampling point is therefore 24 cycles (1.5-bit periods) after detecting the falling edge of the start bit.

Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

Figure 40-3: Start Bit Detection

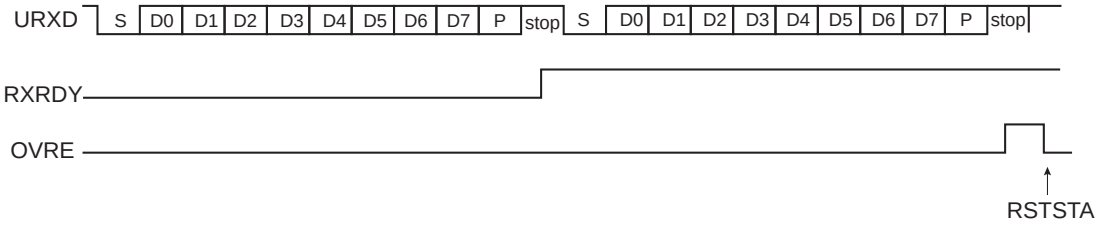
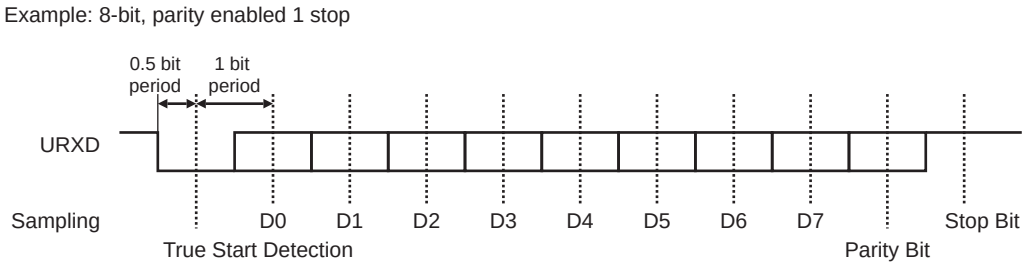


Figure 40-4: Character Reception

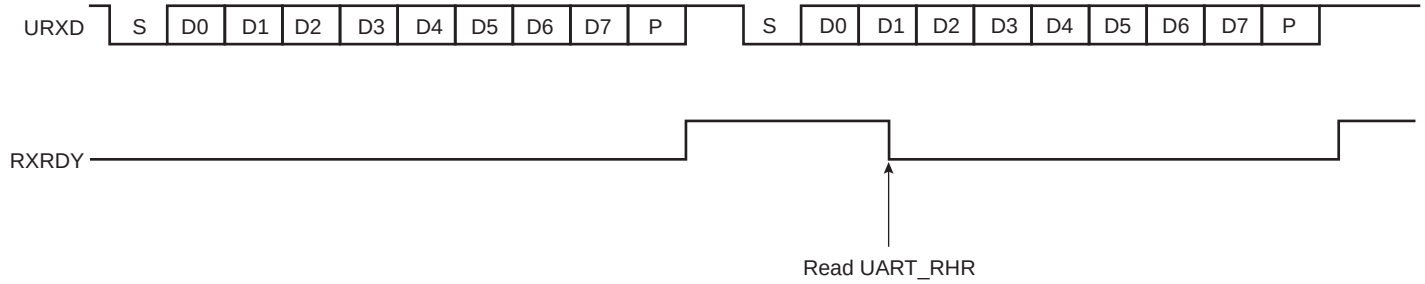


40.5.2.3 Receiver Ready

When a complete character is received, it is transferred to the Receive Holding Register (UART_RHR) and the RXRDY status bit in the Status Register (UART_SR) is set. The bit RXRDY is automatically cleared when UART_RHR is read.

© 2017 Microchip Technology Inc.

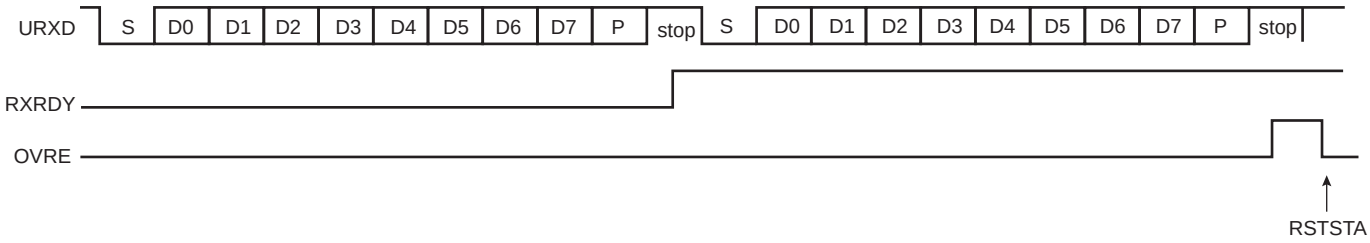
Figure 40-5: Receiver Ready



40.5.2.4 Receiver Overrun

The OVRE status bit in UART_SR is set if UART_RHR has not been read by the software (or the DMA Controller) since the last transfer, the RXRDY bit is still set and a new character is received. OVRE is cleared when the software writes a 1 to the bit RSTSTA (Reset Status) in UART_CR.

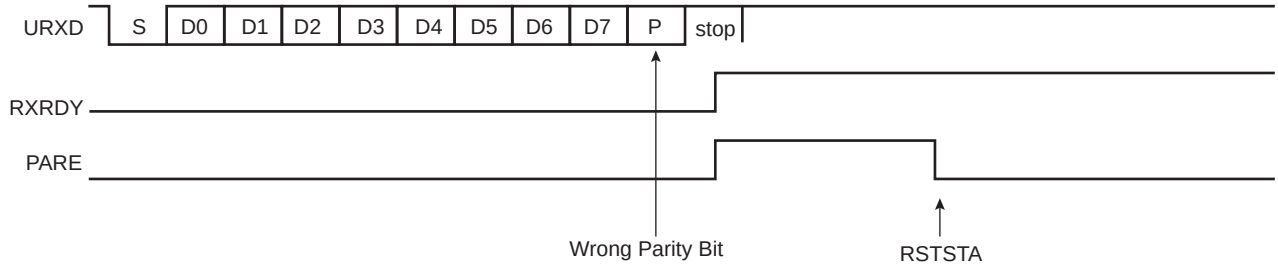
Figure 40-6: Receiver Overrun



40.5.2.5 Parity Error

Each time a character is received, the receiver calculates the parity of the received data bits, in accordance with the field PAR in the Mode Register (UART_MR). It then compares the result with the received parity bit. If different, the parity error bit PARE in UART_SR is set at the same time RXRDY is set. The parity bit is cleared when UART_CR is written with the bit RSTSTA (Reset Status) at 1. If a new character is received before the reset status command is written, the PARE bit remains at 1.

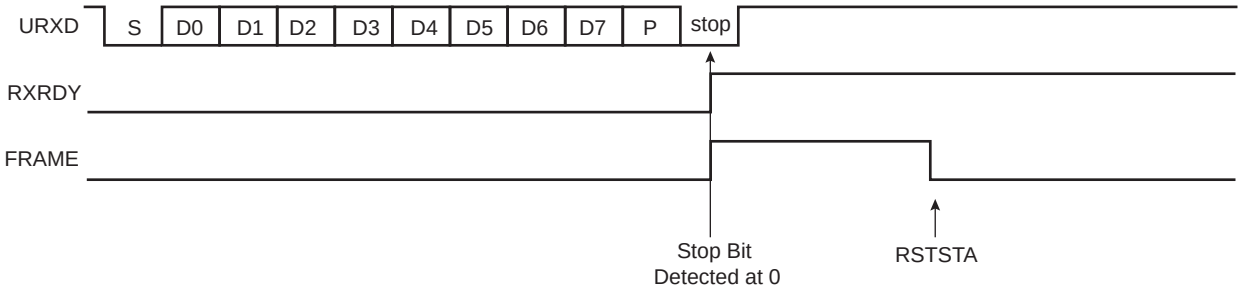
Figure 40-7: Parity Error



40.5.2.6 Receiver Framing Error

When a start bit is detected, it generates a character reception when all the data bits have been sampled. The stop bit is also sampled and when it is detected at 0, the FRAME (Framing Error) bit in UART_SR is set at the same time the RXRDY bit is set. The FRAME bit remains high until the Control Register (UART_CR) is written with the bit RSTSTA at 1.

Figure 40-8: Receiver Framing Error



40.5.3 Transmitter

40.5.3.1 Transmitter Reset, Enable and Disable

After device reset, the UART transmitter is disabled and must be enabled before being used. The transmitter is enabled by writing UART_CR with the bit TXEN at 1. From this command, the transmitter waits for a character to be written in the Transmit Holding Register (UART_THR) before actually starting the transmission.

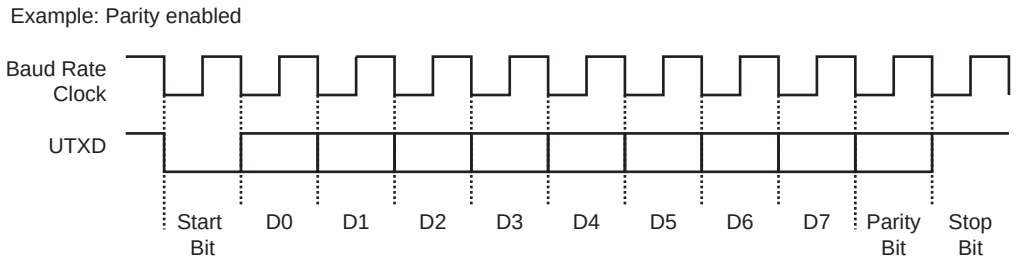
The programmer can disable the transmitter by writing UART_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the internal shift register and/or a character has been written in the UART_THR, the characters are completed before the transmitter is actually stopped.

The programmer can also put the transmitter in its reset state by writing the UART_CR with the bit RSTTX at 1. This immediately stops the transmitter, whether or not it is processing characters.

40.5.3.2 Transmit Format

The UART transmitter drives the pin UTXD at the baud rate clock speed. The line is driven depending on the format defined in UART_MR and the data stored in the internal shift register. One start bit at level 0, then the 8 data bits, from the lowest to the highest bit, one optional parity bit and one stop bit at 1 are consecutively shifted out as shown in the following figure. The field PARE in UART_MR defines whether or not a parity bit is shifted out. When a parity bit is enabled, it can be selected between an odd parity, an even parity, or a fixed space or mark bit.

Figure 40-9: Character Transmission



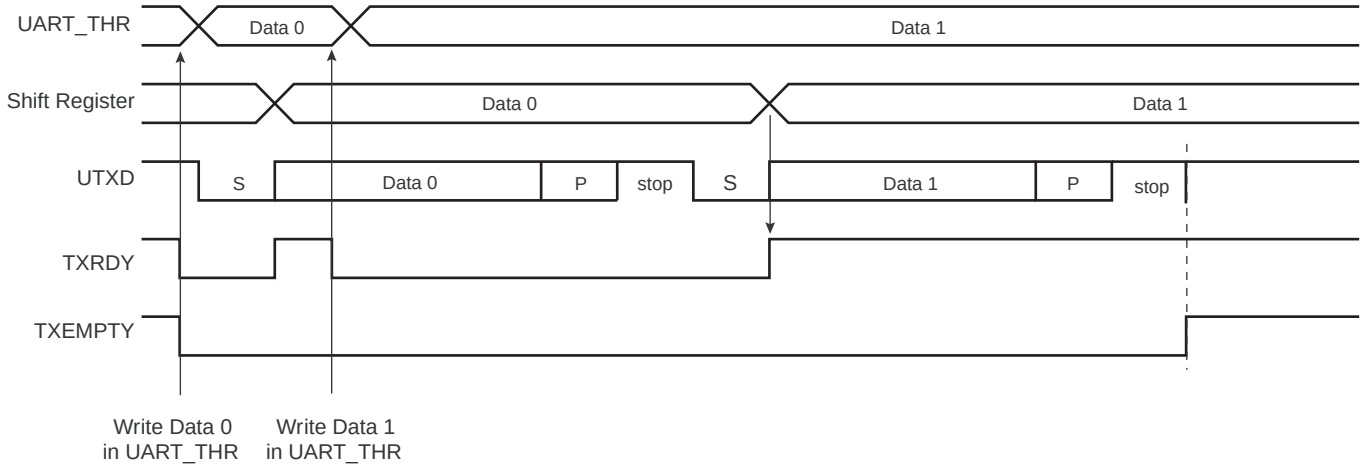
40.5.3.3 Transmitter Control

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in UART_SR. The transmission starts when the programmer writes in the UART_THR, and after the written character is transferred from UART_THR to the internal shift register. The TXRDY bit remains high until a second character is written in UART_THR. As soon as the first character is completed, the last character written in UART_THR is transferred into the internal shift register and TXRDY rises again, showing that the holding register is empty.

When both the internal shift register and UART_THR are empty, i.e., all the characters written in UART_THR have been processed, the TXEMPTY bit rises after the last stop bit has been completed.

© 2017 Microchip Technology Inc.

Figure 40-10: Transmitter Control



40.5.4 DMA Support

Both the receiver and the transmitter of the UART are connected to a DMA Controller (DMAC) channel.

The DMA Controller channels are programmed via registers that are mapped within the DMAC user interface.

40.5.5 Test Modes

The UART supports three test modes. These modes of operation are programmed by using the CHMODE field in UART_MR.

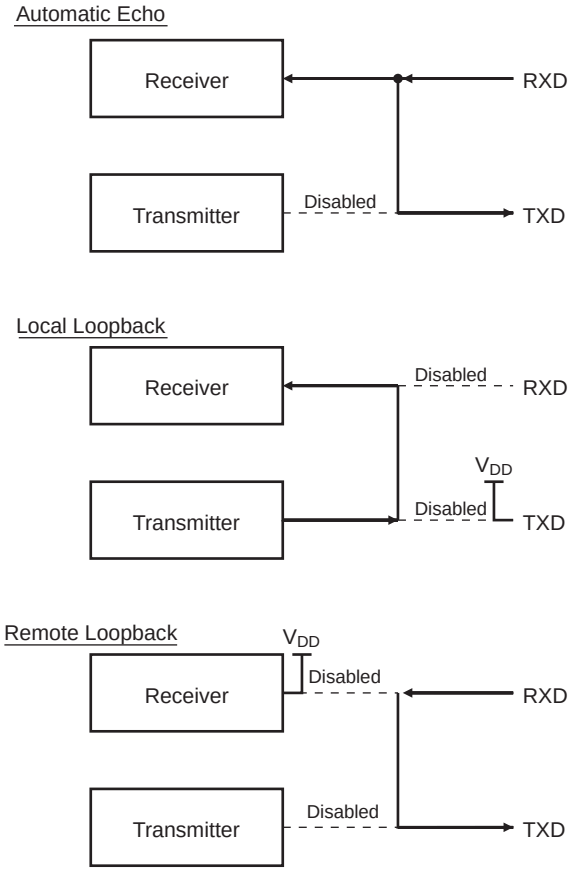
The Automatic echo mode allows a bit-by-bit retransmission. When a bit is received on the URXD line, it is sent to the UTXD line. The transmitter operates normally, but has no effect on the UTXD line.

The Local loopback mode allows the transmitted characters to be received. UTXD and URXD pins are not used and the output of the transmitter is internally connected to the input of the receiver. The URXD pin level has no effect and the UTXD line is held high, as in idle state.

The Remote loopback mode directly connects the URXD pin to the UTXD line. The transmitter and the receiver are disabled and have no effect. This mode allows a bit-by-bit retransmission.

© 2017 Microchip Technology Inc.

Figure 40-11: Test Modes



© 2017 Microchip Technology Inc.

40.6 Universal Asynchronous Receiver Transmitter (UART) User Interface

Table 40-4: Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Control Register	UART_CR	Write-only	–
0x0004	Mode Register	UART_MR	Read/Write	0x0
0x0008	Interrupt Enable Register	UART_IER	Write-only	–
0x000C	Interrupt Disable Register	UART_IDR	Write-only	–
0x0010	Interrupt Mask Register	UART_IMR	Read-only	0x0
0x0014	Status Register	UART_SR	Read-only	–
0x0018	Receive Holding Register	UART_RHR	Read-only	0x0
0x001C	Transmit Holding Register	UART_THR	Write-only	–
0x0020	Baud Rate Generator Register	UART_BRGR	Read/Write	0x0
0x0024	Reserved	–	–	–
0x0028–0x003C	Reserved	–	–	–
0x0040–0x00E8	Reserved	–	–	–
0x00EC–0x00FC	Reserved	–	–	–

40.6.1 UART Control Register

Name:UART_CR

Address:0xF8040000 (0), 0xF8044000 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RSTSTA
7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

RSTRX: Reset Receiver

0: No effect.

1: The receiver logic is reset and disabled. If a character is being received, the reception is aborted.

RSTTX: Reset Transmitter

0: No effect.

1: The transmitter logic is reset and disabled. If a character is being transmitted, the transmission is aborted.

RXEN: Receiver Enable

0: No effect.

1: The receiver is enabled if RXDIS is 0.

RXDIS: Receiver Disable

0: No effect.

1: The receiver is disabled. If a character is being processed and RSTRX is not set, the character is completed before the receiver is stopped.

TXEN: Transmitter Enable

0: No effect.

1: The transmitter is enabled if TXDIS is 0.

TXDIS: Transmitter Disable

0: No effect.

1: The transmitter is disabled. If a character is being processed and a character has been written in the UART_THR and RSTTX is not set, both characters are completed before the transmitter is stopped.

RSTSTA: Reset Status

0: No effect.

1: Resets the status bits PARE, FRAME and OVRE in the UART_SR.

© 2017 Microchip Technology Inc.

40.6.2 UART Mode Register

Name:UART_MR

Address:0xF8040004 (0), 0xF8044004 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CHMODE		–	–	PAR			–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

PAR: Parity Type

Value	Name	Description
0	EVEN	Even Parity
1	ODD	Odd Parity
2	SPACE	Space: parity forced to 0
3	MARK	Mark: parity forced to 1
4	NO	No parity

CHMODE: Channel Mode

Value	Name	Description
0	NORMAL	Normal mode
1	AUTOMATIC	Automatic echo
2	LOCAL_LOOPBACK	Local loopback
3	REMOTE_LOOPBACK	Remote loopback

40.6.3 UART Interrupt Enable Register

Name:UART_IER

Address:0xF8040008 (0), 0xF8044008 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

RXRDY: Enable RXRDY Interrupt

TXRDY: Enable TXRDY Interrupt

OVRE: Enable Overrun Error Interrupt

FRAME: Enable Framing Error Interrupt

PARE: Enable Parity Error Interrupt

TXEMPTY: Enable TXEMPTY Interrupt

© 2017 Microchip Technology Inc.

40.6.4 UART Interrupt Disable Register

Name:UART_IDR

Address:0xF804000C (0), 0xF804400C (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

RXRDY: Disable RXRDY Interrupt

TXRDY: Disable TXRDY Interrupt

OVRE: Disable Overrun Error Interrupt

FRAME: Disable Framing Error Interrupt

PARE: Disable Parity Error Interrupt

TXEMPTY: Disable TXEMPTY Interrupt

40.6.5 UART Interrupt Mask Register

Name:UART_IMR

Address:0xF8040010 (0), 0xF8044010 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

RXRDY: Mask RXRDY Interrupt

TXRDY: Disable TXRDY Interrupt

OVRE: Mask Overrun Error Interrupt

FRAME: Mask Framing Error Interrupt

PARE: Mask Parity Error Interrupt

TXEMPTY: Mask TXEMPTY Interrupt

© 2017 Microchip Technology Inc.

40.6.6 UART Status Register

Name:UART_SR

Address:0xF8040014 (0), 0xF8044014 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

RXRDY: Receiver Ready

0: No character has been received since the last read of the UART_RHR, or the receiver is disabled.

1: At least one complete character has been received, transferred to UART_RHR and not yet read.

TXRDY: Transmitter Ready

0: A character has been written to UART_THR and not yet transferred to the internal shift register, or the transmitter is disabled.

1: There is no character written to UART_THR not yet transferred to the internal shift register.

OVRE: Overrun Error

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

FRAME: Framing Error

0: No framing error has occurred since the last RSTSTA.

1: At least one framing error has occurred since the last RSTSTA.

PARE: Parity Error

0: No parity error has occurred since the last RSTSTA.

1: At least one parity error has occurred since the last RSTSTA.

TXEMPTY: Transmitter Empty

0: There are characters in UART_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1: There are no characters in UART_THR and there are no characters being processed by the transmitter.

© 2017 Microchip Technology Inc.

40.6.7 UART Receiver Holding Register

Name:UART_RHR

Address:0xF8040018 (0), 0xF8044018 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
RXCHR							

RXCHR: Received Character

Last received character if RXRDY is set.

© 2017 Microchip Technology Inc.

40.6.8 UART Transmit Holding Register

Name:UART_THR

Address:0xF804001C (0), 0xF804401C (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
TXCHR							

TXCHR: Character to be Transmitted

Next character to be transmitted after the current character if TXRDY is not set.

40.6.9 UART Baud Rate Generator Register

Name:UART_BRGR

Address:0xF8040020 (0), 0xF8044020 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CD							
7	6	5	4	3	2	1	0
CD							

CD: Clock Divisor

0: Baud rate clock is disabled

1 to 65,535:

$$CD = \frac{f_{peripheral\ clock}}{16 \times Baud\ Rate}$$

© 2017 Microchip Technology Inc.

41. Analog-to-Digital Converter (ADC)

41.1 Description

The ADC is based on a 10-bit Analog-to-Digital Converter (ADC) managed by an ADC Controller. Refer to Figure 41-1 “Analog-to-Digital Converter Block Diagram with Touchscreen Mode”. It also integrates a 12-to-1 analog multiplexer, making possible the analog-to-digital conversions of 12 analog lines. The conversions extend from 0V to the voltage carried on pin ADVREF.

The ADC digital controller embeds circuitry to reduce the resolution down to 8 bits. The 8-bit resolution mode prevents using 16-bit Peripheral DMA transfer into memory when only 8-bit resolution is required by the application. Note that using this low resolution mode does not increase the conversion rate.

Conversion results are reported in a common register for all channels, as well as in a channel-dedicated register.

Software trigger, external trigger on rising edge of the ADTRG pin or internal triggers from Timer Counter output(s) are configurable.

The comparison circuitry allows automatic detection of values below a threshold, higher than a threshold, in a given range or outside the range, thresholds and ranges being fully configurable.

The ADC also integrates a Sleep mode and a conversion sequencer and connects with a DMA channel. These features reduce both power consumption and processor intervention.

Finally, the user can configure ADC timings, such as startup time and tracking time.

This ADC Controller includes a Resistive Touchscreen Controller. It supports 4-wire and 5-wire technologies.

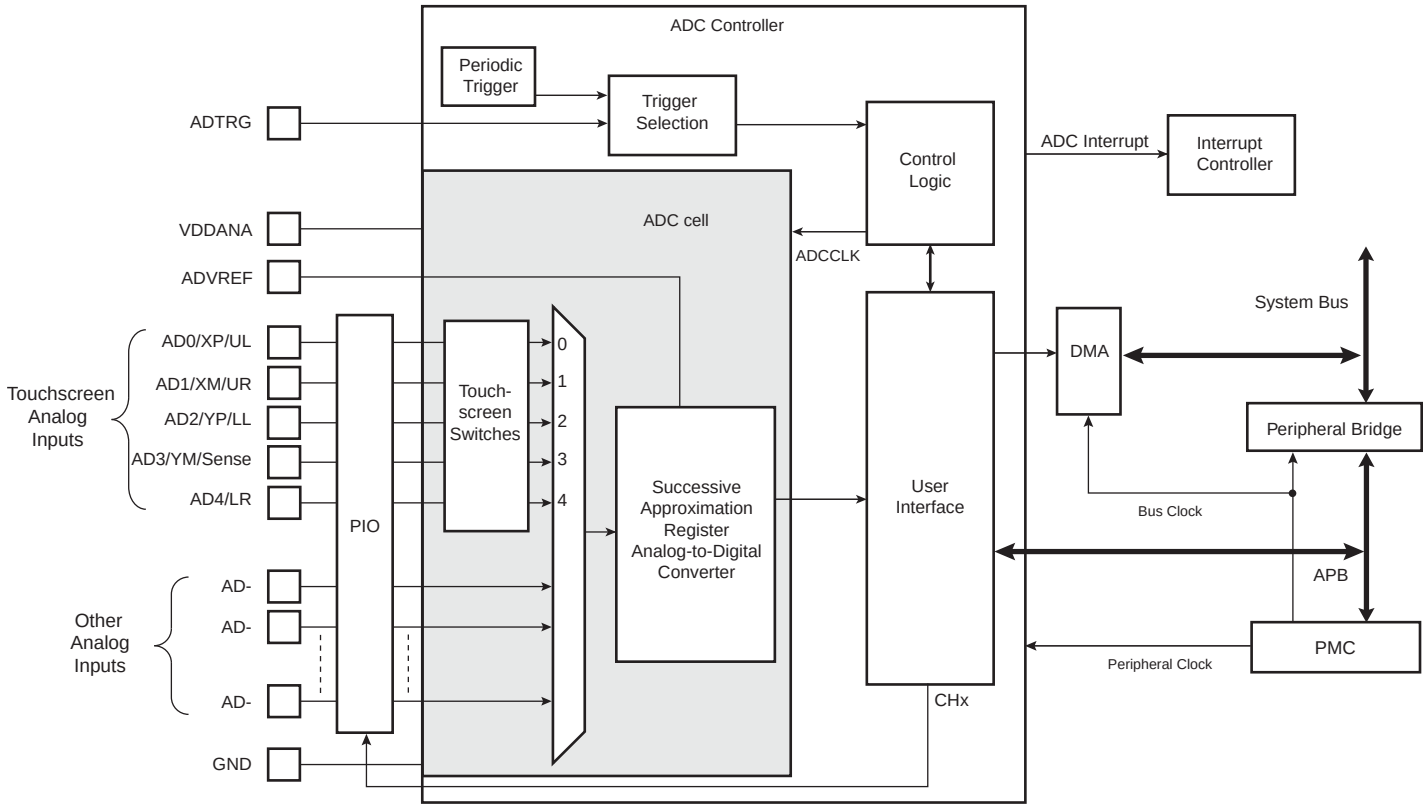
41.2 Embedded Characteristics

- 10-bit Resolution
- 300 sps Conversion Rate
- Wide Range of Power Supply Operation
- Resistive 4-wire and 5-wire Touchscreen Controller
  - Position and Pressure Measurement for 4-wire Screens
  - Position Measurement for 5-wire Screens
  - Average of Up to 8 Measures for Noise Filtering
- Programmable Pen Detection Sensitivity
- Integrated Multiplexer Offering Up to 12 Independent Analog Inputs
- Individual Enable and Disable of Each Channel
- Hardware or Software Trigger
  - External Trigger Pin
  - Internal Trigger Counter
  - Trigger on Pen Contact Detection
- DMA Support
- Possibility of ADC Timings Configuration
- Two Sleep Modes and Conversion Sequencer
  - Automatic Wakeup on Trigger and Back to Sleep Mode after Conversions of all Enabled Channels
  - Possibility of Customized Channel Sequence
- Standby Mode for Fast Wakeup Time Response
  - Power Down Capability
- Automatic Window Comparison of Converted Values
- Register Write Protection

© 2017 Microchip Technology Inc.

41.3 Block Diagram

Figure 41-1: Analog-to-Digital Converter Block Diagram with Touchscreen Mode



41.4 Signal Description

Table 41-1: ADC Pin Description

Pin Name	Description
VDDANA	Analog power supply
ADVREF	Reference voltage
AD0–AD11	Analog input channels
ADTRG	External trigger

© 2017 Microchip Technology Inc.

41.5 Product Dependencies

41.5.1 Power Management

The ADC Controller is not continuously clocked. The programmer must first enable the ADC Controller peripheral clock in the Power Management Controller (PMC) before using the ADC Controller. However, if the application does not require ADC operations, the ADC Controller clock can be stopped when not needed and restarted when necessary. Configuring the ADC Controller does not require the ADC Controller clock to be enabled.

41.5.2 Interrupt Sources

The ADC interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the ADC interrupt requires the interrupt controller to be programmed first.

Table 41-2: Peripheral IDs

Instance	ID
ADC	19

41.5.3 I/O Lines

The digital input ADC_ADTRG is multiplexed with digital functions on the I/O line and the selection of ADC_ADTRG is made using the PIO controller.

The analog inputs ADC_ADx are multiplexed with digital functions on the I/O lines. ADC_ADx inputs are selected as inputs of the ADCC when writing a one in the corresponding CHx bit of ADC_CHER and the digital functions are not selected.

Table 41-3: I/O Lines

Instance	Signal	I/O Line	Peripheral
ADC	ADTRG	PB18	B
ADC	AD0	PB11	X1
ADC	AD1	PB12	X1
ADC	AD2	PB13	X1
ADC	AD3	PB14	X1
ADC	AD4	PB15	X1
ADC	AD5	PB16	X1
ADC	AD6	PB17	X1
ADC	AD7	PB6	X1
ADC	AD8	PB7	X1
ADC	AD9	PB8	X1
ADC	AD10	PB9	X1
ADC	AD11	PB10	X1

41.5.4 Timer Triggers

Timer counters may or may not be used as hardware triggers depending on user requirements. Thus, some or all of the timer counters may be unconnected.

41.5.5 Conversion Performances

For performance and electrical characteristics of the ADC, see the section ‘Electrical Characteristics’.

41.6 Functional Description

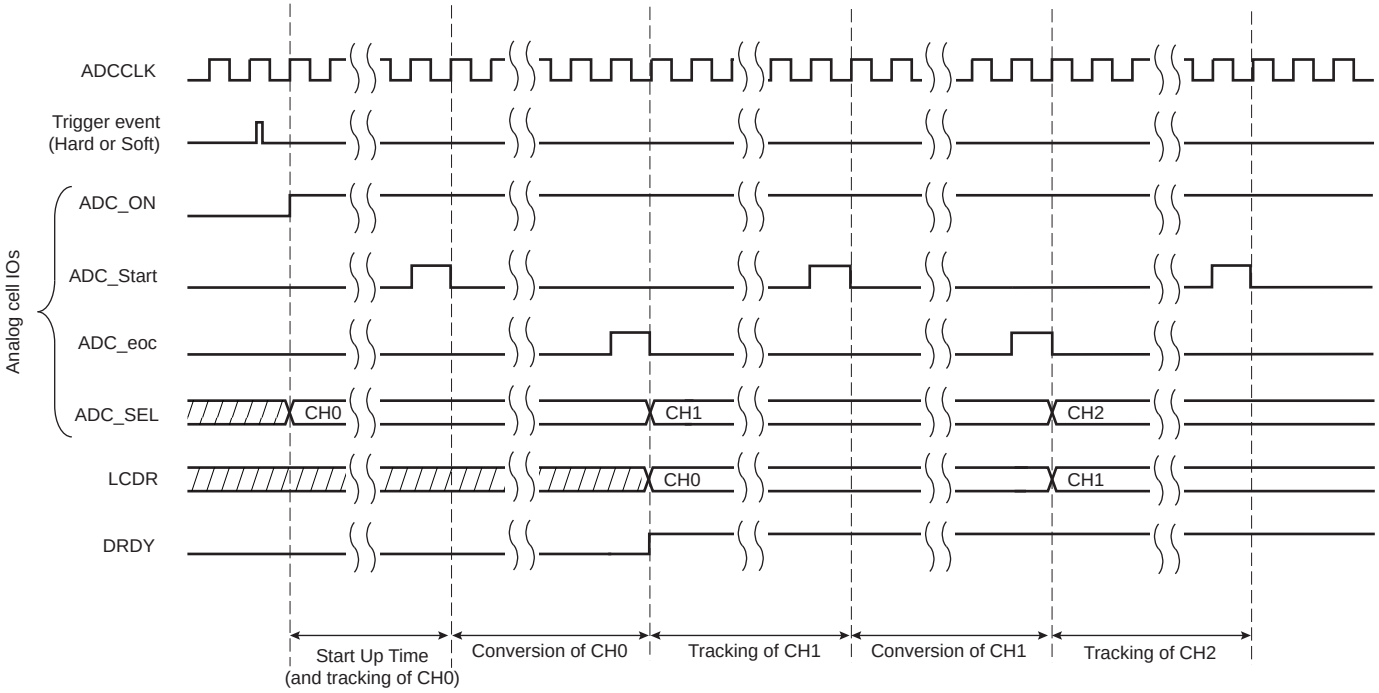
41.6.1 Analog-to-Digital Conversion

ADC conversions are sequenced by two operating times: the tracking time and the conversion time.

- The tracking time represents the time between the channel selection change and the time for the controller to start the ADC. The tracking time is set using the TRACKTIM field of the Mode Register (ADC_MR).
- The conversion time represents the time for the ADC to convert the analog signal.

In order to guarantee a conversion with minimum error, after any start of conversion, the ADC controller waits a number of ADC clock cycles (called hold time) before changing the channel selection again (and so starts a new tracking operation).

Figure 41-2: Sequence of ADC Conversions



41.6.2 ADC Clock

The ADC uses the ADC clock (ADCCLK) to perform conversions. The ADC clock frequency is selected in the PRESCAL field of ADC_MR.

The ADC clock frequency is between $f_{peripheral\ clock}/2$, if PRESCAL is 0, and $f_{peripheral\ clock}/512$, if PRESCAL is set to 255 (0xFF).

PRESCAL must be programmed to provide the ADC clock frequency parameter given in the section ‘Electrical Characteristics’.

41.6.3 ADC Reference Voltage

The conversion is performed on a full range between 0V and the reference voltage pin ADVREF. Analog inputs between these voltages convert to values based on a linear conversion.

41.6.4 Conversion Resolution

The ADC analog cell features a 10-bit resolution.

The ADC digital controller embeds circuitry to reduce the resolution down to 8 bits.

The 8-bit selection is performed by setting the LOWRES bit in ADC_MR. By default, after a reset, the resolution is the highest and the DATA field in the data registers is fully used. By setting the LOWRES bit, the ADC switches to the lowest resolution and the conversion results can be read in the lowest significant bits of the data registers. The two highest bits of the DATA field in the corresponding Channel Data register (ADC_CDR) and of the LDATA field in the Last Converted Data register (ADC_LCDR) read 0.

© 2017 Microchip Technology Inc.

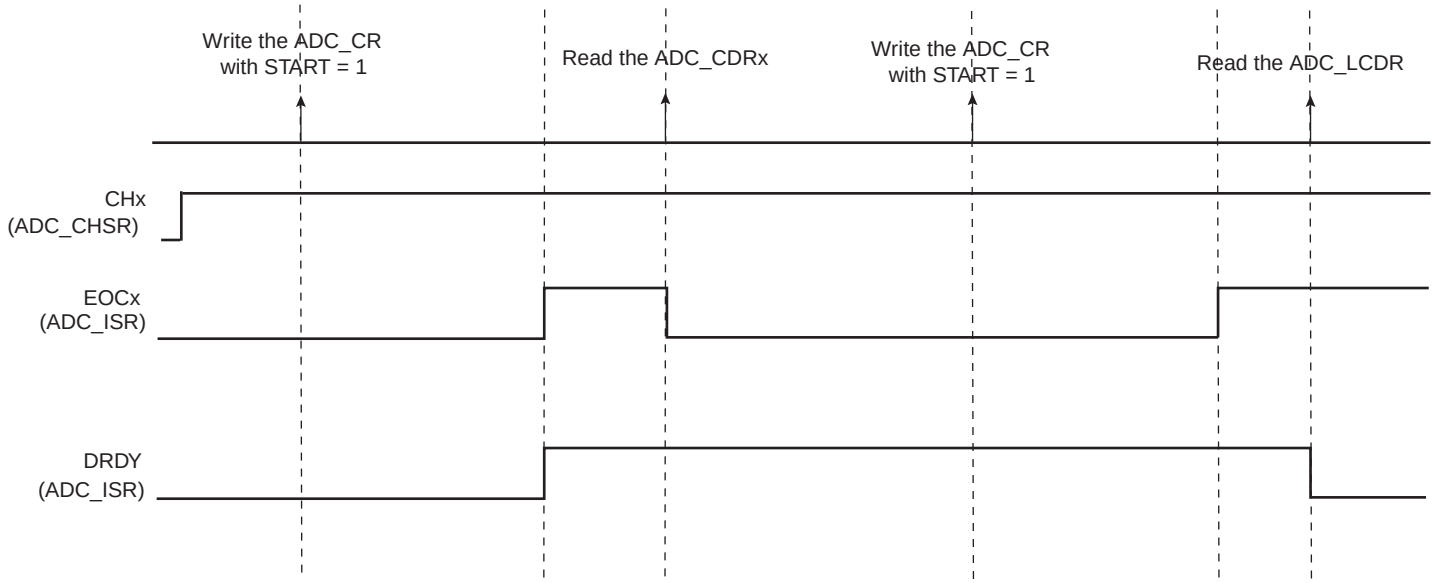
41.6.5 Conversion Results

When a conversion is completed, the resulting digital value is stored in the Channel Data register (ADC_CDRx) of the current channel and in the ADC Last Converted Data register (ADC_LCDR). By setting the TAG option in the Extended Mode Register (ADC_EMR), ADC_LCDR presents the channel number associated with the last converted data in the CHNB field.

The channel EOC bit and the DRDY bit in the Interrupt Status register (ADC_ISR) are set. In the case of a connected DMA channel, DRDY rising triggers a data request. In any case, either EOC and DRDY can trigger an interrupt.

Reading one of the ADC_CDRx clears the corresponding EOC bit. Reading ADC_LCDR clears the DRDY bit.

Figure 41-3: EOCx and DRDY Flag Behavior

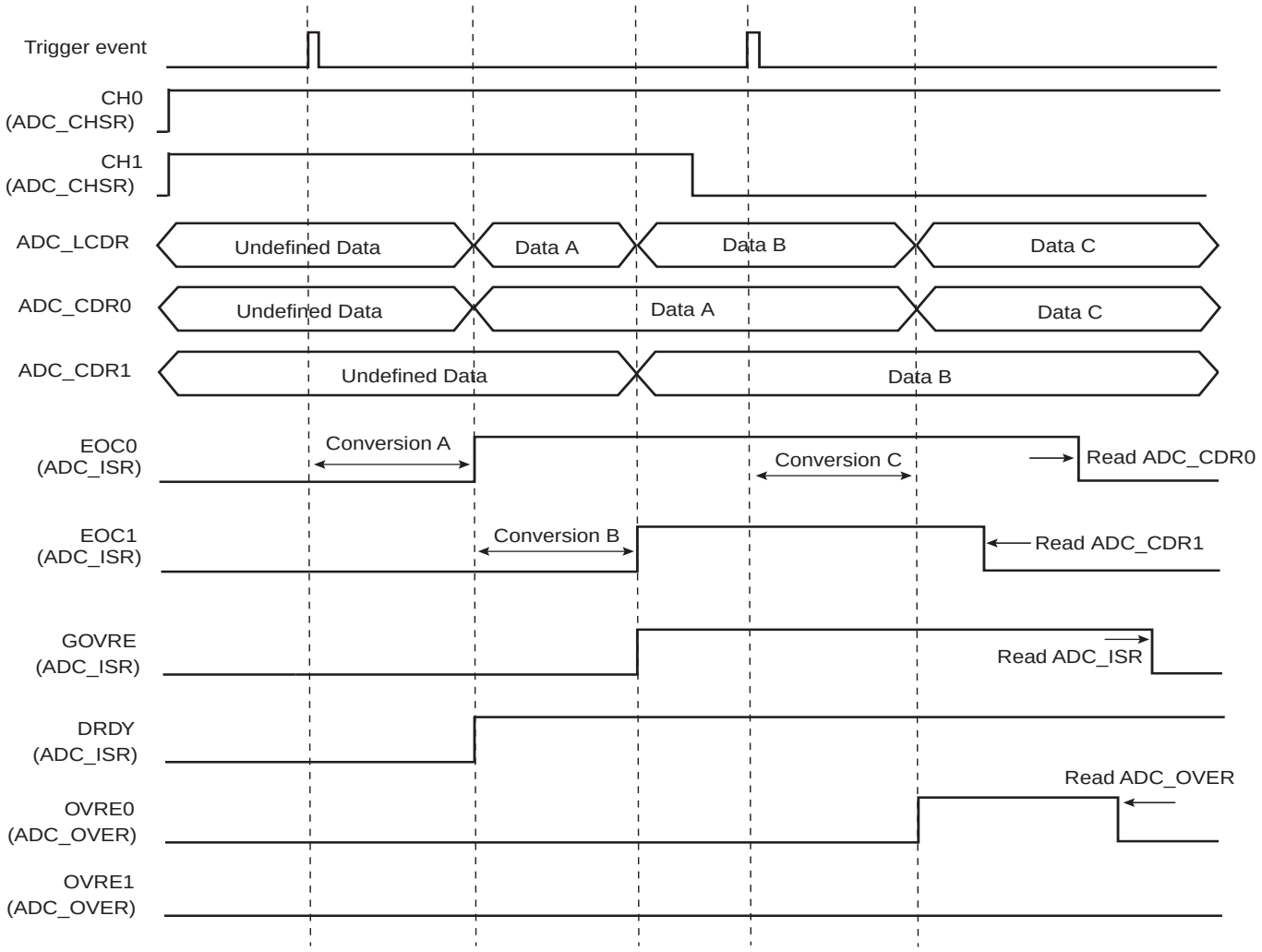


If ADC_CDR is not read before further incoming data is converted, the corresponding OVREx flag is set in the Overrun Status register (ADC_OVER).

New data converted when DRDY is high sets the GOVRE bit in ADC_ISR.

The OVREx flag is automatically cleared when ADC_OVER is read, and the GOVRE flag is automatically cleared when ADC_ISR is read.

Figure 41-4: EOCx, OVREx and GOVREx Flag Behavior



Warning: If the corresponding channel is disabled during a conversion or if it is disabled and then reenabled during a conversion, its associated data and corresponding EOCx and GOVRE flags in ADC_ISR and OVREx flags in ADC_OVER are unpredictable.

41.6.6 Conversion Triggers

Conversions of the active analog channels are started with a software or hardware trigger. The software trigger is provided by writing the Control register (ADC_CR) with the START bit at 1.

The hardware trigger can be one of the TIOA outputs of the Timer Counter channels or the external trigger input of the ADC (ADTRG).

The TRGMOD field in the ADC Trigger Register (ADC_TRGR) selects the hardware trigger from the following:

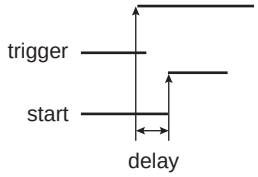
- any edge, either rising or falling or both, detected on the external trigger pin ADTRG
- the Pen Detect, depending on how the PENDET bit is set in the ADC Touchscreen Mode Register (ADC_TSMR)
- a continuous trigger, meaning the ADC Controller restarts the next sequence as soon as it finishes the current one
- a periodic trigger, which is defined by programming the TRGPER field in ADC_TRGR

The minimum time between two consecutive trigger events must be strictly greater than the duration time of the longest conversion sequence according to configuration of registers ADC_MR, ADC_CHSR, ADC_SEQRx, ADC_TSMR.

If a hardware trigger is selected, the start of a conversion is triggered after a delay starting at each rising edge of the selected signal. Due to asynchronous handling, the delay may vary in a range of two peripheral clock periods to one ADC clock period.

© 2017 Microchip Technology Inc.

Figure 41-5: Hardware Trigger Delay



Only one start command is necessary to initiate a conversion sequence on all the channels. The ADC hardware logic automatically performs the conversions on the active channels, then waits for a new request. The Channel Enable (ADC_CHER) and Channel Disable (ADC_CHDR) registers enable the analog channels to be enabled or disabled independently.

If the ADC is used with a DMA, only the transfers of converted data from enabled channels are performed and the resulting data buffers should be interpreted accordingly.

41.6.7 Sleep Mode and Conversion Sequencer

The ADC Sleep mode maximizes power saving by automatically deactivating the ADC when it is not being used for conversions. Sleep mode is selected by setting the SLEEP bit in ADC_MR.

Sleep mode is managed by a conversion sequencer, which automatically processes the conversions of all channels at lowest power consumption.

This mode can be used when the minimum period of time between two successive trigger events is greater than the startup period of the ADC. See the section ‘ADC Characteristics’ in the ‘Electrical Characteristics’.

When a start conversion request occurs, the ADC is automatically activated. As the analog cell requires a startup time, the logic waits during this time and starts the conversion on the enabled channels. When all conversions are complete, the ADC is deactivated until the next trigger. Triggers occurring during the sequence are ignored.

The conversion sequencer allows automatic processing with minimum processor intervention and optimized power consumption. Conversion sequences can be performed periodically using the internal timer (ADC_TRGR). The periodic acquisition of several samples can be processed automatically without any intervention of the processor via the DMA.

The sequence can be customized by programming the Sequence Channel Register ADC_SEQR1 and setting the USEQ bit of the Mode Register (ADC_MR). The user can choose a specific order of channels and can program up to 12 conversions by sequence. The user is free to create a personal sequence by writing channel numbers in ADC_SEQR1. Not only can channel numbers be written in any sequence, channel numbers can be repeated several times. When the bit USEQ in ADC_MR is set, the fields USCHx in ADC_SEQR1 are used to define the sequence. Only enabled USCHx fields will be part of the sequence. Each USCHx field has a corresponding enable, CHx-1, in ADC_CHER.

If all ADC channels (i.e., 12) are used on an application board, there is no restriction of usage of the user sequence. However, if some ADC channels are not enabled for conversion but rather used as pure digital inputs, the respective indexes of these channels cannot be used in the user sequence fields (see ADC_SEQRx). For example, if channel 4 is disabled (ADC_CSR[4] = 0), ADC_SEQRx fields USCH1 up to USCH12 must not contain the value 4. Thus the length of the user sequence may be limited by this behavior.

As an example, if only four channels over 12 (CH0 up to CH3) are selected for ADC conversions, the user sequence length cannot exceed four channels. Each trigger event may launch up to four successive conversions of any combination of channels 0 up to 3 but no more (i.e., in this case the sequence CH0, CH0, CH1, CH1, CH1 is impossible).

A sequence that repeats the same channel several times requires more enabled channels than channels actually used for conversion. For example, the sequence CH0, CH0, CH1, CH1 requires four enabled channels (four free channels on application boards) whereas only CH0, CH1 are really converted.

Note: The reference voltage pins always remain connected in Normal mode as in Sleep mode.

41.6.8 Comparison Window

The ADC Controller features automatic comparison functions. It compares converted values to a low threshold, a high threshold or both, depending on the value of the CMPMODE bit in ADC_EMR. The comparison can be done on all channels or only on the channel specified in the CMPSEL field of ADC_EMR. To compare all channels, the CMPALL bit of ADC_EMR must be set.

The flag can be read on the COMPE bit of the Interrupt Status register (ADC_ISR) and can trigger an interrupt.

The high threshold and the low threshold can be read/write in the Compare Window register (ADC_CWR).

If the comparison window is to be used with the LOWRES bit set in ADC_MR, the thresholds do not need to be adjusted, as the adjustment is done internally. However, whether the LOWRES bit is set or not, thresholds must always be configured in accordance with the maximum ADC resolution.

41.6.9 ADC Timings

Each ADC has its own minimal startup time that is programmed through the field STARTUP in ADC_MR.

A minimal tracking time is necessary for the ADC to guarantee the best converted final value between two channel selections. This time must be programmed in the TRACKTIM field in ADC_MR.

Warning: No input buffer amplifier to isolate the source is included in the ADC. This must be taken into consideration to program a precise value in the TRACKTIM field. See the section ‘ADC Characteristics’ in ‘Electrical Characteristics’.

41.6.10 Touchscreen

41.6.10.1 Touchscreen Mode

The TSMODE parameter of the ADC Touchscreen Mode Register (ADC_TSMR) is used to enable/disable the touchscreen functionality, to select the type of screen (4-wire or 5-wire) and, in the case of a 4-wire screen and to activate (or not) the pressure measurement.

In 4-wire mode, channel 0, 1, 2 and 3 must not be used for classic ADC conversions. Likewise, in 5-wire mode, channel 0, 1, 2, 3, and 4 must not be used for classic ADC conversions.

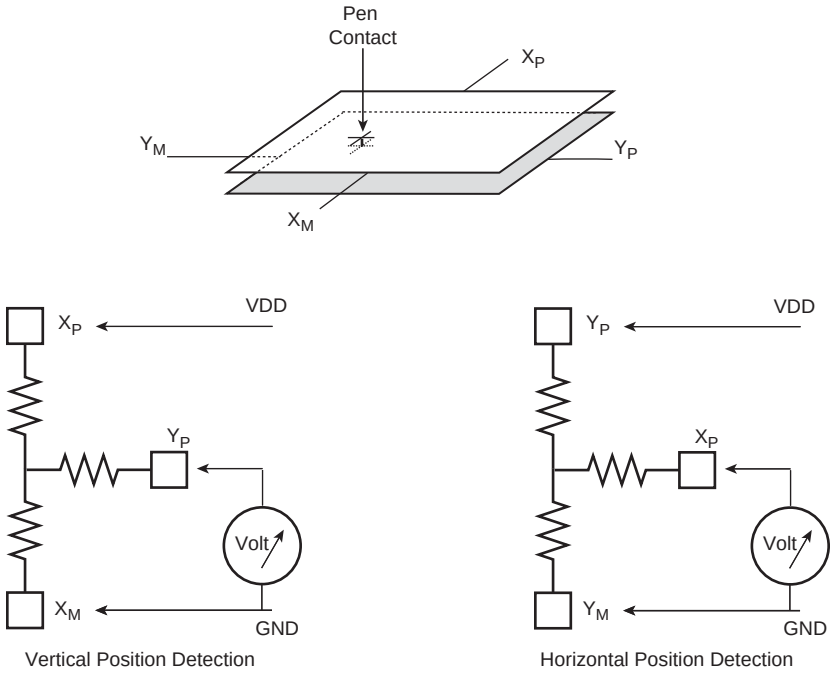
41.6.10.2 4-wire Resistive Touchscreen Principles

A resistive touchscreen is based on two resistive films, each one being fitted with a pair of electrodes, placed at the top and bottom on one film, and on the right and left on the other. In between, there is a layer acting as an insulator, but also enables contact when you press the screen. This is illustrated in Figure 41-6.

The ADC controller has the ability to perform without external components:

- position measurement
- pressure measurement
- pen detection

Figure 41-6: Touchscreen Position Measurement



© 2017 Microchip Technology Inc.

41.6.10.3 4-wire Position Measurement Method

As shown in Figure 41-6, to detect the position of a contact, a supply is first applied from top to bottom. Due to the linear resistance of the film, there is a voltage gradient from top to bottom. When a contact is performed on the screen, the voltage propagates at the point the two surfaces come into contact with the second film. If the input impedance on the right and left electrodes sense is high enough, the film does not affect this voltage, despite its resistive nature.

For the horizontal direction, the same method is used, but by applying supply from left to right. The range depends on the supply voltage and on the loss in the switches that connect to the top and bottom electrodes.

In an ideal world (linear, with no loss through switches), the horizontal position is equal to:

VY_M / VDD or VY_P / VDD.

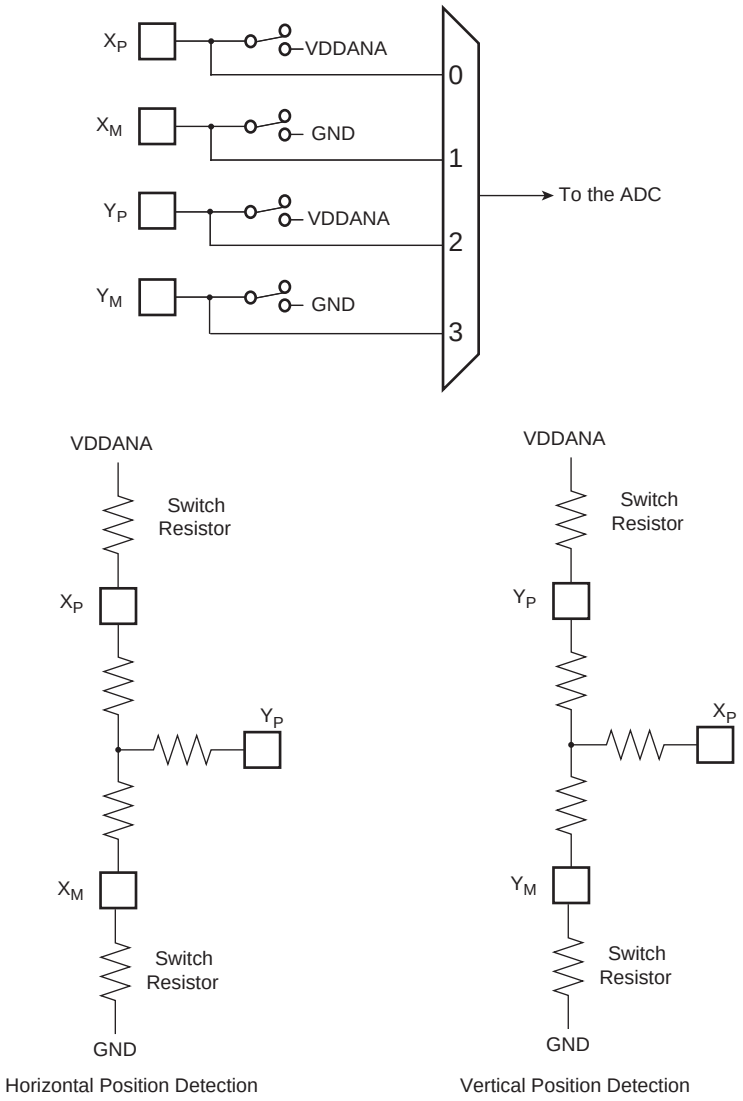
The implementation with on-chip power switches is shown in Figure 41-7. The voltage measurement at the output of the switch compensates for the switches loss.

It is possible to correct for switch loss by performing the operation:

$[VY_P - VX_M] / [VX_P - VX_M]$.

This requires additional measurements, as shown in Figure 41-7.

Figure 41-7: Touchscreen Switches Implementation



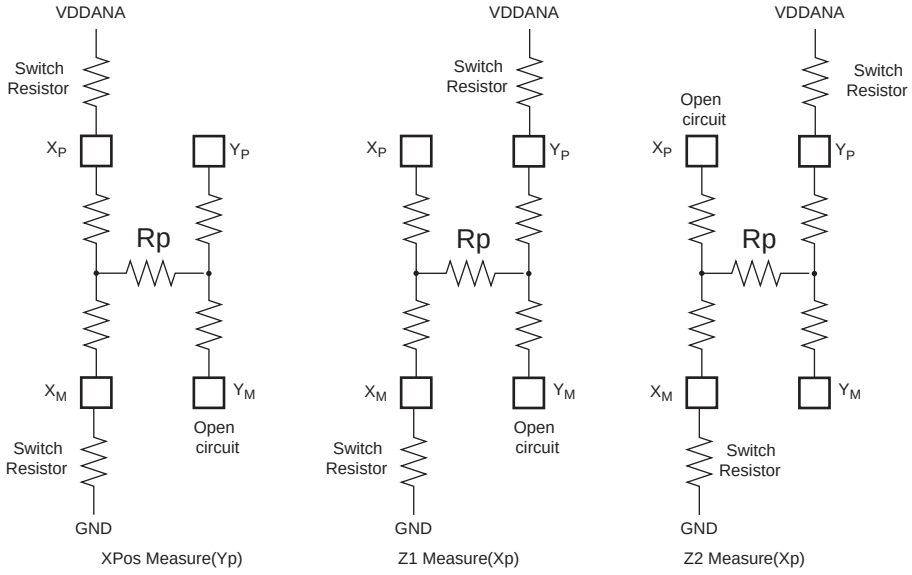
41.6.10.4 4-wire Pressure Measurement Method

The method to measure the pressure (Rp) applied to the touchscreen is based on the known resistance of the X-Panel resistance (Rxp).

Three conversions (Xpos,Z1,Z2) are necessary to determine the value of Rp (Zaxis resistance).

$Rp = Rxp \times (Xpos/1024) \times [(Z2/Z1)-1]$

Figure 41-8: Pressure Measurement



41.6.10.5 5-wire Resistive Touchscreen Principles

To make a 5-wire touchscreen, a resistive layer with a contact point at each corner and a conductive layer are used.

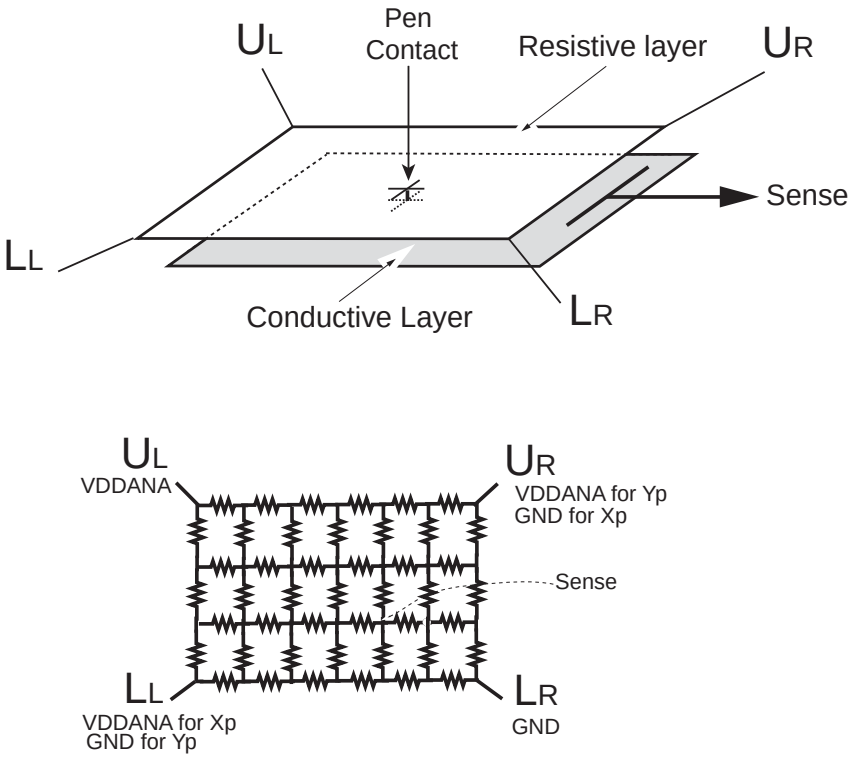
The 5-wire touchscreen differs from the 4-wire type mainly in that the voltage gradient is applied only to one layer, the resistive layer, while the other layer is the sense layer for both measurements.

The measurement of the X position is obtained by biasing the upper left corner and lower left corner to VDDANA and the upper right corner and lower right to ground.

To measure along the Y axis, bias the upper left corner and upper right corner to VDDANA and bias the lower left corner and lower right corner to ground.

© 2017 Microchip Technology Inc.

Figure 41-9: 5-Wire Principle



41.6.10.6 5-wire Position Measurement Method

In an application only monitoring clicks, 100 points per second is typically needed. For handwriting or motion detection, the number of measurements to consider is approximately 200 points per second. This must take into account that multiple measurements are included (over sampling, filtering) to compute the correct point.

The 5-wire touchscreen panel works by applying a voltage at the corners of the resistive layer and measuring the vertical or horizontal resistive network with the sense input. The ADC converts the voltage measured at the point the panel is touched.

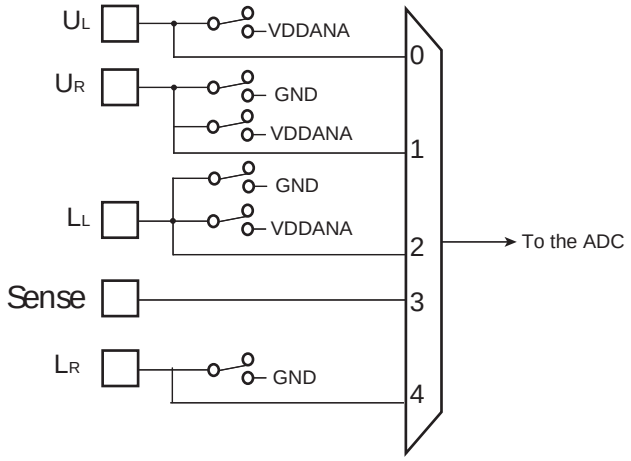
A measurement of the Y position of the pointing device is made by:

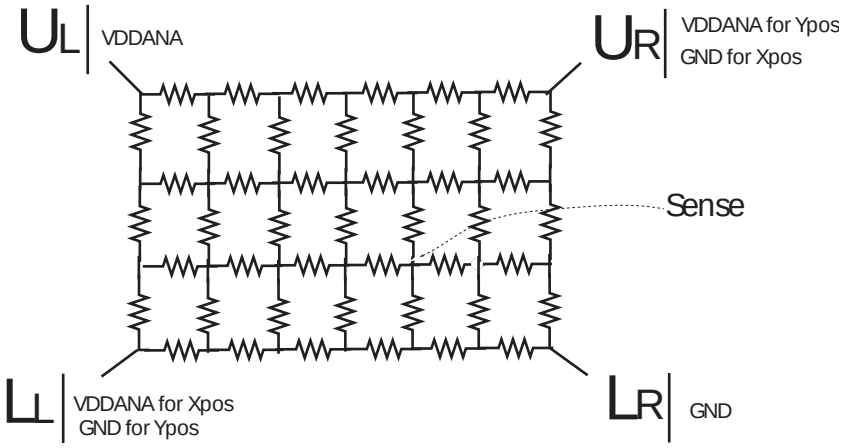
- Connecting Upper left (UL) and upper right (UR) corners to VDDANA
- Connecting Lower left (LL) and lower right (LR) corners to ground.
- The voltage measured is determined by the voltage divider developed at the point of touch (Yposition) and the SENSE input is converted by ADC.

A measurement of the X position of the pointing device is made by:

- Connecting the upper left (UL) and lower left (LL) corners to ground
- Connecting the upper right and lower right corners to VDDANA.
- The voltage measured is determined by the voltage divider developed at the point of touch (Xposition) and the SENSE input is converted by ADC.

Figure 41-10: Touchscreen Switches Implementation





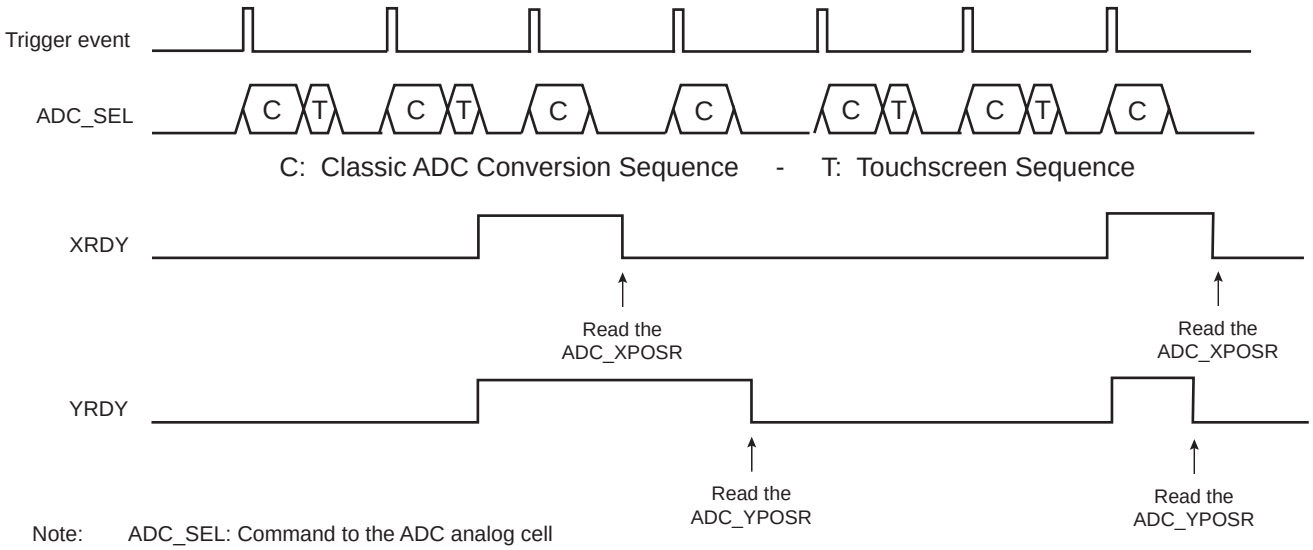
41.6.10.7 Sequence and Noise Filtering

The ADC Controller can manage ADC conversions and touchscreen measurement. On each trigger event the sequence of ADC conversions is performed as described in Section 41.6.7 “Sleep Mode and Conversion Sequencer”. The touchscreen measure frequency can be specified in number of trigger events by writing the TSFREQ parameter in ADC_TSMR. An internal counter counts triggers up to TSFREQ, and every time it rolls out, a touchscreen sequence is appended to the classic ADC conversion sequence (see Figure 41-11).

Additionally the user can average multiple touchscreen measures by writing the TSAV parameter in ADC_TSMR. This can be 1, 2, 4 or 8 measures performed on consecutive triggers as illustrated in Figure 41-11 below. Consequently, the TSFREQ parameter must be greater or equal to the TSAV parameter.

© 2017 Microchip Technology Inc.

Figure 41-11: Insertion of Touchscreen Sequences (TSFREQ = 2; TSAV = 1)



Note: ADC_SEL: Command to the ADC analog cell

41.6.10.8 Measured Values, Registers and Flags

As soon as the controller finishes the Touchscreen sequence, XRDY, YRDY and PRDY are set and can generate an interrupt. These flags can be read in the ADC Interrupt Status register (ADC_ISR). They are reset independently by reading in the ADC Touchscreen X Position register (ADC_XPOSR), the ADC Touchscreen Y Position register (ADC_YPOSR) and the ADC Touchscreen Pressure register (ADC_PRESSR).

ADC_XPOSR presents XPOS (V_X - V_{Xmin}) on its LSB and XSCALE (V_{XMAX} - V_{Xmin}) aligned on the 16th bit.

ADC_YPOSR presents YPOS (V_Y - V_{Ymin}) on its LSB and YSCALE (V_{YMAX} - V_{Ymin}) aligned on the 16th bit.

To improve the quality of the measure, the user must calculate XPOS/XSCALE and YPOS/YSCALE.

V_{XMAX}, V_{Xmin}, V_{YMAX}, and V_{Ymin} are measured at the first start up of the controller. These values can change during use, so it can be necessary to refresh them. Refresh can be done by writing ‘1’ in the TSCALIB field of the control register (ADC_CR).

ADC_PRESSR presents Z1 on its LSB and Z2 aligned on the 16th bit. See Section 41.6.10.4 “4-wire Pressure Measurement Method”.

41.6.10.9 Pen Detect Method

When there is no contact, it is not necessary to perform a conversion. However, it is important to detect a contact by keeping the power consumption as low as possible.

The implementation polarizes one panel by closing the switch on (X_P/U_L) and ties the horizontal panel by an embedded resistor connected to Y_M / Sense. This resistor is enabled by a fifth switch. Since there is no contact, no current is flowing and there is no related power consumption. As soon as a contact occurs, a current is flowing in the Touchscreen and a Schmitt trigger detects the voltage in the resistor.

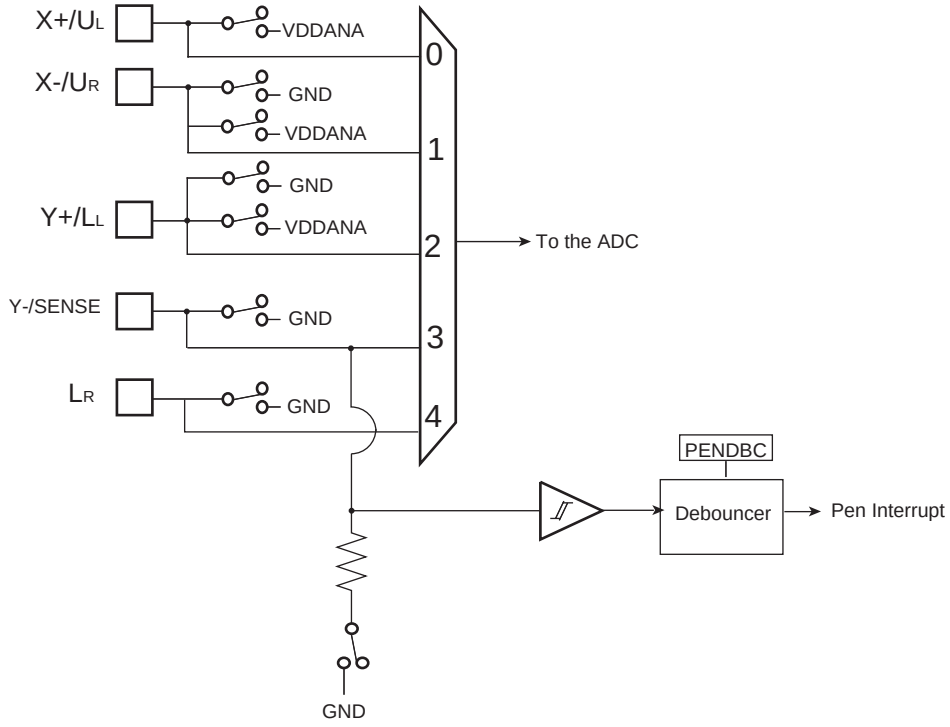
The Touchscreen Interrupt configuration is entered by programming the PENDET bit in ADC_TSMR. If this bit is written at 1, the controller samples the pen contact state when it is not converting and waiting for a trigger.

To complete the circuit, a programmable debouncer is placed at the output of the Schmitt trigger. This debouncer is programmable up to 2^{15} ADC clock periods. The debouncer length can be selected by programming the field PENDBC in ADC_TSMR.

Due to the analog switch’s structure, the debouncer circuitry is only active when no conversion (touchscreen or classic ADC channels) is in progress. Thus, if the time between the end of a conversion sequence and the arrival of the next trigger event is lower than the debouncing time configured on PENDBC, the debouncer will not detect any contact.

© 2017 Microchip Technology Inc.

Figure 41-12: Touchscreen Pen Detect



The touchscreen pen detect can be used to generate an ADC interrupt to wake up the system. The pen detect generates two types of status, reported in ADC_ISR:

- the PEN bit is set as soon as a contact exceeds the debouncing time as defined by PENDBC and remains set until ADC_ISR is read.
- the NOPEN bit is set as soon as no current flows for a time over the debouncing time as defined by PENDBC and remains set until ADC_ISR is read.

Both bits are automatically cleared as soon as ADC_ISR is read, and can generate an interrupt by writing ADC_IER.

Moreover, the rising of either one of them clears the other, they cannot be set at the same time.

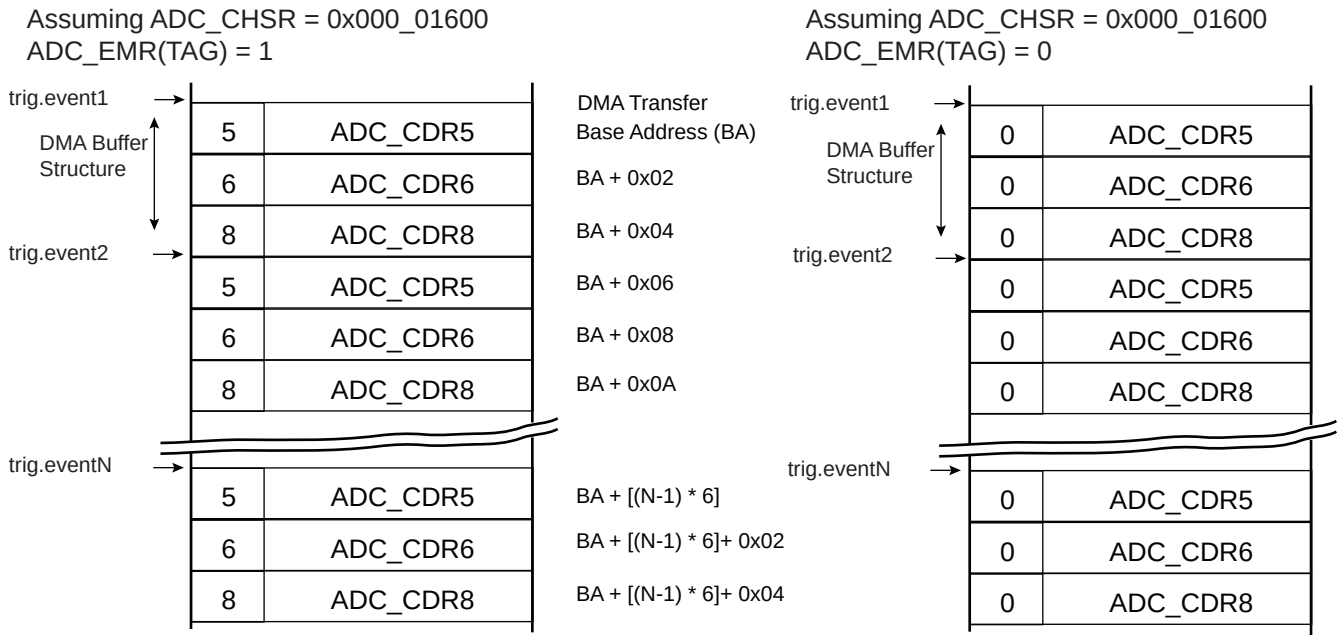
The PENS bit of ADC_ISR shows the current status of the pen contact.

41.6.11 Buffer Structure

The DMA read channel is triggered each time a new data is stored in ADC_LCDR. The same structure of data is repeatedly stored in ADC_LCDR each time a trigger event occurs. Depending on user mode of operation (ADC_MR, ADC_CHSR, ADC_SEQR1, ADC_TSMR) the structure differs. Each data read to DMA buffer, carried on a half-word (16-bit), consists of last converted data right aligned and when TAG is set in ADC_EMR, the four most significant bits are carrying the channel number thus allowing an easier post-processing in the DMA buffer or better checking the DMA buffer integrity.

© 2017 Microchip Technology Inc.

Figure 41-13: Buffer Structure



As soon as touchscreen conversions are required, the pen detection function may help the post-processing of the buffer. Refer to Section 41.6.11.4 “Pen Detection Status”.

41.6.11.1 Classical ADC Channels Only

When no touchscreen conversion is required (i.e., TSMODE = 0 in ADC_TSMR), the structure of data within the buffer is defined by ADC_MR, ADC_CHSR, ADC_SEQRx. See Figure 41-13.

If the user sequence is not used (i.e., USEQ is cleared in ADC_MR) then only the value of ADC_CHSR defines the data structure. For each trigger event, enabled channels will be consecutively stored in ADC_LCDR and automatically read to the buffer.

When the user sequence is configured (i.e., USEQ is set in ADC_MR) not only does ADC_CHSR modify the data structure of the buffer, but ADC_SEQRx registers may modify the data structure of the buffer as well.

41.6.11.2 Touchscreen Channels Only

When only touchscreen conversions are required (i.e., TSMODE ≠ 0 in ADC_TSMR and ADC_CHSR equals 0), the structure of data within the buffer is defined by ADC_TSMR.

When TSMODE = 1 or 3, each trigger event adds two half-words in the buffer (assuming TSAV = 0), first half-word being XPOS of ADC_XPOSR then YPOS of ADC_YPOSR. If TSAV/TSFREQ ≠ 0, the data structure remains unchanged. Not all trigger events add data to the buffer.

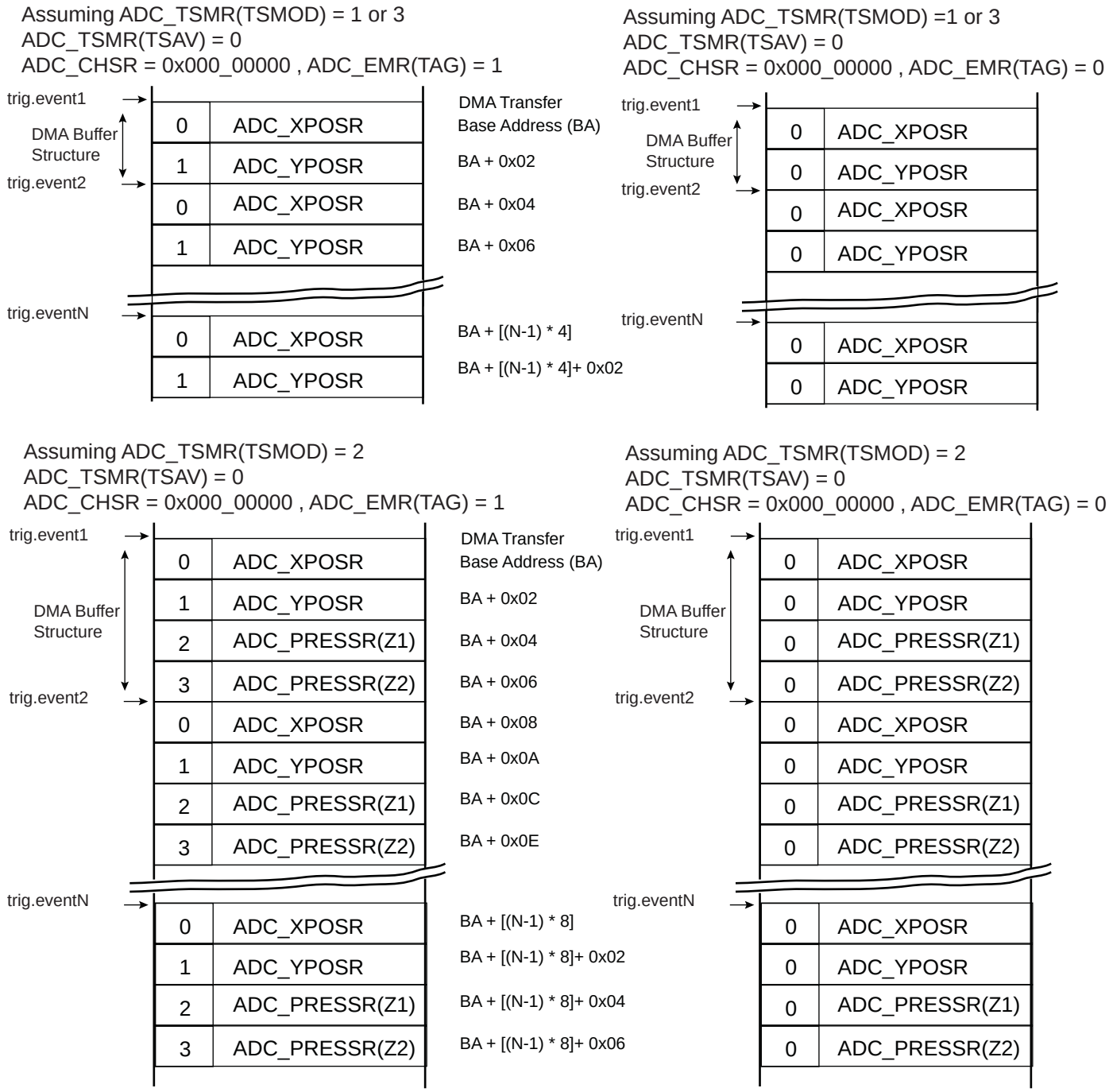
When TSMODE = 2, each trigger event adds four half-words to the buffer (assuming TSAV = 0), first half-word being XPOS of ADC_XPOSR followed by YPOS of ADC_YPOSR and finally Z1 followed by Z2, both located in ADC_PRESSR.

When TAG is set (ADC_EMR), the CHNB field (four most significant bits of ADC_LCDR) is cleared when XPOS is transmitted and set when YPOS is transmitted, allowing an easier post-processing of the buffer or a better checking of the buffer integrity. In case 4-wire with Pressure mode is selected, Z1 value is transmitted to the buffer along with tag set to 2 and Z2 is tagged with value 3.

XSCALE and YSCALE (calibration values) are not transmitted to the buffer because they are supposed to be constant and moreover only measured at the very first start up of the controller or upon user request.

There is no change in buffer structure whatever the value of PENDET bit configuration in ADC_TSMR but it is recommended to use the pen detection function for buffer post-processing (refer to Section 41.6.11.4 “Pen Detection Status”).

Figure 41-14: Buffer Structure When Only Touchscreen Channels are Enabled



41.6.11.3 Interleaved Channels

When both classic ADC channels (CH4/CH5 up to CH12 are set in ADC_CHSR) and touchscreen conversions are required (TSMODE ≠ 0 in ADC_TSMR) the structure of the buffer differs according to TSAV and TSFREQ values.

If TSFREQ ≠ 0, not all events generate touchscreen conversions, therefore the buffer structure is based on 2^{TSFREQ} trigger events. Given a TSFREQ value, the location of touchscreen conversion results depends on TSAV value.

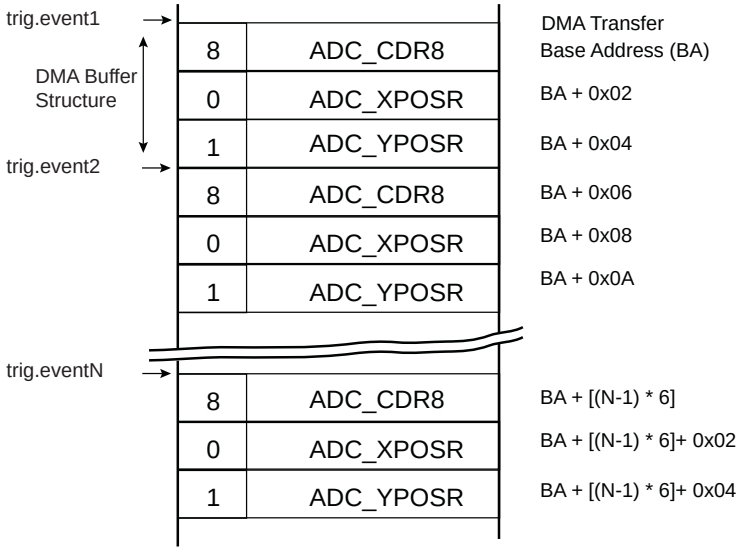
When TSFREQ = 0, TSAV must equal 0.

There is no change in buffer structure whatever the value of PENDET bit configuration in ADC_TSMR but it is recommended to use the pen detection function for buffer post-processing (refer to Section 41.6.11.4 “Pen Detection Status”).

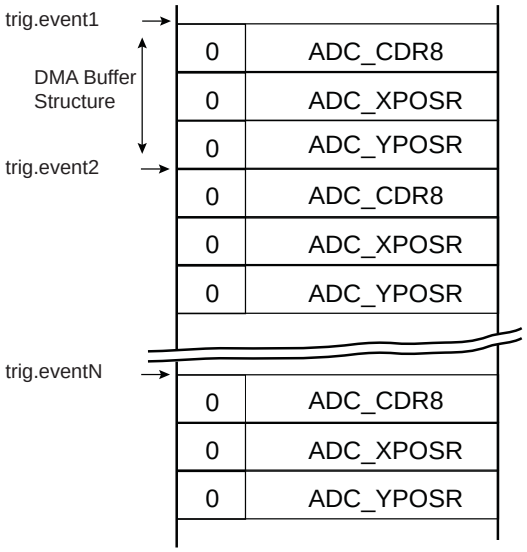
© 2017 Microchip Technology Inc.

Figure 41-15: Buffer Structure When Classic ADC and Touchscreen Channels are Interleaved

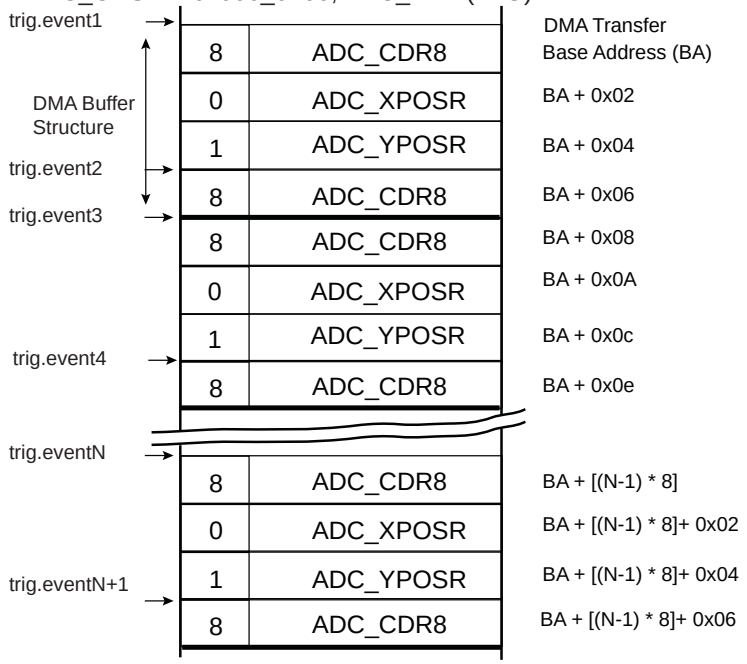
Assuming ADC_TSMR(TSMOD) = 1
ADC_TSMR(TSAV) = ADC_TSMR(TSFREQ) = 0
ADC_CHSR = 0x000_0100, ADC_EMR(TAG) =1



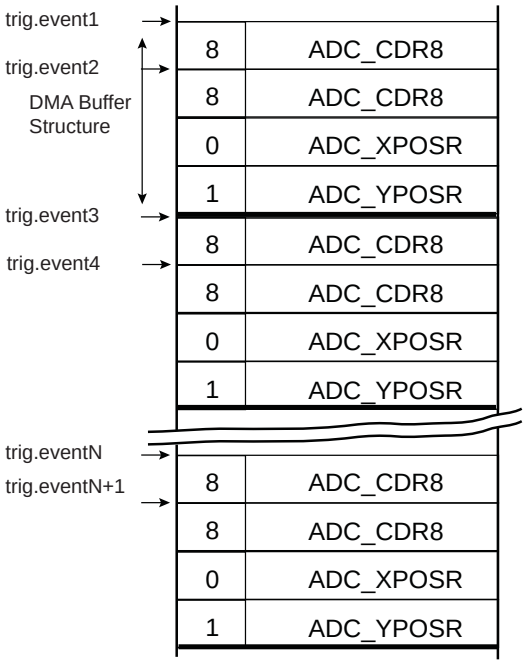
Assuming ADC_TSMR(TSMOD) = 1
ADC_TSMR(TSAV) = ADC_TSMR(TSFREQ) = 0
ADC_CHSR = 0x000_0100, ADC_EMR(TAG) = 0



Assuming ADC_TSMR(TSMOD) = 1
ADC_TSMR(TSAV) = 0 ADC_TSMR(TSFREQ) = 1
ADC_CHSR = 0x000_0100, ADC_EMR(TAG) = 1



Assuming ADC_TSMR(TSMOD) = 1
ADC_TSMR(TSAV) = 1 ADC_TSMR(TSFREQ) = 1
ADC_CHSR = 0x000_0100, ADC_EMR(TAG) = 1



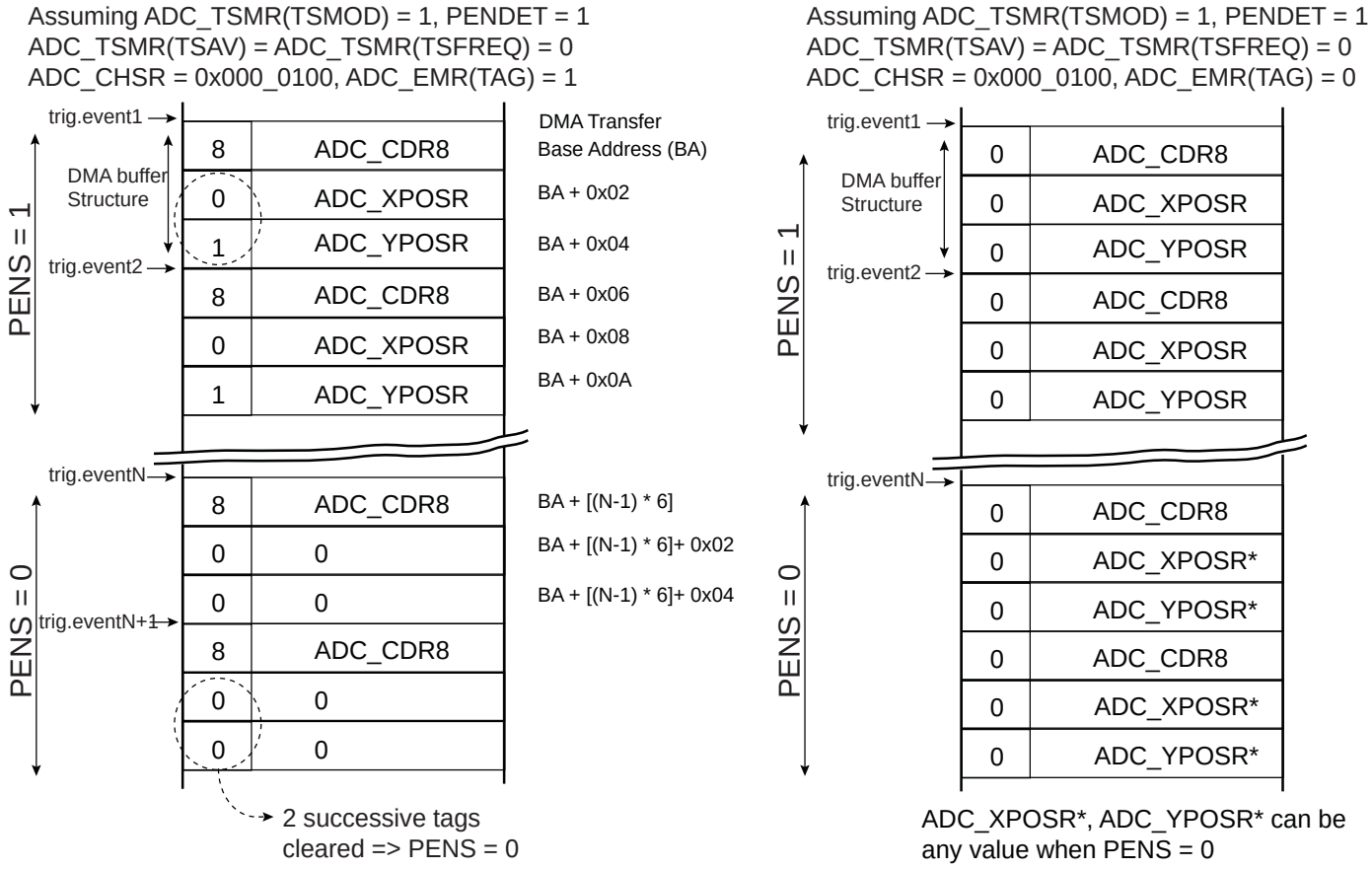
41.6.11.4 Pen Detection Status

If the pen detection measure is enabled (PENDET is set in ADC_TSMR), the XPOS, YPOS, Z1, Z2 values transmitted to the buffer through ADC_LCDR are cleared (including the CHNB field), if the PENS flag of ADC_ISR is 0. When the PENS flag is set, XPOS, YPOS, Z1, Z2 are normally transmitted.

Therefore, using pen detection together with tag function eases the post-processing of the buffer, especially to determine which touch-screen converted values correspond to a period of time when the pen was in contact with the screen.

When the pen detection is disabled or the tag function is disabled, XPOS, YPOS, Z1, Z2 are normally transmitted without tag and no relationship can be found with pen status, thus post-processing may not be easy.

Figure 41-16: Buffer Structure With and Without Pen Detection Enabled



41.6.12 Register Write Protection

To prevent any single software error from corrupting ADC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the "ADC Write Protection Mode Register" (ADC_WPMR).

If a write access to the protected registers is detected, the WPVS flag in the "ADC Write Protection Status Register" (ADC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS flag is automatically reset by reading ADC_WPSR.

The following registers can be write-protected:

- ADC Mode Register
- ADC Channel Sequence 1 Register
- ADC Channel Enable Register
- ADC Channel Disable Register
- ADC Extended Mode Register
- ADC Compare Window Register
- ADC Analog Control Register
- ADC Touchscreen Mode Register
- ADC Trigger Register

© 2017 Microchip Technology Inc.

41.7 Analog-to-Digital (ADC) User Interface

Table 41-4: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	ADC_CR	Write-only	–
0x04	Mode Register	ADC_MR	Read/Write	0x00000000
0x08	Channel Sequence Register 1	ADC_SEQR1	Read/Write	0x00000000
0x0C	Reserved	–	–	–
0x10	Channel Enable Register	ADC_CHER	Write-only	–
0x14	Channel Disable Register	ADC_CHDR	Write-only	–
0x18	Channel Status Register	ADC_CHSR	Read-only	0x00000000
0x1C	Reserved	–	–	–
0x20	Last Converted Data Register	ADC_LCDR	Read-only	0x00000000
0x24	Interrupt Enable Register	ADC_IER	Write-only	–
0x28	Interrupt Disable Register	ADC_IDR	Write-only	–
0x2C	Interrupt Mask Register	ADC_IMR	Read-only	0x00000000
0x30	Interrupt Status Register	ADC_ISR	Read-only	0x00000000
0x34	Reserved	–	–	–
0x38	Reserved	–	–	–
0x3C	Overrun Status Register	ADC_OVER	Read-only	0x00000000
0x40	Extended Mode Register	ADC_EMR	Read/Write	0x00000000
0x44	Compare Window Register	ADC_CWR	Read/Write	0x00000000
0x50	Channel Data Register 0	ADC_CDR0	Read-only	0x00000000
0x54	Channel Data Register 1	ADC_CDR1	Read-only	0x00000000
...	...	...	...	...
0x7C	Channel Data Register 11	ADC_CDR11	Read-only	0x00000000
0x80–0x90	Reserved	–	–	–
0x94	Analog Control Register	ADC_ACR	Read/Write	0x00000100
0x98–0xAC	Reserved	–	–	–
0xB0	Touchscreen Mode Register	ADC_TSMR	Read/Write	0x00000000
0xB4	Touchscreen X Position Register	ADC_XPOSR	Read-only	0x00000000
0xB8	Touchscreen Y Position Register	ADC_YPOSR	Read-only	0x00000000
0xBC	Touchscreen Pressure Register	ADC_PRESSR	Read-only	0x00000000
0xC0	Trigger Register	ADC_TRGR	Read/Write	0x00000000
0xC4–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	ADC_WPMR	Read/Write	0x00000000
0xE8	Write Protection Status Register	ADC_WPSR	Read-only	0x00000000
0xEC–0xFC	Reserved	–	–	–

Note: Any offset not listed in the table must be considered as “reserved”.

41.7.1 ADC Control Register

Name:ADC_CR

Address:0xF804C000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	TSCALIB	START	SWRST

SWRST: Software Reset

0: No effect.

1: Resets the ADC, simulating a hardware reset.

START: Start Conversion

0: No effect.

1: Begins analog-to-digital conversion.

TSCALIB: Touchscreen Calibration

0: No effect.

1: Programs screen calibration (VDD/GND measurement)

If conversion is in progress, the calibration sequence starts at the beginning of a new conversion sequence. If no conversion is in progress, the calibration sequence starts at the second conversion sequence located after the TSCALIB command (Sleep mode, waiting for a trigger event).

TSCALIB measurement sequence does not affect the Last Converted Data Register (ADC_LCDR).

© 2017 Microchip Technology Inc.

41.7.2 ADC Mode Register

Name:ADC_MR

Address:0xF804C004

Access: Read/Write

31	30	29	28	27	26	25	24
USEQ	–	–	–	TRACKTIM			
23	22	21	20	19	18	17	16
–	–	–	–	STARTUP			
15	14	13	12	11	10	9	8
PRESCAL							
7	6	5	4	3	2	1	0
–	–	SLEEP	LOWRES	–			–

This register can only be written if the WPEN bit is cleared in the ADC Write Protection Mode Register.

LOWRES: Resolution

Value	Name	Description
0	BITS_10	10-bit resolution.
1	BITS_8	8-bit resolution

SLEEP: Sleep Mode

Value	Name	Description
0	NORMAL	Normal Mode: The ADC core and reference voltage circuitry are kept ON between conversions.
1	SLEEP	Sleep Mode: The ADC core and reference voltage circuitry are OFF between conversions.

PRESCAL: Prescaler Rate Selection

PRESCAL = ($f_{peripheral\ clock}$ / (2 × f_{ADCCLK})) - 1.

STARTUP: Startup Time

Value	Name	Description
0	SUT0	0 periods of ADCCLK
1	SUT8	8 periods of ADCCLK
2	SUT16	16 periods of ADCCLK
3	SUT24	24 periods of ADCCLK
4	SUT64	64 periods of ADCCLK
5	SUT80	80 periods of ADCCLK
6	SUT96	96 periods of ADCCLK
7	SUT112	112 periods of ADCCLK
8	SUT512	512 periods of ADCCLK
9	SUT576	576 periods of ADCCLK
10	SUT640	640 periods of ADCCLK
11	SUT704	704 periods of ADCCLK
12	SUT768	768 periods of ADCCLK

© 2017 Microchip Technology Inc.

Value	Name	Description
13	SUT832	832 periods of ADCCLK
14	SUT896	896 periods of ADCCLK
15	SUT960	960 periods of ADCCLK

TRACKTIM: Tracking Time

Tracking Time = (TRACKTIM + 1) × ADCCLK periods

USEQ: Use Sequence Enable

Value	Name	Description
0	NUM_ORDER	Normal Mode: The controller converts channels in a simple numeric order depending only on the channel index.
1	REG_ORDER	User Sequence Mode: The sequence respects what is defined in ADC_SEQR1 register and can be used to convert the same channel several times.

© 2017 Microchip Technology Inc.

41.7.3 ADC Channel Sequence 1 Register

Name:ADC_SEQR1

Address:0xF804C008

Access: Read/Write

31	30	29	28	27	26	25	24
USCH8				USCH7			
23	22	21	20	19	18	17	16
USCH6				USCH5			
15	14	13	12	11	10	9	8
USCH4				USCH3			
7	6	5	4	3	2	1	0
USCH2				USCH1			

This register can only be written if the WPEN bit is cleared in the ADC Write Protection Mode Register.

USCHx: User Sequence Number x

The allowed range is 0 up to 11, thus only the sequencer from CH0 to CH11 can be used.

This register activates only if the USEQ field in ADC_MR field is set to ‘1’.

Any USCHx field is processed only if the CHx-1 it in ADC_CHSR reads logical ‘1’, else any value written in USCHx does not add the corresponding channel in the conversion sequence.

Configuring the same value in different fields leads to multiple samples of the same channel during the conversion sequence. This can be done consecutively, or not, according to user needs.

When configuring consecutive fields with the same value, the associated channel is sampled as many time as the number of consecutive values, this part of the conversion sequence being triggered by a unique event.

© 2017 Microchip Technology Inc.

41.7.4 ADC Channel Enable Register

Name:ADC_CHER

Address:0xF804C010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

This register can only be written if the WPEN bit is cleared in the ADC Write Protection Mode Register.

CHx: Channel x Enable

0: No effect.

1: Enables the corresponding channel.

Note: If USEQ = 1 in ADC_MR, CHx corresponds to the enable of sequence number x+1 described in ADC_SEQR1 (e.g. CH0 enables sequence number USCH1).

© 2017 Microchip Technology Inc.

41.7.5 ADC Channel Disable Register

Name:ADC_CHDR

Address:0xF804C014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

This register can only be written if the WPEN bit is cleared in the ADC Write Protection Mode Register.

CHx: Channel x Disable

0: No effect.

1: Disables the corresponding channel.

Warning: If the corresponding channel is disabled during a conversion or if it is disabled and then reenabled during a conversion, its associated data and corresponding EOCx and GOVRE flags in ADC_ISR and OVREx flags in ADC_OVER are unpredictable.

41.7.6 ADC Channel Status Register

Name:ADC_CHSR

Address:0xF804C018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

CHx: Channel x Status

0: The corresponding channel is disabled.

1: The corresponding channel is enabled.

© 2017 Microchip Technology Inc.

41.7.7 ADC Last Converted Data Register

Name:ADC_LCDR

Address:0xF804C020

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CHNB				LDATA			
7	6	5	4	3	2	1	0
LDATA							

LDATA: Last Data Converted

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed.

CHNB: Channel Number

Indicates the last converted channel when the TAG bit is set in ADC_EMR. If the TAG bit is not set, CHNB = 0.

41.7.8 ADC Interrupt Enable Register

Name:ADC_IER

Address:0xF804C024

Access: Write-only

31	30	29	28	27	26	25	24
–	NOPEN	PEN	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–	PRDY	YRDY	XRDY	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

EOCx: End of Conversion Interrupt Enable x

XRDY: Touchscreen Measure XPOS Ready Interrupt Enable

YRDY: Touchscreen Measure YPOS Ready Interrupt Enable

PRDY: Touchscreen Measure Pressure Ready Interrupt Enable

DRDY: Data Ready Interrupt Enable

GOVRE: General Overrun Error Interrupt Enable

COMPE: Comparison Event Interrupt Enable

PEN: Pen Contact Interrupt Enable

NOPEN: No Pen Contact Interrupt Enable

© 2017 Microchip Technology Inc.

41.7.9 ADC Interrupt Disable Register

Name:ADC_IDR

Address:0xF804C028

Access: Write-only

31	30	29	28	27	26	25	24
–	NOPEN	PEN	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–	PRDY	YRDY	XRDY	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

EOCx: End of Conversion Interrupt Disable x

XRDY: Touchscreen Measure XPOS Ready Interrupt Disable

YRDY: Touchscreen Measure YPOS Ready Interrupt Disable

PRDY: Touchscreen Measure Pressure Ready Interrupt Disable

DRDY: Data Ready Interrupt Disable

GOVRE: General Overrun Error Interrupt Disable

COMPE: Comparison Event Interrupt Disable

PEN: Pen Contact Interrupt Disable

NOPEN: No Pen Contact Interrupt Disable

41.7.10 ADC Interrupt Mask Register

Name:ADC_IMR

Address:0xF804C02C

Access: Read-only

31	30	29	28	27	26	25	24
–	NOPEN	PEN	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–	PRDY	YRDY	XRDY	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

EOCx: End of Conversion Interrupt Mask x

XRDY: Touchscreen Measure XPOS Ready Interrupt Mask

YRDY: Touchscreen Measure YPOS Ready Interrupt Mask

PRDY: Touchscreen Measure Pressure Ready Interrupt Mask

DRDY: Data Ready Interrupt Mask

GOVRE: General Overrun Error Interrupt Mask

COMPE: Comparison Event Interrupt Mask

PEN: Pen Contact Interrupt Mask

NOPEN: No Pen Contact Interrupt Mask

© 2017 Microchip Technology Inc.

41.7.11 ADC Interrupt Status Register

Name:ADC_ISR

Address:0xF804C030

Access: Read-only

31	30	29	28	27	26	25	24
PENS	NOPEN	PEN	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–	PRDY	YRDY	XRDY	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

EOCx: End of Conversion x (automatically set / cleared)

0: The corresponding analog channel is disabled, or the conversion is not finished. This flag is cleared when reading the corresponding ADC_CDRx registers.

1: The corresponding analog channel is enabled and conversion is complete.

XRDY: Touchscreen XPOS Measure Ready (cleared on read)

0: No measure has been performed since the last read of ADC_XPOSR.

1: At least one measure has been performed since the last read of ADC_ISR.

YRDY: Touchscreen YPOS Measure Ready (cleared on read)

0: No measure has been performed since the last read of ADC_YPOSR.

1: At least one measure has been performed since the last read of ADC_ISR.

PRDY: Touchscreen Pressure Measure Ready (cleared on read)

0: No measure has been performed since the last read of ADC_PRESSR.

1: At least one measure has been performed since the last read of ADC_ISR.

DRDY: Data Ready (automatically set / cleared)

0: No data has been converted since the last read of ADC_LCDR.

1: At least one data has been converted and is available in ADC_LCDR.

GOVRE: General Overrun Error (cleared on read)

0: No general overrun error occurred since the last read of ADC_ISR.

1: At least one general overrun error has occurred since the last read of ADC_ISR.

COMPE: Comparison Event (cleared on read)

0: No comparison event since the last read of ADC_ISR.

1: At least one comparison event (defined in ADC_EMR and ADC_CWR) has occurred since the last read of ADC_ISR.

PEN: Pen contact (cleared on read)

0: No pen contact since the last read of ADC_ISR.

1: At least one pen contact since the last read of ADC_ISR.

NOPEN: No Pen Contact (cleared on read)

0: No loss of pen contact since the last read of ADC_ISR.

1: At least one loss of pen contact since the last read of ADC_ISR.

PENS: Pen Detect Status

0: The pen does not press the screen.

1: The pen presses the screen.

Note: PENS is not a source of interruption.

© 2017 Microchip Technology Inc.

41.7.12 ADC Overrun Status Register

Name:ADC_OVER

Address:0xF804C03C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	OVRE11	OVRE10	OVRE9	OVRE8
7	6	5	4	3	2	1	0
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0

OVREx: Overrun Error x

0: No overrun error on the corresponding channel since the last read of ADC_OVER.

1: An overrun error has occurred on the corresponding channel since the last read of ADC_OVER.

41.7.13 ADC Extended Mode Register

Name:ADC_EMR

Address:0xF804C040

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	TAG
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	CMPALL	–
7	6	5	4	3	2	1	0
CMPSEL				–	–	CMPMODE	

This register can only be written if the WPEN bit is cleared in the ADC Write Protection Mode Register.

CMPMODE: Comparison Mode

Value	Name	Description
0	LOW	Generates an event when the converted data is lower than the low threshold of the window.
1	HIGH	Generates an event when the converted data is higher than the high threshold of the window.
2	IN	Generates an event when the converted data is in the comparison window.
3	OUT	Generates an event when the converted data is out of the comparison window.

CMPSEL: Comparison Selected Channel

If CMPALL = 0: CMPSEL indicates which channel has to be compared.

If CMPALL = 1: No effect.

CMPALL: Compare All Channels

0: Only channel indicated in CMPSEL field is compared.

1: All channels are compared.

TAG: Tag of ADC_LCDR

0: Sets CHNB field to zero in ADC_LCDR.

1: Appends the channel number to the conversion result in ADC_LCDR.

© 2017 Microchip Technology Inc.

41.7.14 ADC Compare Window Register

Name:ADC_CWR

Address:0xF804C044

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	HIGHTHRES			
23	22	21	20	19	18	17	16
HIGHTHRES							
15	14	13	12	11	10	9	8
–	–	–	–	LOWTHRES			
7	6	5	4	3	2	1	0
LOWTHRES							

This register can only be written if the WPEN bit is cleared in the ADC Write Protection Mode Register.

LOWTHRES: Low Threshold

Low threshold associated to compare settings of ADC_EMR.

If LOWRES is set in ADC_MR, only the 10 LSB of LOWTHRES must be programmed. The two LSB will be automatically discarded to match the value carried on ADC_CDR (8-bit).

HIGHTHRES: High Threshold

High threshold associated to compare settings of ADC_EMR.

If LOWRES is set in ADC_MR, only the 10 LSB of HIGHTHRES must be programmed. The two LSB will be automatically discarded to match the value carried on ADC_CDR (8-bit).

© 2017 Microchip Technology Inc.

41.7.15 ADC Channel Data Register

Name:ADC_CDRx [x=0..11]

Address:0xF804C050

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	DATA			
7	6	5	4	3	2	1	0
DATA							

DATA: Converted Data

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed. ADC_CDRx is only loaded if the corresponding analog channel is enabled.

© 2017 Microchip Technology Inc.

41.7.16 ADC Analog Control Register

Name:ADC_ACR

Address:0xF804C094

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	PENDETSENS	

This register can only be written if the WPEN bit is cleared in the ADC Write Protection Mode Register.

PENDETSENS: Pen Detection Sensitivity

Modifies the pen detection input pull-up resistor value. See the section ‘Electrical Characteristics’ for further details.

41.7.17 ADC Touchscreen Mode Register

Name:ADC_TSMR

Address:0xF804C0B0

Access: Read/Write

31	30	29	28	27	26	25	24
PENDBC				–	–	–	PENDET
23	22	21	20	19	18	17	16
–	NOTSDMA	–	–	TSSCTIM			
15	14	13	12	11	10	9	8
–	–	–	–	TSFREQ			
7	6	5	4	3	2	1	0
–	–	TSAV		–	–	TSMODE	

This register can only be written if the WPEN bit is cleared in the ADC Write Protection Mode Register.

TSMODE: Touchscreen Mode

Value	Name	Description
0	NONE	No Touchscreen
1	4_WIRE_NO_PM	4-wire Touchscreen without pressure measurement
2	4_WIRE	4-wire Touchscreen with pressure measurement
3	5_WIRE	5-wire Touchscreen

When TSMOD equals 01 or 10 (i.e., 4-wire mode), channels 0, 1, 2 and 3 must not be used for classic ADC conversions. When TSMOD equals 11 (i.e., 5-wire mode), channels 0, 1, 2, 3, and 4 must not be used.

TSAV: Touchscreen Average

Value	Name	Description
0	NO_FILTER	No Filtering. Only one ADC conversion per measure
1	AVG2CONV	Averages 2 ADC conversions
2	AVG4CONV	Averages 4 ADC conversions
3	AVG8CONV	Averages 8 ADC conversions

TSFREQ: Touchscreen Frequency

Defines the touchscreen frequency compared to the trigger frequency.

TSFREQ must be greater or equal to TSAV.

The touchscreen frequency is:

Touchscreen Frequency = Trigger Frequency / 2^{TSFREQ}

TSSCTIM: Touchscreen Switches Closure Time

Defines closure time of analog switches necessary to establish the measurement conditions.

The closure time is:

Switch Closure Time = (TSSCTIM $\times$ 4) ADCCLK periods.

© 2017 Microchip Technology Inc.

PENDET: Pen Contact Detection Enable

0: Pen contact detection disabled.

1: Pen contact detection enabled.

When PENDET = 1, XPOS, YPOS, Z1, Z2 values of ADC_XPOSR, ADC_YPOSR, ADC_PRESSR are automatically cleared when PENS = 0 in ADC_ISR.

NOTSDMA: No TouchScreen DMA

0: XPOS, YPOS, Z1, Z2 are transmitted in ADC_LCDR.

1: XPOS, YPOS, Z1, Z2 are never transmitted in ADC_LCDR, therefore the buffer does not contains touchscreen values.

PENDBC: Pen Detect Debouncing Period

Debouncing period = 2^{PENDBC} ADCCLK periods.

41.7.18 ADC Touchscreen X Position Register

Name:ADC_XPOSR

Address:0xF804C0B4

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	XSCALE			
23	22	21	20	19	18	17	16
XSCALE							
15	14	13	12	11	10	9	8
–	–	–	–	XPOS			
7	6	5	4	3	2	1	0
XPOS							

XPOS: X Position

The position measured is stored here. If XPOS = 0 or XPOS = XSIZE, the pen is on the border.

When pen detection is enabled (PENDET set to ‘1’ in ADC_TSMR), XPOS is tied to 0 while there is no detection of contact on the touchscreen (i.e., when PENS bit is cleared in ADC_ISR).

XSCALE: Scale of XPOS

Indicates the max value that XPOS can reach. This value should be close to 2^{10}.

© 2017 Microchip Technology Inc.

41.7.19 ADC Touchscreen Y Position Register

Name:ADC_YPOSR

Address:0xF804C0B8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	YSCALE			
23	22	21	20	19	18	17	16
YSCALE							
15	14	13	12	11	10	9	8
–	–	–	–	YPOS			
7	6	5	4	3	2	1	0
YPOS							

YPOS: Y Position

The position measured is stored here. If YPOS = 0 or YPOS = YSIZE, the pen is on the border.

When pen detection is enabled (PENDET set to ‘1’ in ADC_TSMR), YPOS is tied to 0 while there is no detection of contact on the touch-screen (i.e., when PENS bit is cleared in ADC_ISR).

YSCALE: Scale of YPOS

Indicates the max value that YPOS can reach. This value should be close to 2^{10}.

41.7.20 ADC Touchscreen Pressure Register

Name:ADC_PRESSR

Address:0xF804C0BC

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	Z2			
23	22	21	20	19	18	17	16
Z2							
15	14	13	12	11	10	9	8
–	–	–	–	Z1			
7	6	5	4	3	2	1	0
Z1							

Z1: Data of Z1 Measurement

Data Z1 necessary to calculate pen pressure.

When pen detection is enabled (PENDET set to ‘1’ in ADC_TSMR), Z1 is tied to 0 while there is no detection of contact on the touchscreen (i.e., when PENS bit is cleared in ADC_ISR).

Z2: Data of Z2 Measurement

Data Z2 necessary to calculate pen pressure.

When pen detection is enabled (PENDET set to ‘1’ in ADC_TSMR), Z2 is tied to 0 while there is no detection of contact on the touchscreen (i.e., when PENS bit is cleared in ADC_ISR).

Note: These two values are unavailable if TSMODE is not set to 2 in ADC_TSMR.

© 2017 Microchip Technology Inc.

41.7.21 ADC Trigger Register

Name:ADC_TRGR

Address:0xF804C0C0

Access: Read/Write

31	30	29	28	27	26	25	24
TRGPER							
23	22	21	20	19	18	17	16
TRGPER							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	TRGMOD		

TRGMOD: Trigger Mode

Value	Name	Description
0	NO_TRIGGER	No trigger, only software trigger can start conversions
1	EXT_TRIG_RISE	External trigger rising edge
2	EXT_TRIG_FALL	External trigger falling edge
3	EXT_TRIG_ANY	External trigger any edge
4	PEN_TRIG	Pen Detect Trigger (shall be selected only if PENDET is set and TSAMOD = Touchscreen only mode)
5	PERIOD_TRIG	ADC internal periodic trigger (see field TRGPER)
6	CONTINUOUS	Continuous Mode

TRGPER: Trigger Period

Effective only if TRGMOD defines a periodic trigger.

Defines the periodic trigger period, with the following equation:

Trigger Period = (TRGPER + 1) / ADCCLK

The minimum time between two consecutive trigger events must be strictly greater than the duration time of the longest conversion sequence depending on the configuration of registers ADC_MR, ADC_CHSR, ADC_SEQRx, ADC_TSMR.

When TRGMOD is set to pen detect trigger (i.e., 100) and averaging is used (i.e., field TSAV ≠ 0 in ADC_TSMR) only one measure is performed. Thus, XRDY, YRDY, PRDY, DRDY will not rise on pen contact trigger. To achieve measurement, several triggers must be provided either by software or by setting the TRGMOD on continuous trigger (i.e., 110) until flags rise.

41.7.22 ADC Write Protection Mode Register

Name:ADC_WPMR

Address:0xF804C0E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

WPEN: Write Protection Enable

0: Disables the write protection if WPKEY value corresponds to 0x414443 (“ADC” in ASCII).

1: Enables the write protection if WPKEY value corresponds to 0x414443 (“ADC” in ASCII).

See Section 41.6.12 “Register Write Protection” for the list of write-protected registers.

WPKEY: Write Protection Key

Value	Name	Description
0x414443	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0

© 2017 Microchip Technology Inc.

41.7.23 ADC Write Protection Status Register

Name:ADC_WPSR

Address:0xF804C0E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
WPVSRC							
15	14	13	12	11	10	9	8
WPVSRC							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

WPVS: Write Protection Violation Status

0: No write protection violation has occurred since the last read of ADC_WPSR.

1: A write protection violation has occurred since the last read of ADC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

WPVSRC: Write Protection Violation Source

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

42. Synchronous Serial Controller (SSC)

42.1 Description

The Synchronous Serial Controller (SSC) provides a synchronous communication link with external devices. It supports many serial synchronous communication protocols generally used in audio and telecom applications such as I2S, Short Frame Sync, Long Frame Sync, etc.

The SSC contains an independent receiver and transmitter and a common clock divider. The receiver and the transmitter each interface with three signals: the TD/RD signal for data, the TK/RK signal for the clock and the TF/RF signal for the Frame Sync. The transfers can be programmed to start automatically or on different events detected on the Frame Sync signal.

The SSC high-level of programmability and its use of DMA permit a continuous high bit rate data transfer without processor intervention.

Featuring connection to the DMA, the SSC permits interfacing with low processor overhead to the following:

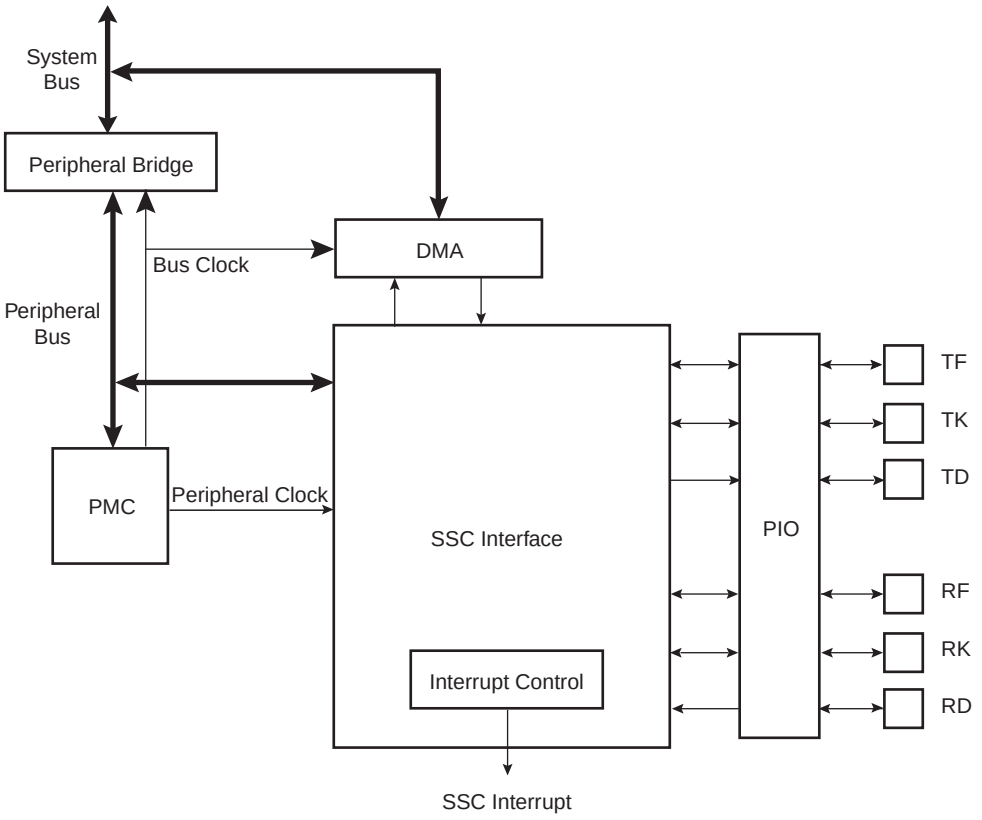
- Codecs in master or slave mode
- DAC through dedicated serial interface, particularly I2S
- Magnetic card reader

42.2 Embedded Characteristics

- Provides Serial Synchronous Communication Links Used in Audio and Telecom Applications
- Contains an Independent Receiver and Transmitter and a Common Clock Divider
- Interfaced with the DMA Controller (DMAC) to Reduce Processor Overhead
- Offers a Configurable Frame Sync and Data Length
- Receiver and Transmitter Can be Programmed to Start Automatically or on Detection of Different Events on the Frame Sync Signal
- Receiver and Transmitter Include a Data Signal, a Clock Signal and a Frame Synchronization Signal

42.3 Block Diagram

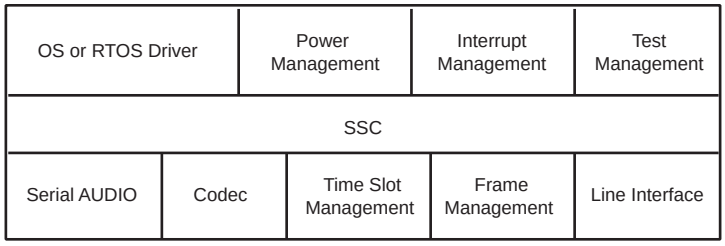
Figure 42-1: Block Diagram



© 2017 Microchip Technology Inc.

42.4 Application Block Diagram

Figure 42-2: Application Block Diagram



42.5 SSC Application Examples

The SSC can support several serial communication modes used in audio or high speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the SSC are not listed here.

Figure 42-3: Audio Application Block Diagram

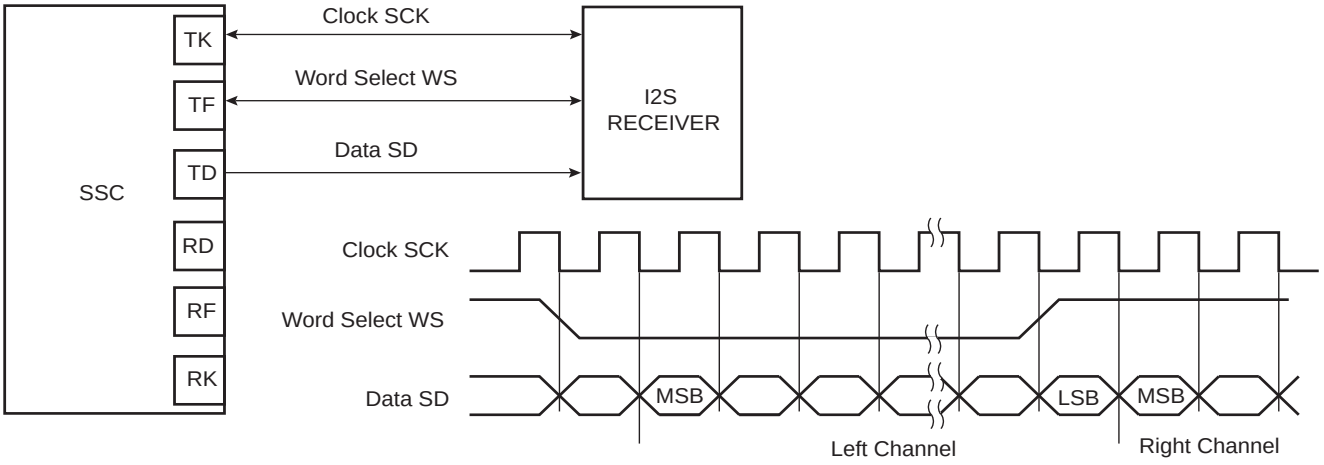


Figure 42-4: Codec Application Block Diagram

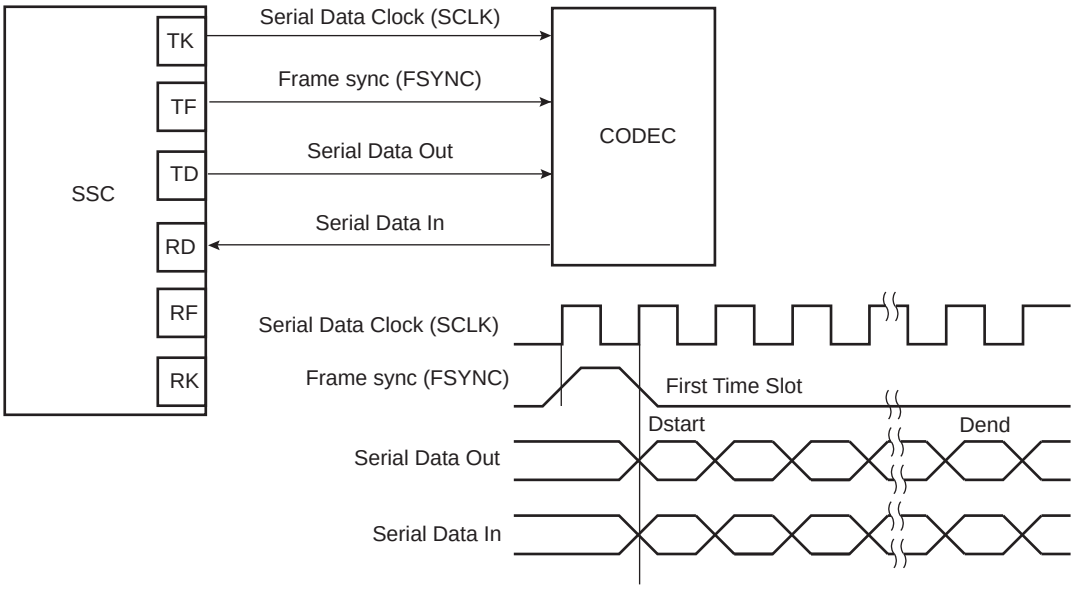
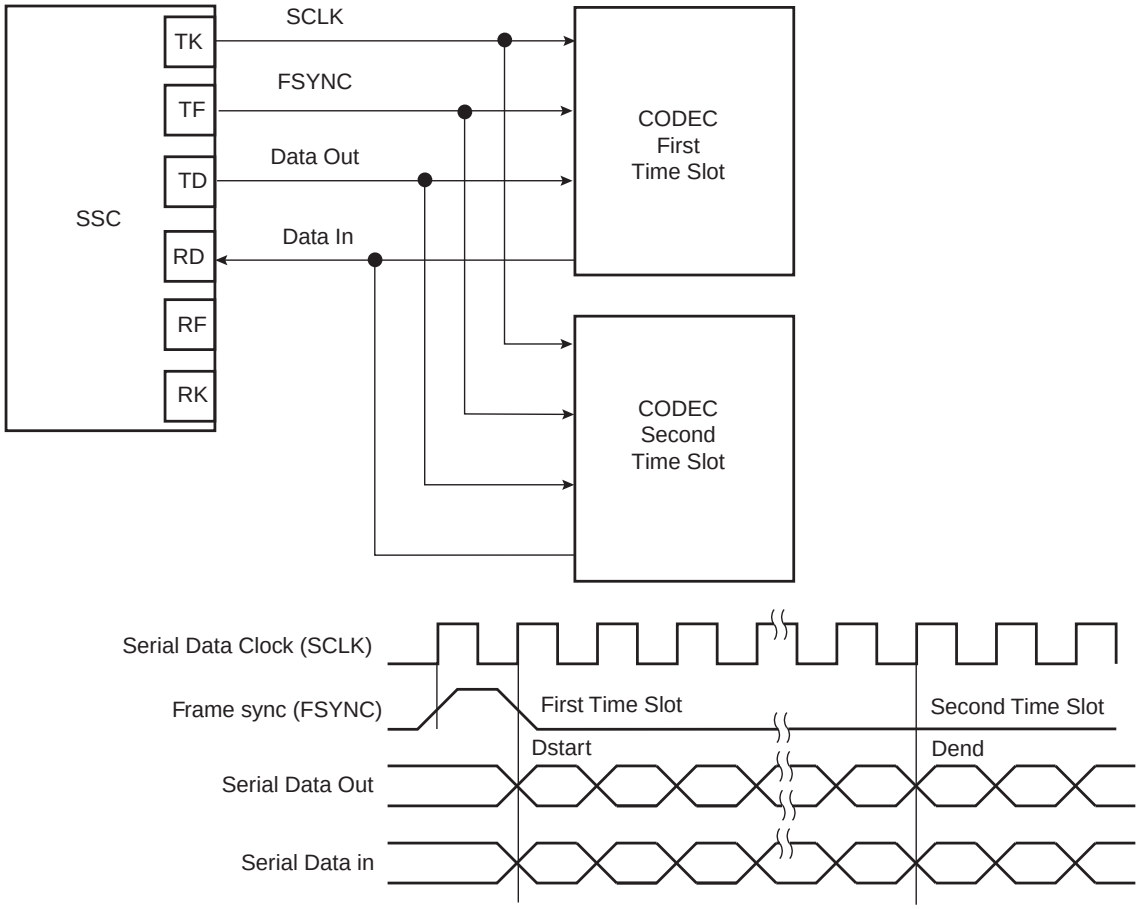


Figure 42-5: Time Slot Application Block Diagram



© 2017 Microchip Technology Inc.

42.6 Pin Name List

Table 42-1: I/O Lines Description

Pin Name	Pin Description	Type
RF	Receiver Frame Synchro	Input/Output
RK	Receiver Clock	Input/Output
RD	Receiver Data	Input
TF	Transmitter Frame Synchro	Input/Output
TK	Transmitter Clock	Input/Output
TD	Transmitter Data	Output

42.7 Product Dependencies

42.7.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the SSC receiver, the PIO controller must be configured to dedicate the SSC receiver I/O lines to the SSC peripheral mode.

Before using the SSC transmitter, the PIO controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

Table 42-2: I/O Lines

Instance	Signal	I/O Line	Peripheral
SSC	RD	PA27	B
SSC	RF	PA29	B
SSC	RK	PA28	B
SSC	TD	PA26	B
SSC	TF	PA25	B
SSC	TK	PA24	B

42.7.2 Power Management

The SSC is not continuously clocked. The SSC interface may be clocked through the Power Management Controller (PMC), therefore the programmer must first configure the PMC to enable the SSC clock.

42.7.3 Interrupt

The SSC interface has an interrupt line connected to the interrupt controller. Handling interrupts requires programming the interrupt controller before configuring the SSC.

All SSC interrupts can be enabled/disabled configuring the SSC Interrupt Mask Register. Each pending and unmasked SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC Interrupt Status Register.

Table 42-3: Peripheral IDs

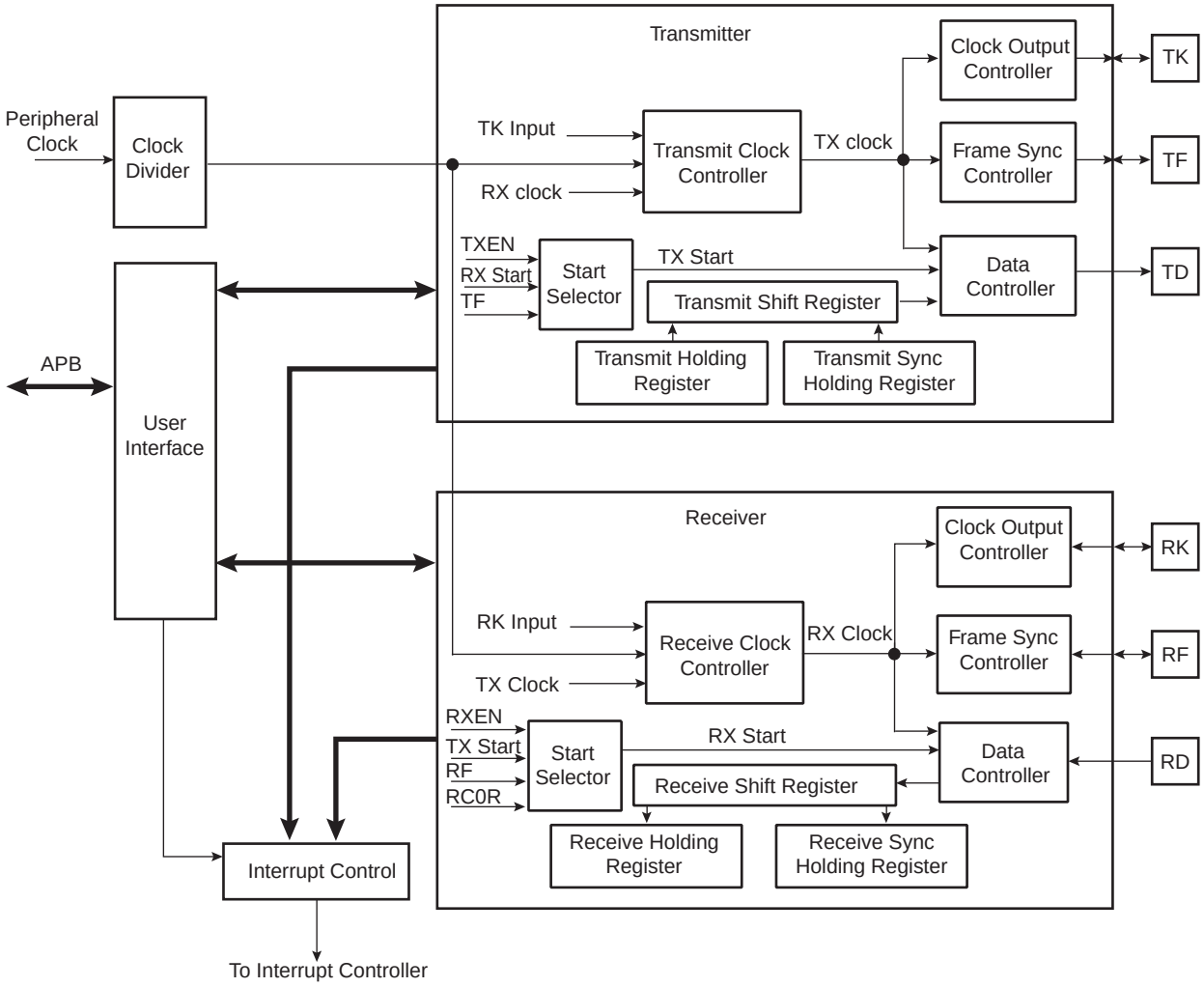
Instance	ID
SSC	28

42.8 Functional Description

This chapter contains the functional description of the following: SSC Functional Block, Clock Management, Data format, Start, Transmitter, Receiver and Frame Sync.

The receiver and transmitter operate separately. However, they can work synchronously by programming the receiver to use the transmit clock and/or to start a data transfer when transmission starts. Alternatively, this can be done by programming the transmitter to use the receive clock and/or to start a data transfer when reception starts. The transmitter and the receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. The maximum clock speed allowed on the TK and RK pins is the peripheral clock divided by 2.

Figure 42-6: SSC Functional Block Diagram



© 2017 Microchip Technology Inc.

42.8.1 Clock Management

The transmitter clock can be generated by:

- an external clock received on the TK I/O pad
- the receiver clock
- the internal clock divider

The receiver clock can be generated by:

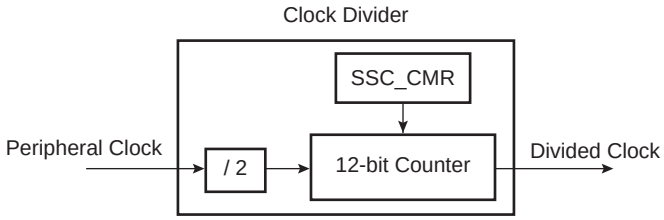
- an external clock received on the RK I/O pad
- the transmitter clock
- the internal clock divider

Furthermore, the transmitter block can generate an external clock on the TK I/O pad, and the receiver block can generate an external clock on the RK I/O pad.

This allows the SSC to support many Master and Slave Mode data transfers.

42.8.1.1 Clock Divider

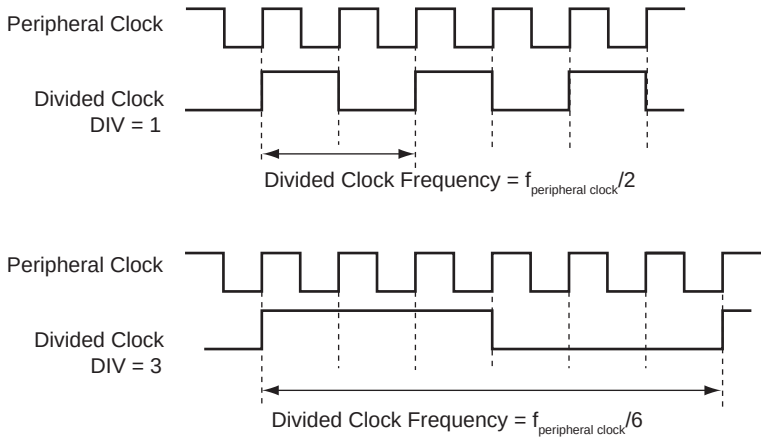
Figure 42-7: Divided Clock Block Diagram



The peripheral clock divider is determined by the 12-bit field DIV counter and comparator (so its maximal value is 4095) in the Clock Mode Register (SSC_CMR), allowing a peripheral clock division by up to 8190. The Divided Clock is provided to both the Receiver and Transmitter. When this field is programmed to 0, the Clock Divider is not used and remains inactive.

When DIV is set to a value equal to or greater than 1, the Divided Clock has a frequency of peripheral clock divided by 2 times DIV. Each level of the Divided Clock has a duration of the peripheral clock multiplied by DIV. This ensures a 50% duty cycle for the Divided Clock regardless of whether the DIV value is even or odd.

Figure 42-8: Divided Clock Generation

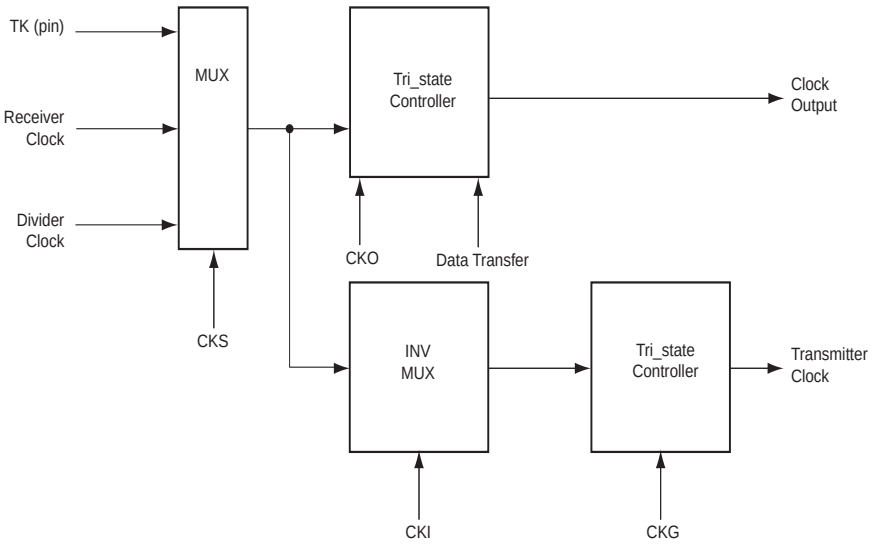


42.8.1.2 Transmitter Clock Management

The transmitter clock is generated from the receiver clock or the divider clock or an external clock scanned on the TK I/O pad. The transmitter clock is selected by the CKS field in the Transmit Clock Mode Register (SSC_TCMR). Transmit Clock can be inverted independently by the CKI bits in the SSC_TCMR.

The transmitter can also drive the TK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_TCMR. The Transmit Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the SSC_TCMR to select TK pin (CKS field) and at the same time Continuous Transmit Clock (CKO field) can lead to unpredictable results.

Figure 42-9: Transmitter Clock Management

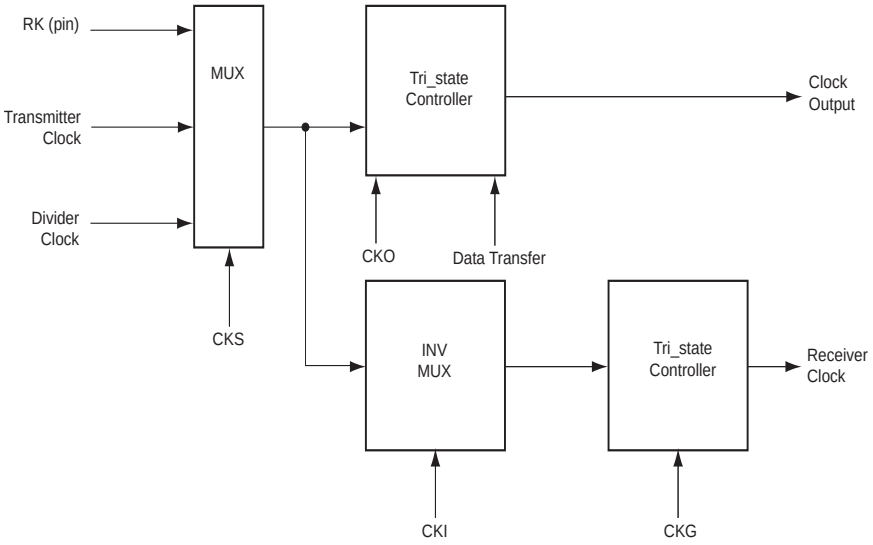


42.8.1.3 Receiver Clock Management

The receiver clock is generated from the transmitter clock or the divider clock or an external clock scanned on the RK I/O pad. The Receive Clock is selected by the CKS field in SSC_RCMR (Receive Clock Mode Register). Receive Clocks can be inverted independently by the CKI bits in SSC_RCMR.

The receiver can also drive the RK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_RCMR. The Receive Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the SSC_RCMR to select RK pin (CKS field) and at the same time Continuous Receive Clock (CKO field) can lead to unpredictable results.

Figure 42-10: Receiver Clock Management



© 2017 Microchip Technology Inc.

42.8.1.4 Serial Clock Ratio Considerations

The Transmitter and the Receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. In this case, the maximum clock speed allowed on the RK pin is:

- Peripheral clock divided by 2 if Receiver Frame Synchro is input
- Peripheral clock divided by 3 if Receiver Frame Synchro is output

In addition, the maximum clock speed allowed on the TK pin is:

- Peripheral clock divided by 6 if Transmit Frame Synchro is input
- Peripheral clock divided by 2 if Transmit Frame Synchro is output

42.8.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

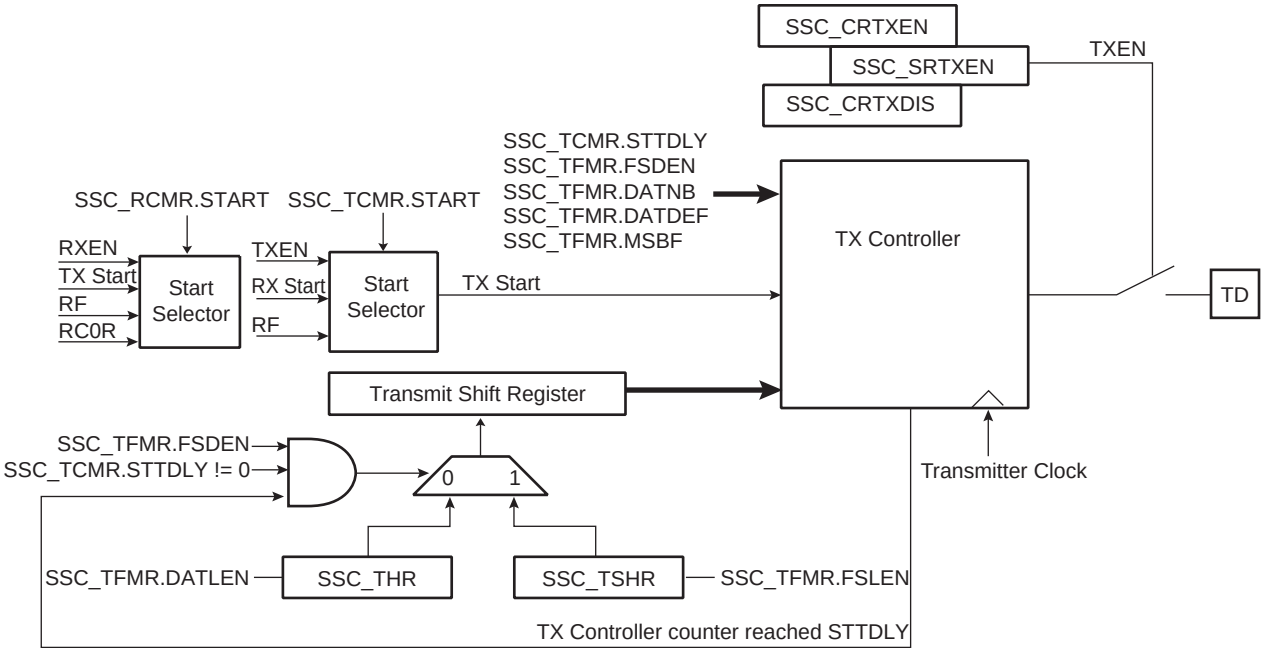
The start event is configured by setting the SSC_TCMR. See Section 42.8.4 “Start”.

The frame synchronization is configured setting the Transmit Frame Mode Register (SSC_TFMR). See Section 42.8.5 “Frame Sync”.

To transmit data, the transmitter uses a shift register clocked by the transmitter clock signal and the start mode selected in the SSC_TCMR. Data is written by the application to the SSC_THR then transferred to the shift register according to the data format selected.

When both the SSC_THR and the transmit shift register are empty, the status flag TXEMPTY is set in the SSC_SR. When the Transmit Holding register is transferred in the transmit shift register, the status flag TXRDY is set in the SSC_SR and additional data can be loaded in the holding register.

Figure 42-11: Transmitter Block Diagram



42.8.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

The start event is configured setting the Receive Clock Mode Register (SSC_RCMR). See Section 42.8.4 “Start”.

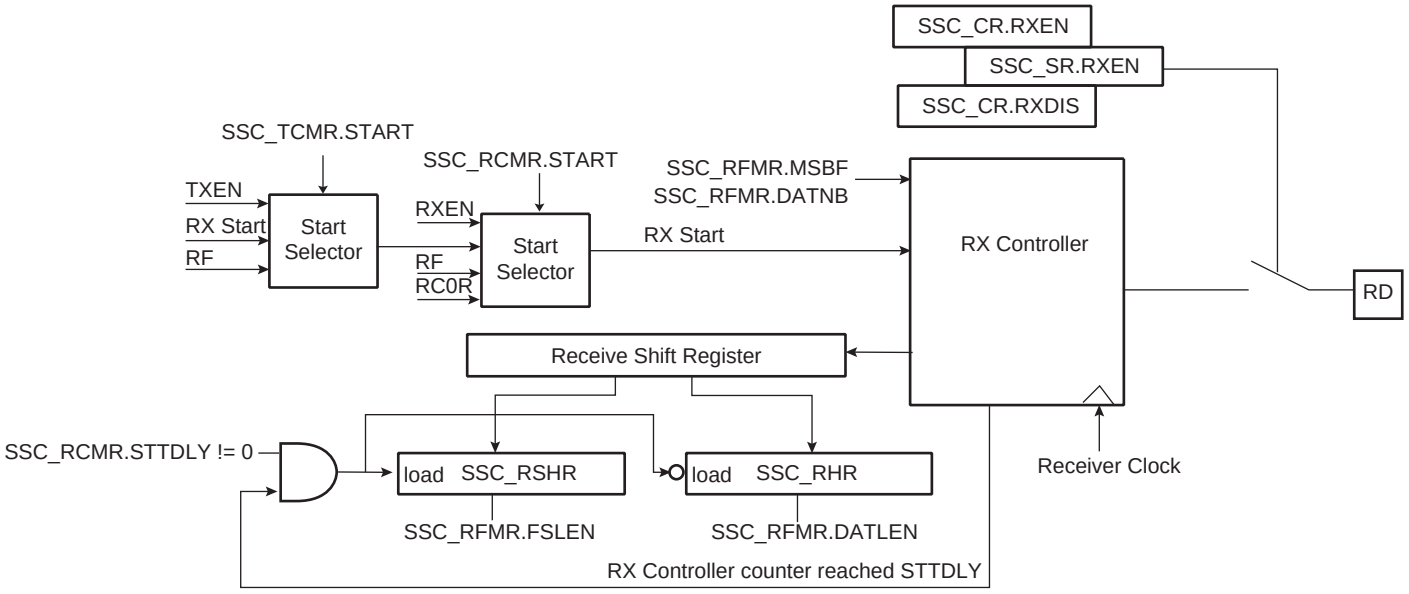
The frame synchronization is configured setting the Receive Frame Mode Register (SSC_RFMR). See Section 42.8.5 “Frame Sync”.

The receiver uses a shift register clocked by the receiver clock signal and the start mode selected in the SSC_RCMR. The data is transferred from the shift register depending on the data format selected.

When the receiver shift register is full, the SSC transfers this data in the holding register, the status flag RXRDY is set in the SSC_SR and the data can be read in the receiver holding register. If another transfer occurs before read of the Receive Holding Register (SSC_RHR), the status flag OVERUN is set in the SSC_SR and the receiver shift register is transferred in the SSC_RHR.

© 2017 Microchip Technology Inc.

Figure 42-12: Receiver Block Diagram



42.8.4 Start

The transmitter and receiver can both be programmed to start their operations when an event occurs, respectively in the Transmit Start Selection (START) field of SSC_TCMR and in the Receive Start Selection (START) field of SSC_RCMR.

Under the following conditions the start event is independently programmable:

- Continuous. In this case, the transmission starts as soon as a word is written in SSC_THR and the reception starts as soon as the Receiver is enabled.
- Synchronously with the transmitter/receiver
- On detection of a falling/rising edge on TF/RF
- On detection of a low level/high level on TF/RF
- On detection of a level change or an edge on TF/RF

A start can be programmed in the same manner on either side of the Transmit/Receive Clock Register (SSC_RCMR/SSC_TCMR). Thus, the start could be on TF (Transmit) or RF (Receive).

Moreover, the Receiver can start when data is detected in the bit stream with the Compare Functions.

Detection on TF/RF input/output is done by the field FSOS of the Transmit/Receive Frame Mode Register (SSC_TFMR/SSC_RFMR).

© 2017 Microchip Technology Inc.

Figure 42-13: Transmit Start Mode

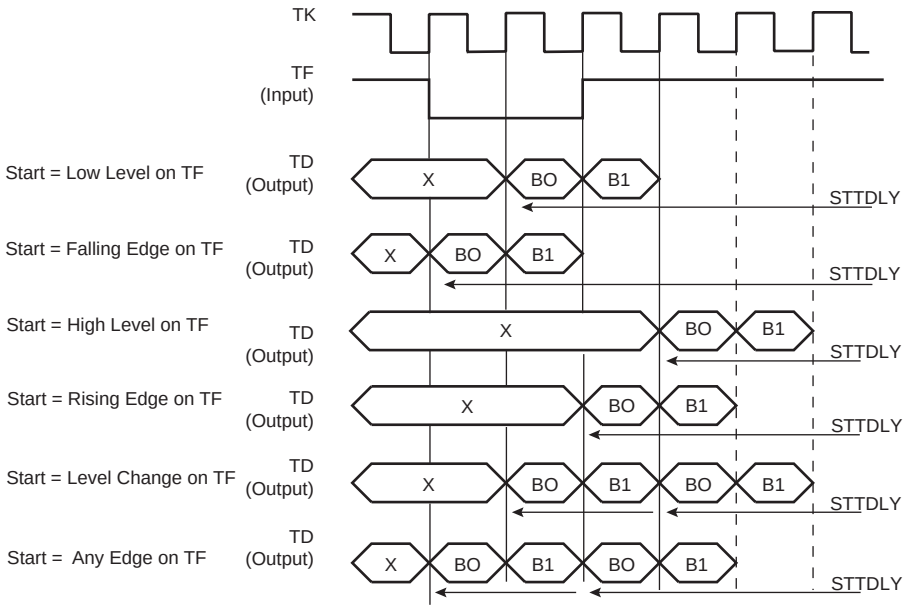
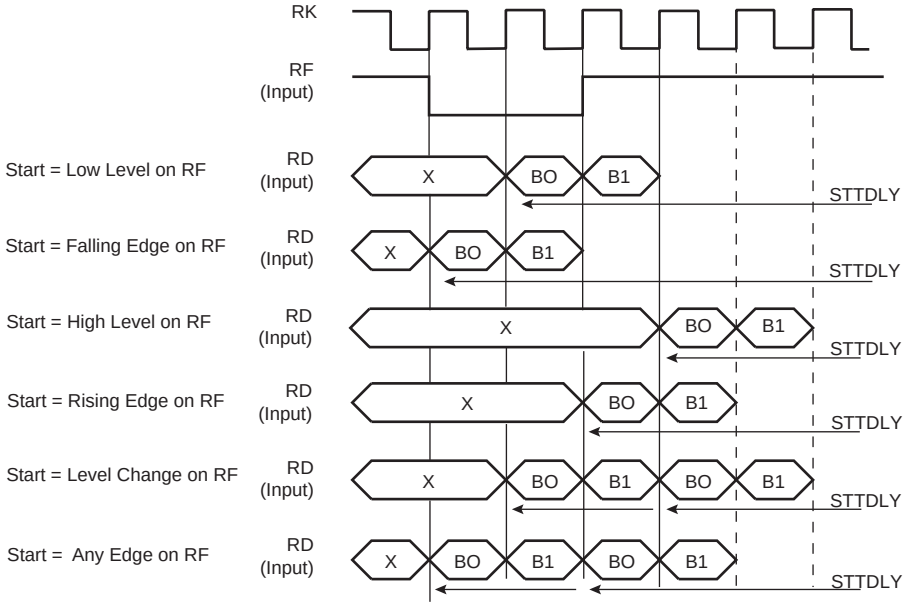


Figure 42-14: Receive Pulse/Edge Start Modes



42.8.5 Frame Sync

The Transmitter and Receiver Frame Sync pins, TF and RF, can be programmed to generate different kinds of frame synchronization signals. The Frame Sync Output Selection (FSOS) field in the Receive Frame Mode Register (SSC_RFMR) and in the Transmit Frame Mode Register (SSC_TFMR) are used to select the required waveform.

- Programmable low or high levels during data transfer are supported.
- Programmable high levels before the start of data transfers or toggling are also supported.

If a pulse waveform is selected, the Frame Sync Length (FSLEN) field in SSC_RFMR and SSC_TFMR programs the length of the pulse, from 1 bit time up to 256 bit times.

The periodicity of the Receive and Transmit Frame Sync pulse output can be programmed through the Period Divider Selection (PERIOD) field in SSC_RCMR and SSC_TCMR.

42.8.5.1 Frame Sync Data

Frame Sync Data transmits or receives a specific tag during the Frame Sync signal.

During the Frame Sync signal, the Receiver can sample the RD line and store the data in the Receive Sync Holding Register and the transmitter can transfer Transmit Sync Holding Register in the shift register. The data length to be sampled/shifted out during the Frame Sync signal is programmed by the FSLEN field in SSC_RFMR/SSC_TFMR and has a maximum value of 256.

Concerning the Receive Frame Sync Data operation, if the Frame Sync Length is equal to or lower than the delay between the start event and the actual data reception, the data sampling operation is performed in the Receive Sync Holding Register through the receive shift register.

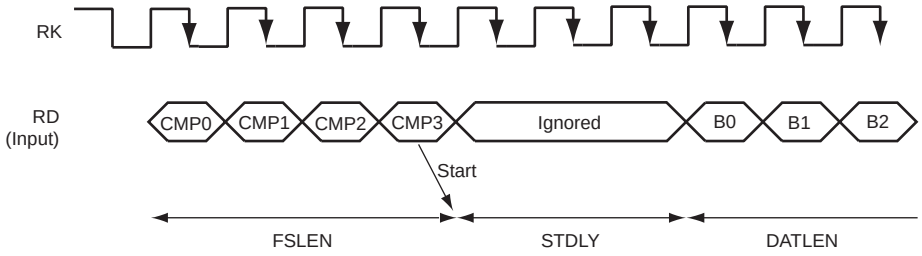
The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in SSC_TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register, then shifted out.

42.8.5.2 Frame Sync Edge Detection

The Frame Sync Edge detection is programmed by the FSEDGE field in SSC_RFMR/SSC_TFMR. This sets the corresponding flags RXSYN/TXSYN in the SSC Status Register (SSC_SR) on frame synchro edge detection (signals RF/TF).

42.8.6 Receive Compare Modes

Figure 42-15: Receive Compare Modes



42.8.6.1 Compare Functions

The length of the comparison patterns (Compare 0, Compare 1) and thus the number of bits they are compared to is defined by FSLEN, but with a maximum value of 256 bits. Comparison is always done by comparing the last bits received with the comparison pattern. Compare 0 can be one start event of the Receiver. In this case, the receiver compares at each new sample the last bits received at the Compare 0 pattern contained in the Compare 0 Register (SSC_RC0R). When this start event is selected, the user can program the Receiver to start a new data transfer either by writing a new Compare 0, or by receiving continuously until Compare 1 occurs. This selection is done with the STOP bit in the SSC_RCMR.

42.8.7 Data Format

The data framing format of both the transmitter and the receiver are programmable through the Transmitter Frame Mode Register (SSC_TFMR) and the Receiver Frame Mode Register (SSC_RFMR). In either case, the user can independently select the following parameters:

- Event that starts the data transfer (START)
- Delay in number of bit periods between the start event and the first data bit (STTDLY)

© 2017 Microchip Technology Inc.

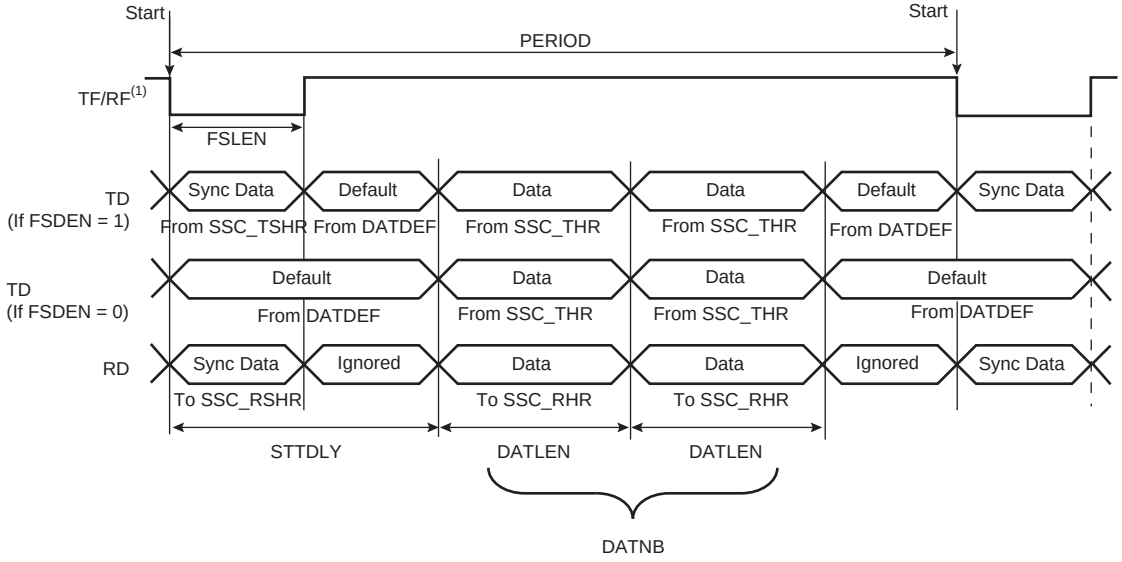
- Length of the data (DATLEN)
- Number of data to be transferred for each start event (DATNB)
- Length of synchronization transferred for each start event (FSLEN)
- Bit sense: most or lowest significant bit first (MSBF)

Additionally, the transmitter can be used to transfer synchronization and select the level driven on the TD pin while not in data transfer operation. This is done respectively by the Frame Sync Data Enable (FSDEN) and by the Data Default Value (DATDEF) bits in SSC_TFMR.

Table 42-4: Data Frame Registers

Transmitter	Receiver	Field	Length	Comment
SSC_TFMR	SSC_RFMR	DATLEN	Up to 32	Size of word
SSC_TFMR	SSC_RFMR	DATNB	Up to 16	Number of words transmitted in frame
SSC_TFMR	SSC_RFMR	MSBF	–	Most significant bit first
SSC_TFMR	SSC_RFMR	FSLEN	Up to 256	Size of Synchro data register
SSC_TFMR	–	DATDEF	0 or 1	Data default value ended
SSC_TFMR	–	FSDEN	–	Enable send SSC_TSHR
SSC_TCMR	SSC_RCMR	PERIOD	Up to 512	Frame size
SSC_TCMR	SSC_RCMR	STTDLY	Up to 255	Size of transmit start delay

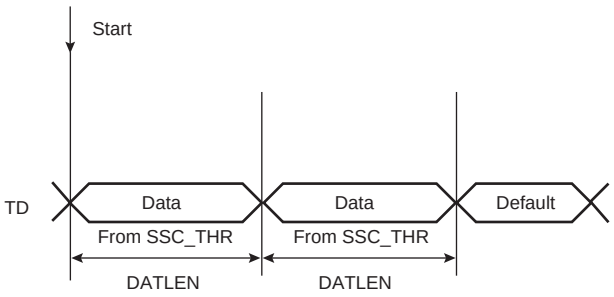
Figure 42-16: Transmit and Receive Frame Format in Edge/Pulse Start Modes



Note: 1. Example of input on falling edge of TF/RF.

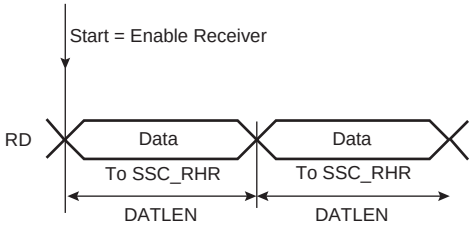
In the example illustrated in Figure 42-17, the SSC_THR is loaded twice. The FSDEN value has no effect on the transmission. SyncData cannot be output in continuous mode.

Figure 42-17: Transmit Frame Format in Continuous Mode (STTDLY = 0)



Start: 1. TXEMPTY set to 1
2. Write into the SSC_THR

Figure 42-18: Receive Frame Format in Continuous Mode (STTDLY = 0)



42.8.8 Loop Mode

The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in the SSC_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

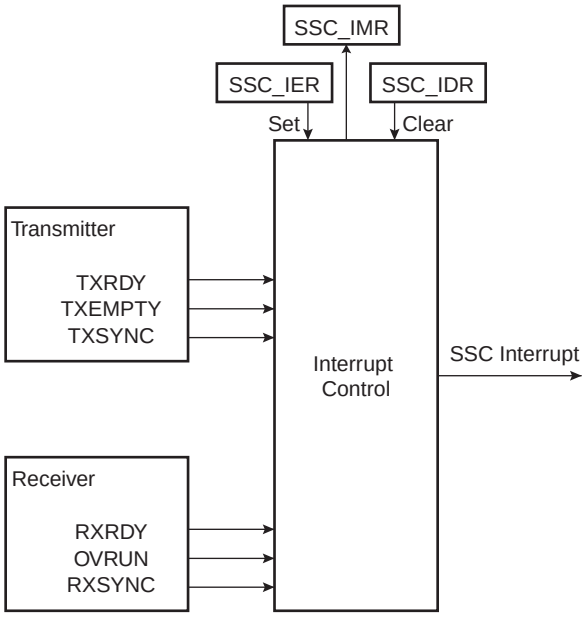
42.8.9 Interrupt

Most bits in the SSC_SR have a corresponding bit in interrupt management registers.

The SSC can be programmed to generate an interrupt when it detects an event. The interrupt is controlled by writing the Interrupt Enable Register (SSC_IER) and Interrupt Disable Register (SSC_IDR). These registers enable and disable, respectively, the corresponding interrupt by setting and clearing the corresponding bit in the Interrupt Mask Register (SSC_IMR), which controls the generation of interrupts by asserting the SSC interrupt line connected to the interrupt controller.

© 2017 Microchip Technology Inc.

Figure 42-19: Interrupt Block Diagram



42.8.10 Register Write Protection

To prevent any single software error from corrupting AIC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the SSC Write Protection Mode Register (SSC_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the SSC Write Protection Status Register (SSC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the SSC_WPSR.

The following registers can be write-protected:

- SSC Clock Mode Register
- SSC Receive Clock Mode Register
- SSC Receive Frame Mode Register
- SSC Transmit Clock Mode Register
- SSC Transmit Frame Mode Register
- SSC Receive Compare 0 Register
- SSC Receive Compare 1 Register

© 2017 Microchip Technology Inc.

42.9 Synchronous Serial Controller (SSC) User Interface

Table 42-5: Register Mapping

Offset	Register	Name	Access	Reset
0x0	Control Register	SSC_CR	Write-only	–
0x4	Clock Mode Register	SSC_CMR	Read/Write	0x0
0x8–0xC	Reserved	–	–	–
0x10	Receive Clock Mode Register	SSC_RCMR	Read/Write	0x0
0x14	Receive Frame Mode Register	SSC_RFMR	Read/Write	0x0
0x18	Transmit Clock Mode Register	SSC_TCMR	Read/Write	0x0
0x1C	Transmit Frame Mode Register	SSC_TFMR	Read/Write	0x0
0x20	Receive Holding Register	SSC_RHR	Read-only	0x0
0x24	Transmit Holding Register	SSC_THR	Write-only	–
0x28–0x2C	Reserved	–	–	–
0x30	Receive Sync. Holding Register	SSC_RSHR	Read-only	0x0
0x34	Transmit Sync. Holding Register	SSC_TSHR	Read/Write	0x0
0x38	Receive Compare 0 Register	SSC_RC0R	Read/Write	0x0
0x3C	Receive Compare 1 Register	SSC_RC1R	Read/Write	0x0
0x40	Status Register	SSC_SR	Read-only	0x000000CC
0x44	Interrupt Enable Register	SSC_IER	Write-only	–
0x48	Interrupt Disable Register	SSC_IDR	Write-only	–
0x4C	Interrupt Mask Register	SSC_IMR	Read-only	0x0
0x50–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	SSC_WPMR	Read/Write	0x0
0xE8	Write Protection Status Register	SSC_WPSR	Read-only	0x0
0xEC–0xFC	Reserved	–	–	–
0x100–0x124	Reserved	–	–	–

© 2017 Microchip Technology Inc.

42.9.1 SSC Control Register

Name:SSC_CR

Address:0xF0010000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
SWRST	–	–	–	–	–	TXDIS	TXEN
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RXDIS	RXEN

RXEN: Receive Enable

0: No effect.

1: Enables Receive if RXDIS is not set.

RXDIS: Receive Disable

0: No effect.

1: Disables Receive. If a character is currently being received, disables at end of current character reception.

TXEN: Transmit Enable

0: No effect.

1: Enables Transmit if TXDIS is not set.

TXDIS: Transmit Disable

0: No effect.

1: Disables Transmit. If a character is currently being transmitted, disables at end of current character transmission.

SWRST: Software Reset

0: No effect.

1: Performs a software reset. Has priority on any other bit in SSC_CR.

42.9.2 SSC Clock Mode Register

Name:SSC_CMR

Address:0xF0010004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	DIV			

7	6	5	4	3	2	1	0
DIV							

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

DIV: Clock Divider

0: The Clock Divider is not active.

Any other value: The divided clock equals the peripheral clock divided by 2 times DIV.
The maximum bit rate is $f_{peripheral\ clock}/2$. The minimum bit rate is $f_{peripheral\ clock}/2 \times 4095 = f_{peripheral\ clock}/8190$.

© 2017 Microchip Technology Inc.

42.9.3 SSC Receive Clock Mode Register

Name:SSC_RCMR

Address:0xF0010010

Access: Read/Write

31	30	29	28	27	26	25	24
PERIOD							
23	22	21	20	19	18	17	16
STTDLY							
15	14	13	12	11	10	9	8
–	–	–	STOP	START			
7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

CKS: Receive Clock Selection

Value	Name	Description
0	MCK	Divided Clock
1	TK	TK Clock signal
2	RK	RK pin

CKO: Receive Clock Output Mode Selection

Value	Name	Description
0	NONE	None, RK pin is an input
1	CONTINUOUS	Continuous Receive Clock, RK pin is an output
2	TRANSFER	Receive Clock only during data transfers, RK pin is an output

CKI: Receive Clock Inversion

0: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock falling edge. The Frame Sync signal output is shifted out on Receive Clock rising edge.

1: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock rising edge. The Frame Sync signal output is shifted out on Receive Clock falling edge.

CKI affects only the Receive Clock and not the output clock signal.

CKG: Receive Clock Gating Selection

Value	Name	Description
0	CONTINUOUS	None
1	EN_RF_LOW	Receive Clock enabled only if RF Low
2	EN_RF_HIGH	Receive Clock enabled only if RF High

© 2017 Microchip Technology Inc.

START: Receive Start Selection

Value	Name	Description
0	CONTINUOUS	Continuous, as soon as the receiver is enabled, and immediately after the end of transfer of the previous data.
1	TRANSMIT	Transmit start
2	RF_LOW	Detection of a low level on RF signal
3	RF_HIGH	Detection of a high level on RF signal
4	RF_FALLING	Detection of a falling edge on RF signal
5	RF_RISING	Detection of a rising edge on RF signal
6	RF_LEVEL	Detection of any level change on RF signal
7	RF_EDGE	Detection of any edge on RF signal
8	CMP_0	Compare 0

STOP: Receive Stop Selection

0: After completion of a data transfer when starting with a Compare 0, the receiver stops the data transfer and waits for a new compare 0.

1: After starting a receive with a Compare 0, the receiver operates in a continuous mode until a Compare 1 is detected.

STTDLY: Receive Start Delay

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of reception. When the Receiver is programmed to start synchronously with the Transmitter, the delay is also applied.

Note: It is very important that STTDLY be set carefully. If STTDLY must be set, it should be done in relation to TAG (Receive Sync Data) reception.

PERIOD: Receive Period Divider Selection

This field selects the divider to apply to the selected Receive Clock in order to generate a new Frame Sync Signal. If 0, no PERIOD signal is generated. If not 0, a PERIOD signal is generated each 2 x (PERIOD + 1) Receive Clock.

© 2017 Microchip Technology Inc.

42.9.4 SSC Receive Frame Mode Register

Name:SSC_RFMR

Address:0xF0010014

Access: Read/Write

31	30	29	28	27	26	25	24
FSLEN_EXT				–	–	–	FSEDGE
23	22	21	20	19	18	17	16
–	FSOS			FSLEN			
15	14	13	12	11	10	9	8
–	–	–	–	DATNB			
7	6	5	4	3	2	1	0
MSBF	–	LOOP	DATLEN				

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

DATLEN: Data Length

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits.

LOOP: Loop Mode

0: Normal operating mode.

1: RD is driven by TD, RF is driven by TF and TK drives RK.

MSBF: Most Significant Bit First

0: The lowest significant bit of the data register is sampled first in the bit stream.

1: The most significant bit of the data register is sampled first in the bit stream.

DATNB: Data Number per Frame

This field defines the number of data words to be received after each transfer start, which is equal to (DATNB + 1).

FSLEN: Receive Frame Sync Length

This field defines the number of bits sampled and stored in the Receive Sync Data Register. When this mode is selected by the START field in the Receive Clock Mode Register, it also determines the length of the sampled data to be compared to the Compare 0 or Compare 1 register.

This field is used with FSLEN_EXT to determine the pulse length of the Receive Frame Sync signal.

Pulse length is equal to FSLEN + (FSLEN_EXT × 16) + 1 Receive Clock periods.

FSOS: Receive Frame Sync Output Selection

Value	Name	Description
0	NONE	None, RF pin is an input
1	NEGATIVE	Negative Pulse, RF pin is an output
2	POSITIVE	Positive Pulse, RF pin is an output
3	LOW	Driven Low during data transfer, RF pin is an output
4	HIGH	Driven High during data transfer, RF pin is an output
5	TOGGLING	Toggling at each start of data transfer, RF pin is an output

© 2017 Microchip Technology Inc.

FSEDGE: Frame Sync Edge Detection

Determines which edge on Frame Sync will generate the interrupt RXSYN in the SSC Status Register.

Value	Name	Description
0	POSITIVE	Positive Edge Detection
1	NEGATIVE	Negative Edge Detection

FSLEN_EXT: FSLEN Field Extension

Extends FSLEN field. For details, refer to FSLEN bit description above.

© 2017 Microchip Technology Inc.

42.9.5 SSC Transmit Clock Mode Register

Name:SSC_TCMR

Address:0xF0010018

Access: Read/Write

31	30	29	28	27	26	25	24
PERIOD							
23	22	21	20	19	18	17	16
STTDLY							
15	14	13	12	11	10	9	8
–	–	–	–	START			
7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

CKS: Transmit Clock Selection

Value	Name	Description
0	MCK	Divided Clock
1	RK	RK Clock signal
2	TK	TK pin

CKO: Transmit Clock Output Mode Selection

Value	Name	Description
0	NONE	None, TK pin is an input
1	CONTINUOUS	Continuous Transmit Clock, TK pin is an output
2	TRANSFER	Transmit Clock only during data transfers, TK pin is an output

CKI: Transmit Clock Inversion

0: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock falling edge. The Frame sync signal input is sampled on Transmit clock rising edge.

1: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock rising edge. The Frame sync signal input is sampled on Transmit clock falling edge.

CKI affects only the Transmit Clock and not the output clock signal.

CKG: Transmit Clock Gating Selection

Value	Name	Description
0	CONTINUOUS	None
1	EN_TF_LOW	Transmit Clock enabled only if TF Low
2	EN_TF_HIGH	Transmit Clock enabled only if TF High

© 2017 Microchip Technology Inc.

START: Transmit Start Selection

Value	Name	Description
0	CONTINUOUS	Continuous, as soon as a word is written in the SSC_THR (if Transmit is enabled), and immediately after the end of transfer of the previous data
1	RECEIVE	Receive start
2	TF_LOW	Detection of a low level on TF signal
3	TF_HIGH	Detection of a high level on TF signal
4	TF_FALLING	Detection of a falling edge on TF signal
5	TF_RISING	Detection of a rising edge on TF signal
6	TF_LEVEL	Detection of any level change on TF signal
7	TF_EDGE	Detection of any edge on TF signal

STTDLY: Transmit Start Delay

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of transmission of data. When the Transmitter is programmed to start synchronously with the Receiver, the delay is also applied.

Note: Note: STTDLY must be set carefully. If STTDLY is too short in respect to TAG (Transmit Sync Data) emission, data is emitted instead of the end of TAG.

PERIOD: Transmit Period Divider Selection

This field selects the divider to apply to the selected Transmit Clock to generate a new Frame Sync Signal. If 0, no period signal is generated. If not 0, a period signal is generated at each $2 \times (PERIOD + 1)$ Transmit Clock.

© 2017 Microchip Technology Inc.

42.9.6 SSC Transmit Frame Mode Register

Name:SSC_TFMR

Address:0xF001001C

Access: Read/Write

31	30	29	28	27	26	25	24
FSLEN_EXT				–	–	–	FSEDGE

23	22	21	20	19	18	17	16
FSDEN	FSOS			FSLEN			

15	14	13	12	11	10	9	8
–	–	–	–	DATNB			

7	6	5	4	3	2	1	0
MSBF	–	DATDEF	DATLEN				

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

DATLEN: Data Length

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits. .

DATDEF: Data Default Value

This bit defines the level driven on the TD pin while out of transmission. Note that if the pin is defined as multi-drive by the PIO Controller, the pin is enabled only if the SCC TD output is 1.

MSBF: Most Significant Bit First

0: The lowest significant bit of the data register is shifted out first in the bit stream.

1: The most significant bit of the data register is shifted out first in the bit stream.

DATNB: Data Number per Frame

This field defines the number of data words to be transferred after each transfer start, which is equal to (DATNB + 1).

FSLEN: Transmit Frame Sync Length

This field defines the length of the Transmit Frame Sync signal and the number of bits shifted out from the Transmit Sync Data Register if FSDEN is 1.

This field is used with FSLEN_EXT to determine the pulse length of the Transmit Frame Sync signal.

Pulse length is equal to FSLEN + (FSLEN_EXT × 16) + 1 Transmit Clock period.

FSOS: Transmit Frame Sync Output Selection

Value	Name	Description
0	NONE	None, TF pin is an input
1	NEGATIVE	Negative Pulse, TF pin is an output
2	POSITIVE	Positive Pulse, TF pin is an output
3	LOW	Driven Low during data transfer
4	HIGH	Driven High during data transfer
5	TOGGLING	Toggling at each start of data transfer

© 2017 Microchip Technology Inc.

FSDEN: Frame Sync Data Enable

0: The TD line is driven with the default value during the Transmit Frame Sync signal.

1: SSC_TSHR value is shifted out during the transmission of the Transmit Frame Sync signal.

FSEDGE: Frame Sync Edge Detection

Determines which edge on frame sync will generate the interrupt TXSYN (Status Register).

Value	Name	Description
0	POSITIVE	Positive Edge Detection
1	NEGATIVE	Negative Edge Detection

FSLEN_EXT: FSLEN Field Extension

Extends FSLEN field. For details, refer to FSLEN bit description above.

© 2017 Microchip Technology Inc.

42.9.7 SSC Receive Holding Register

Name:SSC_RHR

Address:0xF0010020

Access: Read-only

31	30	29	28	27	26	25	24
RDAT							

23	22	21	20	19	18	17	16
RDAT							

15	14	13	12	11	10	9	8
RDAT							

7	6	5	4	3	2	1	0
RDAT							

RDAT: Receive Data

Right aligned regardless of the number of data bits defined by DATLEN in SSC_RFMR.

42.9.8 SSC Transmit Holding Register

Name:SSC_THR

Address:0xF0010024

Access: Write-only

31	30	29	28	27	26	25	24
TDAT							

23	22	21	20	19	18	17	16
TDAT							

15	14	13	12	11	10	9	8
TDAT							

7	6	5	4	3	2	1	0
TDAT							

TDAT: Transmit Data

Right aligned regardless of the number of data bits defined by DATLEN in SSC_TFMR.

© 2017 Microchip Technology Inc.

42.9.9 SSC Receive Synchronization Holding Register

Name:SSC_RSHR

Address:0xF0010030

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RSDAT							

7	6	5	4	3	2	1	0
RSDAT							

RSDAT: Receive Synchronization Data

42.9.10 SSC Transmit Synchronization Holding Register

Name:SSC_TSHR

Address:0xF0010034

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TSDAT							

7	6	5	4	3	2	1	0
TSDAT							

TSDAT: Transmit Synchronization Data

© 2017 Microchip Technology Inc.

42.9.11 SSC Receive Compare 0 Register

Name:SSC_RC0R

Address:0xF0010038

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CP0							

7	6	5	4	3	2	1	0
CP0							

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

CP0: Receive Compare Data 0

42.9.12 SSC Receive Compare 1 Register

Name:SSC_RC1R

Address:0xF001003C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CP1							
7	6	5	4	3	2	1	0
CP1							

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

CP1: Receive Compare Data 1

© 2017 Microchip Technology Inc.

42.9.13 SSC Status Register

Name:SSC_SR

Address:0xF0010040

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXEN	TXEN
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

TXRDY: Transmit Ready

0: Data has been loaded in SSC_THR and is waiting to be loaded in the transmit shift register (TSR).

1: SSC_THR is empty.

TXEMPTY: Transmit Empty

0: Data remains in SSC_THR or is currently transmitted from TSR.

1: Last data written in SSC_THR has been loaded in TSR and last data loaded in TSR has been transmitted.

RXRDY: Receive Ready

0: SSC_RHR is empty.

1: Data has been received and loaded in SSC_RHR.

OVRUN: Receive Overrun

0: No data has been loaded in SSC_RHR while previous data has not been read since the last read of the Status Register.

1: Data has been loaded in SSC_RHR while previous data has not yet been read since the last read of the Status Register.

CP0: Compare 0

0: A compare 0 has not occurred since the last read of the Status Register.

1: A compare 0 has occurred since the last read of the Status Register.

CP1: Compare 1

0: A compare 1 has not occurred since the last read of the Status Register.

1: A compare 1 has occurred since the last read of the Status Register.

TXSYN: Transmit Sync

0: A Tx Sync has not occurred since the last read of the Status Register.

1: A Tx Sync has occurred since the last read of the Status Register.

RXSYN: Receive Sync

0: An Rx Sync has not occurred since the last read of the Status Register.

1: An Rx Sync has occurred since the last read of the Status Register.

TXEN: Transmit Enable

0: Transmit is disabled.

1: Transmit is enabled.

RXEN: Receive Enable

0: Receive is disabled.

1: Receive is enabled.

© 2017 Microchip Technology Inc.

42.9.14 SSC Interrupt Enable Register

Name:SSC_IER

Address:0xF0010044

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

TXRDY: Transmit Ready Interrupt Enable

0: No effect.

1: Enables the Transmit Ready Interrupt.

TXEMPTY: Transmit Empty Interrupt Enable

0: No effect.

1: Enables the Transmit Empty Interrupt.

RXRDY: Receive Ready Interrupt Enable

0: No effect.

1: Enables the Receive Ready Interrupt.

OVRUN: Receive Overrun Interrupt Enable

0: No effect.

1: Enables the Receive Overrun Interrupt.

CP0: Compare 0 Interrupt Enable

0: No effect.

1: Enables the Compare 0 Interrupt.

CP1: Compare 1 Interrupt Enable

0: No effect.

1: Enables the Compare 1 Interrupt.

TXSYN: Tx Sync Interrupt Enable

0: No effect.

1: Enables the Tx Sync Interrupt.

RXSYN: Rx Sync Interrupt Enable

0: No effect.

1: Enables the Rx Sync Interrupt.

42.9.15 SSC Interrupt Disable Register

Name:SSC_IDR

Address:0xF0010048

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

TXRDY: Transmit Ready Interrupt Disable

0: No effect.

1: Disables the Transmit Ready Interrupt.

TXEMPTY: Transmit Empty Interrupt Disable

0: No effect.

1: Disables the Transmit Empty Interrupt.

RXRDY: Receive Ready Interrupt Disable

0: No effect.

1: Disables the Receive Ready Interrupt.

OVRUN: Receive Overrun Interrupt Disable

0: No effect.

1: Disables the Receive Overrun Interrupt.

CP0: Compare 0 Interrupt Disable

0: No effect.

1: Disables the Compare 0 Interrupt.

CP1: Compare 1 Interrupt Disable

0: No effect.

1: Disables the Compare 1 Interrupt.

TXSYN: Tx Sync Interrupt Enable

0: No effect.

1: Disables the Tx Sync Interrupt.

RXSYN: Rx Sync Interrupt Enable

0: No effect.

1: Disables the Rx Sync Interrupt.

© 2017 Microchip Technology Inc.

42.9.16 SSC Interrupt Mask Register

Name:SSC_IMR

Address:0xF001004C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

TXRDY: Transmit Ready Interrupt Mask

0: The Transmit Ready Interrupt is disabled.

1: The Transmit Ready Interrupt is enabled.

TXEMPTY: Transmit Empty Interrupt Mask

0: The Transmit Empty Interrupt is disabled.

1: The Transmit Empty Interrupt is enabled.

RXRDY: Receive Ready Interrupt Mask

0: The Receive Ready Interrupt is disabled.

1: The Receive Ready Interrupt is enabled.

OVRUN: Receive Overrun Interrupt Mask

0: The Receive Overrun Interrupt is disabled.

1: The Receive Overrun Interrupt is enabled.

CP0: Compare 0 Interrupt Mask

0: The Compare 0 Interrupt is disabled.

1: The Compare 0 Interrupt is enabled.

CP1: Compare 1 Interrupt Mask

0: The Compare 1 Interrupt is disabled.

1: The Compare 1 Interrupt is enabled.

TXSYN: Tx Sync Interrupt Mask

0: The Tx Sync Interrupt is disabled.

1: The Tx Sync Interrupt is enabled.

RXSYN: Rx Sync Interrupt Mask

0: The Rx Sync Interrupt is disabled.

1: The Rx Sync Interrupt is enabled.

42.9.17 SSC Write Protection Mode Register

Name:SSC_WPMR

Address:0xF00100E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

WPEN: Write Protection Enable

0: Disables the write protection if WPKEY corresponds to 0x535343 (“SSC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x535343 (“SSC” in ASCII).

See Section 42.8.10 “Register Write Protection” for the list of registers that can be protected.

WPKEY: Write Protection Key

Value	Name	Description
0x535343	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

© 2017 Microchip Technology Inc.

42.9.18 SSC Write Protection Status Register

Name:SSC_WPSR

Address:0xF00100E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

WPVS: Write Protection Violation Status

0: No write protection violation has occurred since the last read of the SSC_WPSR.

1: A write protection violation has occurred since the last read of the SSC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

WPVSRC: Write Protect Violation Source

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

43. LCD Controller (LCDC)

43.1 Description

The LCD Controller (LCDC) consists of logic for transferring LCD image data from an external display buffer to an LCD module. The LCDC has one display input buffer that fetches pixels through the AB master interface and a lookup table to allow palletized display configurations. The LCDC is programmable on a per overlay basis, and supports different LCD resolution, window size, image format and pixel depth.

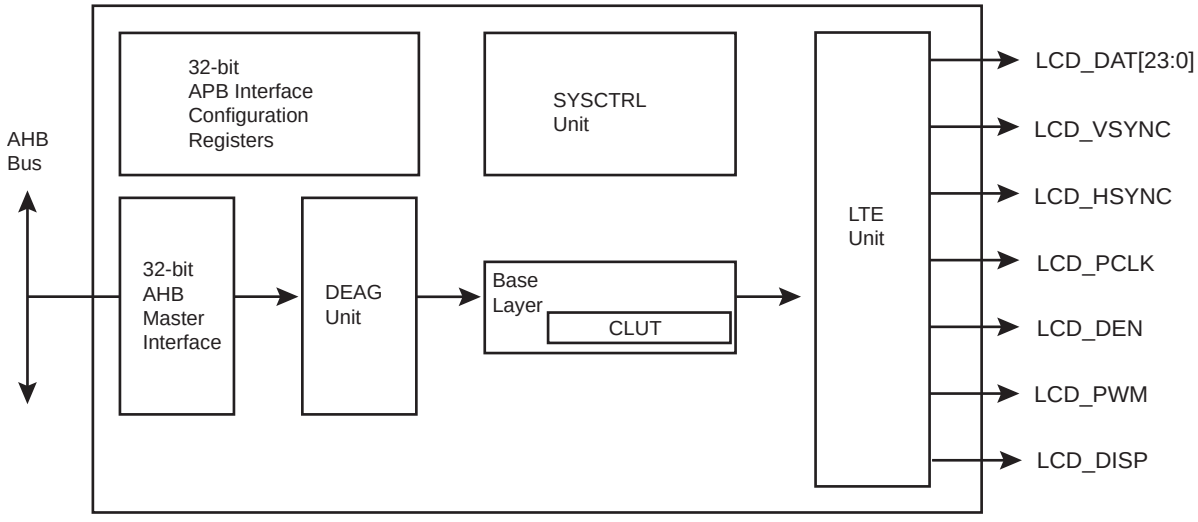
The LCDC is connected to the Arm Advanced High Performance Bus (AHB) as a master for reading pixel data. It also integrates an APB interface to configure its registers.

43.2 Embedded Characteristics

- One AHB Master Interface
- Supports Single Scan Active TFT Display
- Supports 12-, 16-, 18- and 24-bit Output Mode through the Spatial Dithering Unit
- Asynchronous Output Mode Supported
- 1, 2, 4, 8 bits per pixel (palletized)
- 12, 16, 18, 19, 24, 25 and 32 bits per pixel (non-palletized)
- Supports One Base Layer (background)
- Little Endian Memory Organization
- Programmable Timing Engine, with Integer Clock Divider
- Programmable Polarity for Data, Line Synchro and Frame Synchro
- Display Size up to 1280 × 860
- Color Lookup Table with up to 256 entries
- Programmable Negative and Positive Row Striding
- DMA User interface uses Linked List Structure and Add-to-queue Structure

43.3 Block Diagram

Figure 43-1: Block Diagram



DEAG: DMA Engine Address Generation
LTE: LCD Timing Engine

© 2017 Microchip Technology Inc.

43.4 I/O Lines Description

Table 43-1: I/O Lines Description

Name	Description	Type
LCD_PWM	Contrast control signal, using Pulse Width Modulation	Output
LCD_HSYNC	Horizontal Synchronization Pulse	Output
LCD_VSYNC	Vertical Synchronization Pulse	Output
LCD_DAT[23:0]	LCD 24-bit data bus	Output
LCD_DEN	Data Enable	Output
LCD_DISP	Display Enable signal	Output
LCD_PCLK	Pixel Clock	Output

43.5 Product Dependencies

43.5.1 I/O Lines

The pins used for interfacing the LCDC may be multiplexed with PIO lines. The programmer must first program the PIO Controller to assign the pins to their peripheral function. If I/O lines of the LCDC are not used by the application, they can be used for other purposes by the PIO Controller.

Table 43-2: I/O Lines

Instance	Signal	I/O Line	Peripheral
LCDC	LCDDAT0	PC0	A
LCDC	LCDDAT1	PC1	A
LCDC	LCDDAT2	PC2	A
LCDC	LCDDAT3	PC3	A
LCDC	LCDDAT4	PC4	A
LCDC	LCDDAT5	PC5	A
LCDC	LCDDAT6	PC6	A
LCDC	LCDDAT7	PC7	A
LCDC	LCDDAT8	PC8	A
LCDC	LCDDAT9	PC9	A
LCDC	LCDDAT10	PC10	A
LCDC	LCDDAT11	PC11	A
LCDC	LCDDAT12	PC12	A
LCDC	LCDDAT13	PC13	A
LCDC	LCDDAT14	PC14	A
LCDC	LCDDAT15	PC15	A
LCDC	LCDDAT16	PC16	A
LCDC	LCDDAT17	PC17	A
LCDC	LCDDAT18	PC18	A
LCDC	LCDDAT19	PC19	A
LCDC	LCDDAT20	PC20	A

Table 43-2: I/O Lines (Continued)

Instance	Signal	I/O Line	Peripheral
LCDC	LCDDAT21	PC21	A
LCDC	LCDDAT22	PC22	A
LCDC	LCDDAT23	PC23	A
LCDC	LCDDEN	PC29	A
LCDC	LCDDISP	PC24	A
LCDC	LCDHSYNC	PC28	A
LCDC	LCDPCK	PC30	A
LCDC	LCDPWM	PC26	A
LCDC	LCDVSYNC	PC27	A

43.5.2 Power Management

The LCDC is not continuously clocked. The user must first enable the LCDC clock in the Power Management Controller before using it (PMC_PCER).

43.5.3 Interrupt Sources

The LCDC interrupt line is connected to one of the internal sources of the Advanced Interrupt Controller. Using the LCDC interrupt requires prior programming of the AIC.

Table 43-3: Peripheral IDs

Instance	ID
LCDC	25

43.6 Functional Description

The LCD module integrates the following digital blocks:

- DMA Engine Address Generation (DEAG)—This block performs data prefetch and requests access to the AHB interface.
- Input FIFO stores the stream of pixels.
- Color Lookup Table (CLUT)—These 256 RAM-based lookup table entries are selected when the color depth is set to 1, 2, 4 or 8 bpp.
- Output FIFO—stores the pixel prior to display.
- LCD Timing Engine—provides a fully programmable HSYNC-VSYNC interface.

The DMA controller reads the image through the AHB master interface. The LCDC engine formats the display data and writes the final pixel into the output FIFO. The programmable timing engine drives a valid pixel onto the LCD_DAT[23:0] display bus.

43.6.1 Timing Engine Configuration

43.6.1.1 Pixel Clock Period Configuration

The pixel clock (PCLK) generated by the timing engine is the source clock (SCLK) divided by the field CLKDIV in the LCDC_LCDCFG0 register. The source clock can be selected between the system clock and the 2x system clock with the field CLKSEL located in the LCDC_LCDCFG0 register.

Pixel Clock period formula:

$$PCLK = \frac{SCLK}{CLKDIV + 2}$$

The Pixel Clock polarity is also programmable.

© 2017 Microchip Technology Inc.

43.6.1.2 Horizontal and Vertical Synchronization Configuration

The following fields are used to configure the timing engine:

- LCDC_LCDCFG1.HSPW
- LCDC_LCDCFG1.VSPW
- LCDC_LCDCFG2.VFPW
- LCDC_LCDCFG2.VBPW
- LCDC_LCDCFG3.HFPW
- LCDC_LCDCFG3.HBPW
- LCDC_LCDCFG4.PPL
- LCDC_LCDCFG4.RPF

The polarity of output signals is also programmable.

43.6.1.3 Timing Engine Power Up Software Operation

The following sequence is used to enable the display:

1. Configure LCD timing parameters, signal polarity and clock period.
2. Enable the Pixel Clock by writing one to to bit LCDC_LCDEN.CLKEN.
3. Poll bit LCDC_LCDSR.CLKSTS to check that the clock is running.
4. Enable Horizontal and Vertical Synchronization by writing one to bit LCDC_LCDEN.SYNCEN.
5. Poll bit LCDC_LCDSR.LCDSTS to check that the synchronization is up.
6. Enable the display power signal writing one to bit LCDC_LCDEN.DISPEN.
7. Poll bit LCDC_LCDSR.DISPSTS to check that the power signal is activated.

The field LCDC_LCDCFG5.GUARDTIME is used to configure the number of frames before the assertion of the DISP signal.

43.6.1.4 Timing Engine Power Down Software Operation

The following sequence is used to disable the display:

1. Disable the DISP signal writing bit LCDC_LCDDIS.DISPDIS.
2. Poll bit LCDC_LCDSR.DISPSTS to verify that the DISP is no longer activated.
3. Disable the HSYNC and VSYNC signals by writing one to to bit LCDC_LCDDIS.SYNCDIS.
4. Poll bit LCDC_LCDSR.LCDSTS to check that the synchronization is off.
5. Disable the Pixel clock by writing one to bit LCDC_LCDDIS.CLKDIS.

43.6.2 DMA Software Operations

43.6.2.1 DMA Channel Descriptor (DSCR) Alignment and Structure

The DMA Channel Descriptor (DSCR) must be word aligned.

The DMA Channel Descriptor structure contains three fields:

- DSCR.CHXADDR: Frame Buffer base address register
- DSCR.CHXCTRL: Transfer Control register
- DSCR.CHXNEXT: Next Descriptor Address register

Table 43-4: DMA Channel Descriptor Structure

System Memory	Structure Field for channel CHX
DSCR + 0x0	ADDR
DSCR + 0x4	CTRL
DSCR + 0x8	NEXT

43.6.2.2 Programming a DMA Channel

1. Check the status of the channel reading the CHXCHSR register.
2. Write the channel descriptor (DSCR) structure in the system memory by writing DSCR.CHXADDR Frame base address, DSCR.CHXCTRL channel control and DSCR.CHXNEXT next descriptor location.
3. If more than one descriptor is expected, the DFETCH bit of DSCR.CHXCTRL is set to one to enable the descriptor fetch operation.

4. Write the DSCR.CHXNEXT register with the address location of the descriptor structure and set DFETCH bit of the DSCR.CHXCTRL register to one.
5. Enable the relevant channel by writing one to the CHEN bit of the CHXCHER register.
6. An interrupt may be raised if unmasked when the descriptor has been loaded.

43.6.2.3 Disabling a DMA channel

1. Clear the DFETCH bit in the DSCR.CHXCTRL field of the DSCR structure will disable the channel at the end of the frame.
2. Set the DSCR.CHXNEXT register of the DSCR structure will disable the channel at the end of the frame.
3. Writing one to the CHDIS bit of the CHXCHDR register will disable the channel at the end of the frame.
4. Writing one to the CHRST bit of the CHXCHDR register will disable the channel immediately. This may occur in the middle of the image.
5. Poll CHSR bit in the CHXCHSR register until the channel is successfully disabled.

43.6.2.4 DMA Dynamic Linking of a New Transfer Descriptor

1. Write the new descriptor structure in the system memory.
2. Write the address of the new structure in the CHXHEAD register.
3. Add the new structure to the queue of descriptors by writing one to the A2QEN bit of the CHXCHER register.
4. The new descriptor will be added to the queue on the next frame.
5. An interrupt will be raised if unmasked, when the head descriptor structure has been loaded by the DMA channel.

43.6.2.5 DMA Interrupt Generation

The DMA controller operation sets the following interrupt flags in the interrupt status register CHXISR:

- DMA field indicates that the DMA transfer is completed.
- DSCR field indicates that the descriptor structure is loaded in the DMA controller.
- ADD field indicates that a descriptor has been added to the descriptor queue.
- DONE field indicates that the channel transfer has terminated and the channel is automatically disabled.

43.6.2.6 DMA Address Alignment Requirements

When programming the DSCR.CHXADDR register of the DSCR structure the following requirement must be met.

Table 43-5: DMA address alignment when CLUT Mode is selected

CLUT Mode	DMA Address Alignment
1 bpp	8-bit
2 bpp	8-bit
4 bpp	8-bit
8 bpp	8-bit

Table 43-6: DMA address alignment when RGB Mode is selected

RGB Mode	DMA Address Alignment
12 bpp RGB 444	16-bit
16 bpp ARGB 4444	16-bit
16 bpp RGBA 4444	16-bit
16 bpp RGB 565	16-bit
16 bpp TRGB 1555	16-bit
18 bpp RGB 666	32-bit
18 bpp RGB 666 PACKED	8-bit
19 bpp TRGB 1666	32-bit

© 2017 Microchip Technology Inc.

Table 43-6: DMA address alignment when RGB Mode is selected

RGB Mode	DMA Address Alignment
19 bpp TRGB 1666	8-bit
24 bpp RGB 888	32-bit
24 bpp RGB 888 PACKED	8-bit
25 bpp TRGB 1888	32-bit
32 bpp ARGB 8888	32-bit
32 bpp RGBA 8888	32-bit

43.6.3 Display Software Configuration

43.6.3.1 System Bus Access Attributes

These attributes are defined to improve bandwidth of the pixel stream.

- DLBO bit: when set to one only defined burst lengths are performed when the DMA channel retrieves the data from the memory.
- BLEN field: defines the maximum burst length of the DMA channel.

43.6.3.2 Color Attributes

- CLUTMODE field: selects the Color Lookup Table mode
- RGBMODE field: selects the RGB mode

43.6.3.3 Window Attributes Software Operation

1. When required, write the overlay attributes configuration registers.
2. Set UPDATEEN field of the CHXCHER register.
3. Poll UPDATESR field in the CHXCHSR, the update applies when that field is reset.

43.6.4 RGB Frame Buffer Memory Bitmap

43.6.4.1 1 bpp Through Color Lookup Table

Table 43-7: 1 bpp memory mapping, little endian organization

Mem addr	**0x3**								**0x2**								**0x1**								**0x0**							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 1 bpp	p31	p30	p29	p28	p27	p26	p25	p24	p23	p22	p21	p20	p19	p18	p17	p16	p15	p14	p13	p12	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0

43.6.4.2 2 bpp Through Color Lookup Table

Table 43-8: 2 bpp memory mapping, little endian organization

Mem addr	**0x3**								**0x2**								**0x1**								**0x0**							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 2 bpp	p15		p14		p13		p12		p11		p10		p9		p8		p7		p6		p5		p4		p3		p2		p1		p0	

43.6.4.3 4 bpp Through Color Lookup Table

Table 43-9: 4 bpp memory mapping, little endian organization

Mem addr	**0x3**								**0x2**								**0x1**								**0x0**							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 4 bpp	p7				p6				p5				p4				p3				p2				p1				p0			

43.6.4.4 8 bpp Through Color Lookup Table

Table 43-10: 8 bpp memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 8 bpp	p3								p2								p1								p0							

43.6.4.5 12 bpp Memory Mapping, RGB 4:4:4

Table 43-11: 12 bpp memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 12 bpp	–				R1[3:0]				G1[3:0]				B1[3:0]				–				R0[3:0]				G0[3:0]				B0[3:0]			

43.6.4.6 16 bpp Memory Mapping with Alpha Channel, ARGB 4:4:4:4

Table 43-12: 16 bpp memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	A1[3:0]				R1[3:0]				G1[3:0]				B1[3:0]				A0[3:0]				R0[3:0]				G0[3:0]				B0[3:0]			

43.6.4.7 16 bpp Memory Mapping with Alpha Channel, RGBA 4:4:4:4

Table 43-13: 16 bpp memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	R1[3:0]				G13:0]				B1[3:0]				A1[3:0]				R0[3:0]				G0[3:0]				B0[3:0]				A0[3:0]			

43.6.4.8 16 bpp Memory Mapping with Alpha Channel, RGB 5:6:5

Table 43-14: 16 bpp memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16bpp	R1[4:0]					G1[5:0]						B1[4:0]					R0[4:0]					G0[5:0]						B0[4:0]			

43.6.4.9 16 bpp Memory Mapping with Transparency Bit, ARGB 1:5:5:5

Table 43-15: 16 bpp memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 4 bpp	A1	R1[4:0]					G1[4:0]					B1[4:0]					A0	R0[4:0]					G0[4:0]						B0[4:0]			

© 2017 Microchip Technology Inc.

43.6.4.10 18 bpp Unpacked Memory Mapping with Transparency Bit, RGB 6:6:6

Table 43-16: 18 bpp unpacked memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 18 bpp															R0[5:0]						G0[5:0]						B0[5:0]					

43.6.4.11 18 bpp Packed Memory Mapping with Transparency Bit, RGB 6:6:6

Table 43-17: 18 bpp packed memory mapping, little endian organization at address 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 18 bpp	G1[1:0]		B1[5:0]												R0[5:0]						G0[5:0]						B0[5:0]					

Table 43-18: 18 bpp packed memory mapping, little endian organization at address 0x4, 0x5, 0x6, 0x7

Mem addr	0x7								0x6								0x5								0x4							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 18 bpp	R2[3:0]				G2[5:0]				B2[5:0]														R1[5:2]					G1[5:2]			

Table 43-19: 18 bpp packed memory mapping, little endian organization at address 0x8, 0x9, 0xA, 0xB

Mem addr	0xB								0xA								0x9								0x8							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 18 bpp	G4[1:0]		B4[5:0]											R3[5:0]					G3[5:0]					B3[3:0]						R2[5:4]	

43.6.4.12 19 bpp Unpacked Memory Mapping with Transparency Bit, RGB 1:6:6:6

Table 43-20: 19 bpp unpacked memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 19 bpp														A0	R0[5:0]						G0[5:0]						B0[5:0]					

43.6.4.13 19 bpp Packed Memory Mapping with Transparency Bit, ARGB 1:6:6:6

Table 43-21: 19 bpp packed memory mapping, little endian organization at address 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 19 bpp	G1[1:0]		B1[5:0]											A0	R0[5:0]						G0[5:0]						B0[5:0]					

Table 43-22: 19 bpp packed memory mapping, little endian organization at address 0x4, 0x5, 0x6, 0x7

Mem addr	0x7								0x6								0x5								0x4							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 19 bpp	R2[3:0]				G2[5:0]				B2[5:0]													A1	R1[5:2]					G1[5:2]			

Table 43-23: 19 bpp packed memory mapping, little endian organization at address 0x8, 0x9, 0xA, 0xB

Mem addr	0xB								0xA								0x9								0x8							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 19 bpp	G4[1:0]		B4[5:0]									A3	R3[5:0]				G3[5:0]								B3[3:0]						R2[5:4]	

43.6.4.14 24 bpp Unpacked Memory Mapping, RGB 8:8:8

Table 43-24: 24 bpp memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 24 bpp									R0[7:0]								G0[7:0]								B0[7:0]							

43.6.4.15 24 bpp Packed Memory Mapping, RGB 8:8:8

Table 43-25: 24 bpp packed memory mapping, little endian organization at address 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 24 bpp	B1[7:0]								R0[7:0]								G0[7:0]								B0[7:0]							

Table 43-26: 24 bpp packed memory mapping, little endian organization at address 0x4, 0x5, 0x6, 0x7

Mem addr	0x7								0x6								0x5								0x4							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 24 bpp	G2[7:0]								B2[7:0]								R1[7:0]								G1[7:0]							

43.6.4.16 25 bpp Memory Mapping, ARGB 1:8:8:8

Table 43-27: 25 bpp memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 25 bpp								A0	R0[7:0]								G0[7:0]								B0[7:0]							

43.6.4.17 32 bpp Memory Mapping, ARGB 8:8:8:8

Table 43-28: 32 bpp memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 32 bpp	A0[7:0]								R0[7:0]								G0[7:0]								B0[7:0]							

43.6.4.18 32 bpp Memory Mapping, RGBA 8:8:8:8

Table 43-29: 32 bpp memory mapping, little endian organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 32 bpp	R0[7:0]								G0[7:0]								B0[7:0]								A0[7:0]							

© 2017 Microchip Technology Inc.

43.6.5 Output Timing Generation

43.6.5.1 Active Display Timing Mode

Figure 43-2: Active Display Timing

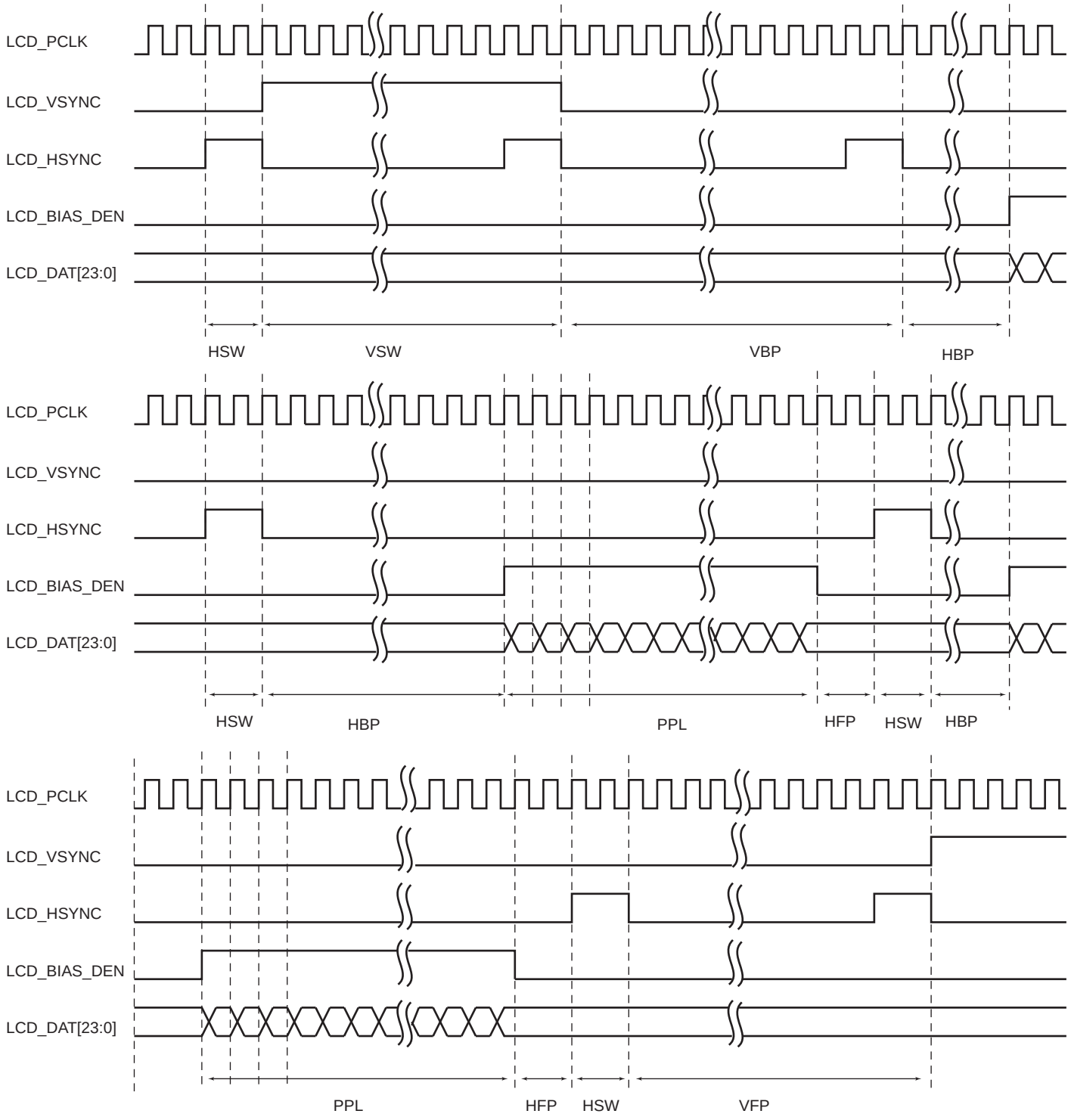
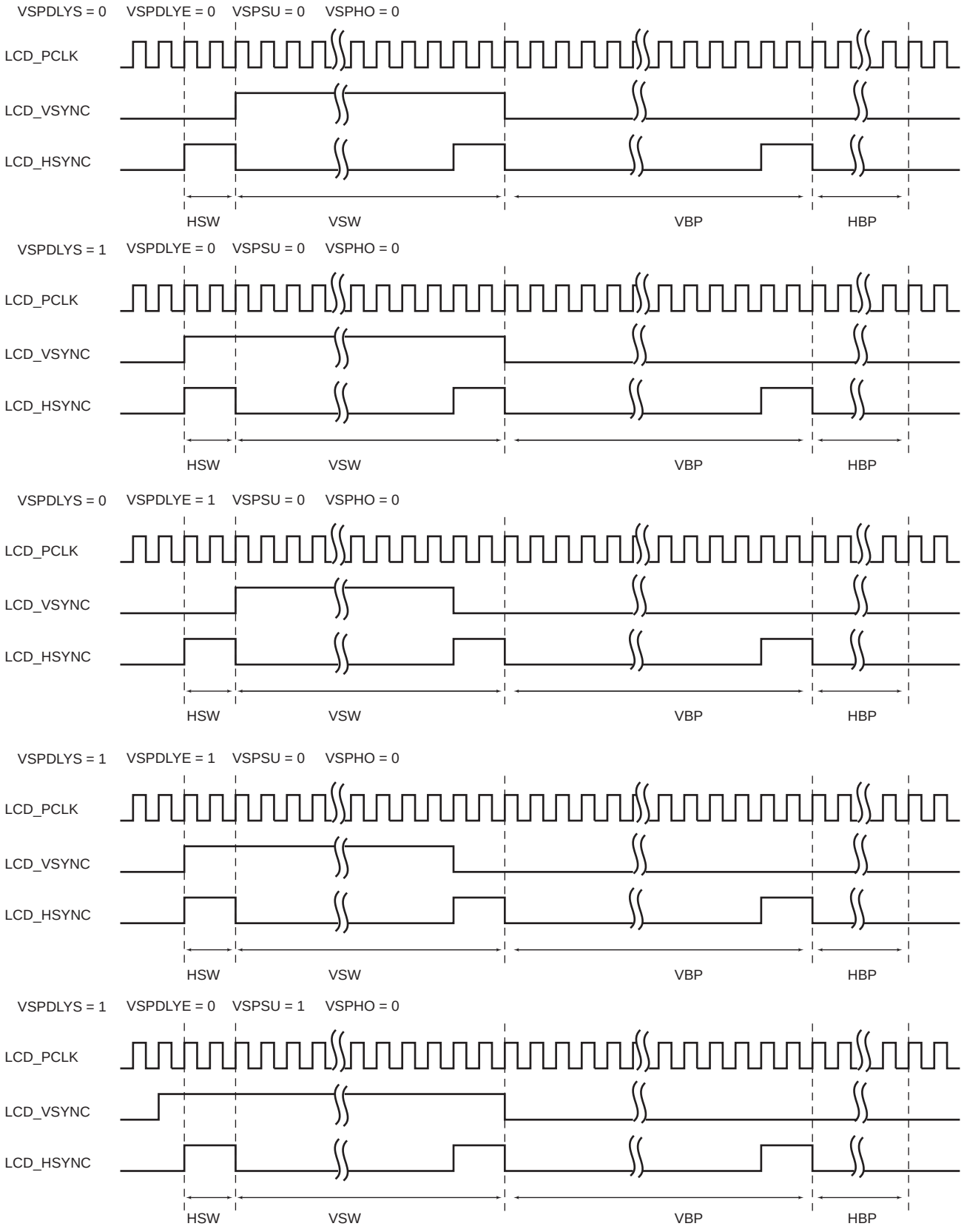


Figure 43-3: Vertical Synchronization Timing (part 1 of 2)



© 2017 Microchip Technology Inc.

Figure 43-4: Vertical Synchronization Timing (part 2 of 2)

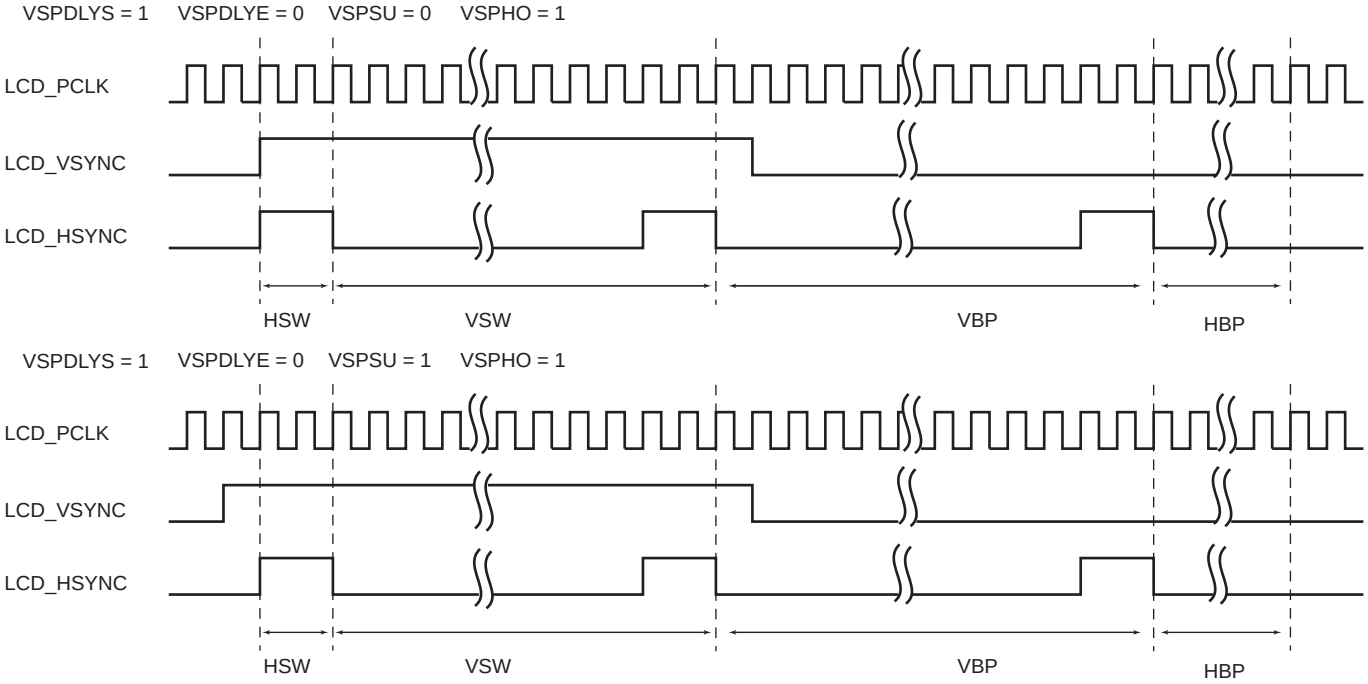
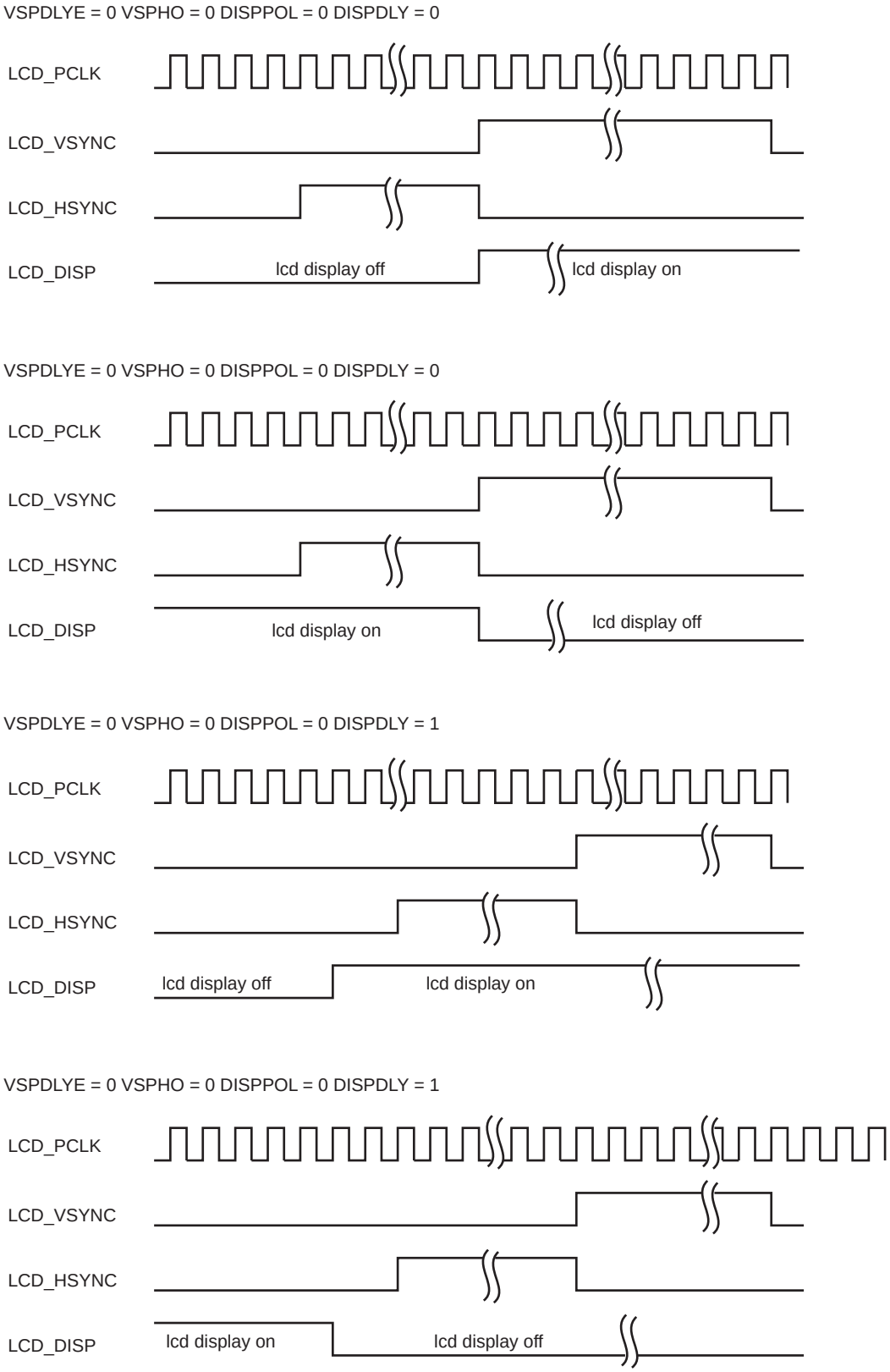


Figure 43-5: DISP Signal Timing Diagram



© 2017 Microchip Technology Inc.

43.6.6 Output Format

43.6.6.1 Active Mode Output Pin Assignment

Table 43-30: Active Mode Output with 24-bit Bus Interface Configuration

Pin ID	TFT 24-bit	TFT 18-bit	TFT 16-bit	TFT 12-bit
LCD_DAT[23]	R[7]	–	–	–
LCD_DAT[22]	R[6]	–	–	–
LCD_DAT[21]	R[5]	–	–	–
LCD_DAT[20]	R[4]	–	–	–
LCD_DAT[19]	R[3]	–	–	–
LCD_DAT[18]	R[2]	–	–	–
LCD_DAT[17]	R[1]	R[5]	–	–
LCD_DAT[16]	R[0]	R[4]	–	–
LCD_DAT[15]	G[7]	R[3]	R[4]	–
LCD_DAT[14]	G[6]	R[2]	R[3]	–
LCD_DAT[13]	G[5]	R[1]	R[2]	–
LCD_DAT[12]	G[4]	R[0]	R[1]	–
LCD_DAT[11]	G[3]	G[5]	R[0]	R[3]
LCD_DAT[10]	G[2]	G[4]	G[5]	R[2]
LCD_DAT[9]	G[1]	G[3]	G[4]	R[1]
LCD_DAT[8]	G[0]	G[2]	G[3]	R[0]
LCD_DAT[7]	B[7]	G[1]	G[2]	G[3]
LCD_DAT[6]	B[6]	G[0]	G[1]	G[2]
LCD_DAT[5]	B[5]	B[5]	G[0]	G[1]
LCD_DAT[4]	B[4]	B[4]	B[4]	G[0]
LCD_DAT[3]	B[3]	B[3]	B[3]	B[3]
LCD_DAT[2]	B[2]	B[2]	B[2]	B[2]
LCD_DAT[1]	B[1]	B[1]	B[1]	B[1]
LCD_DAT[0]	B[0]	B[0]	B[0]	B[0]

© 2017 Microchip Technology Inc.

43.7 LCD Controller (LCDC) User Interface

Table 43-31: Register Mapping

Offset	Register	Name	Access	Reset
0x00000000	LCD Controller Configuration Register 0	LCDC_LCDCFG0	Read/Write	0x00000000
0x00000004	LCD Controller Configuration Register 1	LCDC_LCDCFG1	Read/Write	0x00000000
0x00000008	LCD Controller Configuration Register 2	LCDC_LCDCFG2	Read/Write	0x00000000
0x0000000C	LCD Controller Configuration Register 3	LCDC_LCDCFG3	Read/Write	0x00000000
0x00000010	LCD Controller Configuration Register 4	LCDC_LCDCFG4	Read/Write	0x00000000
0x00000014	LCD Controller Configuration Register 5	LCDC_LCDCFG5	Read/Write	0x00000000
0x00000018	LCD Controller Configuration Register 6	LCDC_LCDCFG6	Read/Write	0x00000000
0x0000001C	Reserved	–	–	–
0x00000020	LCD Controller Enable Register	LCDC_LCDEN	Write-only	–
0x00000024	LCD Controller Disable Register	LCDC_LCDDIS	Write-only	–
0x00000028	LCD Controller Status Register	LCDC_LCDSR	Read-only	0x00000000
0x0000002C	LCD Controller Interrupt Enable Register	LCDC_LCDIER	Write-only	-
0x00000030	LCD Controller Interrupt Disable Register	LCDC_LCDIDR	Write-only	-
0x00000034	LCD Controller Interrupt Mask Register	LCDC_LCDIMR	Read-only	0x00000000
0x00000038	LCD Controller Interrupt Status Register	LCDC_LCDISR	Read-only	0x00000000
0x0000003C	Reserved	–	–	–
0x00000040	Base Layer Channel Enable Register	LCDC_BASECHER	Write-only	–
0x00000044	Base Layer Channel Disable Register	LCDC_BASECHDR	Write-only	–
0x00000048	Base Layer Channel Status Register	LCDC_BASECHSR	Read-only	0x00000000
0x0000004C	Base Layer Interrupt Enable Register	LCDC_BASEIER	Write-only	–
0x00000050	Base Layer Interrupt Disabled Register	LCDC_BASEIDR	Write-only	–
0x00000054	Base Layer Interrupt Mask Register	LCDC_BASEIMR	Read-only	0x00000000
0x00000058	Base Layer Interrupt status Register	LCDC_BASEISR	Read-only	0x00000000
0x0000005C	Base Layer DMA Head Register	LCDC_BASEHEAD	Read/Write	0x00000000
0x00000060	Base Layer DMA Address Register	LCDC_BASEADDR	Read/Write	0x00000000
0x00000064	Base Layer DMA Control Register	LCDC_BASECTRL	Read/Write	0x00000000
0x00000068	Base Layer DMA Next Register	LCDC_BASENEXT	Read/Write	0x00000000
0x0000006C	Base Layer Configuration Register 0	LCDC_BASECFG0	Read/Write	0x00000000
0x00000070	Base Layer Configuration Register 1	LCDC_BASECFG1	Read/Write	0x00000000
0x00000074	Base Layer Configuration Register 2	LCDC_BASECFG2	Read/Write	0x00000000
0x00000078	Base Layer Configuration Register 3	LCDC_BASECFG3	Read/Write	0x00000000
0x0000007C	Base Layer Configuration Register 4	LCDC_BASECFG4	Read/Write	0x00000000
0x80–0x3FC	Reserved	–	–	–
0x400	Base CLUT Register 0[(1)]	LCDC_BASECLUT0	Read/Write	0x00000000
...	...	...	...	...

© 2017 Microchip Technology Inc.

Table 43-31: Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x7FC	Base CLUT Register 255(1)	LCDC_BASECLUT255	Read/Write	0x00000000
0x800–0x1FFC	Reserved	–	–	–

Note 1: The CLUT registers are located in RAM.

43.7.1 LCD Controller Configuration Register 0

Name: LCDC_LCDCFG0

Address:0xF8038000

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CLKDIV							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	CGDISBASE
7	6	5	4	3	2	1	0
–	–	–	–	CLKPWMSEL	CLKSEL	–	CLKPOL

CLKPOL: LCD Controller Clock Polarity

0: Data/Control signals are launched on the rising edge of the Pixel Clock.

1: Data/Control signals are launched on the falling edge of the Pixel Clock.

CLKSEL: LCD Controller Clock Source Selection

0: The Asynchronous output stage of the LCD controller is fed by MCK.

1: The Asynchronous output state of the LCD controller is fed by 2x MCK.

CLKPWMSEL: LCD Controller PWM Clock Source Selection

0: The slow clock is selected and feeds the PWM module.

1: The system clock is selected and feeds the PWM module.

CGDISBASE: Clock Gating Disable Control for the Base Layer

0: Automatic Clock Gating is enabled for the Base Layer.

1: Clock is running continuously.

CLKDIV: LCD Controller Clock Divider

8-bit width clock divider for pixel clock LCD_PCLK.

pixel_clock = selected_clock / (CLKDIV + 2)

where selected_clock is equal to system_clock when CLKSEL field is set to 0 and system_clock2x when CLKSEL is set to one.

© 2017 Microchip Technology Inc.

43.7.2 LCD Controller Configuration Register 1

Name: LCDC_LCDCFG1

Address:0xF8038004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	VSPW					
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	HSPW					

HSPW: Horizontal Synchronization Pulse Width

Width of the LCD_HSYNC pulse, given in pixel clock cycles. Width is (HSPW + 1) LCD_PCLK cycles.

VSPW: Vertical Synchronization Pulse Width

Width of the LCD_VSYNC pulse, given in number of lines. Width is (VSPW + 1) lines.

43.7.3 LCD Controller Configuration Register 2

Name: LCDC_LCDCFG2

Address:0xF8038008

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	VBPW					
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	VFPW					

VFPW: Vertical Front Porch Width

This field indicates the number of lines at the end of the Frame. The blanking interval is equal to (VFPW+1) lines.

VBPW: Vertical Back Porch Width

This field indicates the number of lines at the beginning of the Frame. The blanking interval is equal to VBPW lines.

© 2017 Microchip Technology Inc.

43.7.4 LCD Controller Configuration Register 3

Name: LCDC_LCDCFG3

Address:0xF803800C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
HBPW							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
HFPW							

HFPW: Horizontal Front Porch Width

Number of pixel clock cycles inserted at the end of the active line. The interval is equal to (HFPW + 1) LCD_PCLK cycles.

HBPW: Horizontal Back Porch Width

Number of pixel clock cycles inserted at the beginning of the line. The interval is equal to (HBPW + 1) LCD_PCLK cycles.

43.7.5 LCD Controller Configuration Register 4

Name: LCDC_LCDCFG4

Address:0xF8038010

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	RPF		
23	22	21	20	19	18	17	16
RPF							
15	14	13	12	11	10	9	8
–	–	–	–	–	PPL		
7	6	5	4	3	2	1	0
PPL							

RPF: Number of Active Rows Per Frame

Number of active lines in the frame. The frame height is equal to (RPF + 1) lines.

PPL: Number of Pixels Per Line

Number of pixels in the frame. The number of active pixels in the frame is equal to (PPL + 1) pixels.

© 2017 Microchip Technology Inc.

43.7.6 LCD Controller Configuration Register 5

Name: LCDC_LCDCFG5

Address:0xF8038014

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	GUARDTIME				
15	14	13	12	11	10	9	8
–	–	VSPHO	VSPSU	–	–	MODE	
7	6	5	4	3	2	1	0
DISPDLY	DITHER	–	DISPPOL	VSPDLYE	VSPDLYS	VSPOL	HSPOL

HSPOL: Horizontal Synchronization Pulse Polarity

0: Active High

1: Active Low

VSPOL: Vertical Synchronization Pulse Polarity

0: Active High

1: Active Low

VSPDLYS: Vertical Synchronization Pulse Start

0: The first active edge of the Vertical synchronization pulse is synchronous with the second edge of the horizontal pulse.

1: The first active edge of the Vertical synchronization pulse is synchronous with the first edge of the horizontal pulse.

VSPDLYE: Vertical Synchronization Pulse End

0: The second active edge of the Vertical synchronization pulse is synchronous with the second edge of the horizontal pulse.

1: The second active edge of the Vertical synchronization pulse is synchronous with the first edge of the horizontal pulse.

DISPPOL: Display Signal Polarity

0: Active High

1: Active Low

DITHER: LCD Controller Dithering

0: Dithering logical unit is disabled.

1: Dithering logical unit is activated.

DISPDLY: LCD Controller Display Power Signal Synchronization

0: the LCD_DISP signal is asserted synchronously with the second active edge of the horizontal pulse.

1: the LCD_DISP signal is asserted asynchronously with both edges of the horizontal pulse.

MODE: LCD Controller Output Mode

Value	Name	Description
0	OUTPUT_12BPP	LCD output mode is set to 12 bits per pixel
1	OUTPUT_16BPP	LCD output mode is set to 16 bits per pixel
2	OUTPUT_18BPP	LCD output mode is set to 18 bits per pixel
3	OUTPUT_24BPP	LCD output mode is set to 24 bits per pixel

© 2017 Microchip Technology Inc.

VSPSU: LCD Controller Vertical Synchronization Pulse Setup Configuration

0: The vertical synchronization pulse is asserted synchronously with horizontal pulse edge.

1: The vertical synchronization pulse is asserted one pixel clock cycle before the horizontal pulse.

VSPHO: LCD Controller Vertical Synchronization Pulse Hold Configuration

0: The vertical synchronization pulse is asserted synchronously with horizontal pulse edge.

1: The vertical synchronization pulse is held active one pixel clock cycle after the horizontal pulse.

GUARDTIME: LCD DISPLAY Guard Time

Number of frames inserted during start up before LCD_DISP assertion.

Number of frames inserted after LCD_DISP reset.

© 2017 Microchip Technology Inc.

43.7.7 LCD Controller Configuration Register 6

Name: LCDC_LCDCFG6

Address:0xF8038018

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
PWMCVAL							
7	6	5	4	3	2	1	0
–	–	–	PWMPOL	–	PWMPS		

PWMPS: PWM Clock Prescaler

This field selects the configuration of the counter prescaler module.

Value	Name	Description
0	DIV_1	The counter advances at a rate of $f_{COUNTER} = f_{PWM\_SELECTED\_CLOCK}$
1	DIV_2	The counter advances at a rate of $f_{COUNTER} = f_{PWM\_SELECTED\_CLOCK} / 2$
2	DIV_4	The counter advances at a rate of $f_{COUNTER} = f_{PWM\_SELECTED\_CLOCK} / 4$
3	DIV_8	The counter advances at a rate of $f_{COUNTER} = f_{PWM\_SELECTED\_CLOCK} / 8$
4	DIV_16	The counter advances at a rate of $f_{COUNTER} = f_{PWM\_SELECTED\_CLOCK} / 16$
5	DIV_32	The counter advances at a of rate $f_{COUNTER} = f_{PWM\_SELECTED\_CLOCK} / 32$
6	DIV_64	The counter advances at a of rate $f_{COUNTER} = f_{PWM\_SELECTED\_CLOCK} / 64$

PWMPOL: LCD Controller PWM Signal Polarity

This bit defines the polarity of the PWM output signal.

0: Output pulses are low level

1: Output pulses are high level (The output will be high whenever the value in the counter is less than the value CVAL).

PWMCVAL: LCD Controller PWM Compare Value

PWM compare value. Used to adjust the analog value obtained after an external filter to control the contrast of the display.

43.7.8 LCD Controller Enable Register

Name: LCDC_LCDEN

Address:0xF8038020

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	PWMEN	DISPEN	SYNCEN	CLKEN

CLKEN: LCD Controller Pixel Clock Enable

0: No effect

1: Pixel clock logical unit is activated

SYNCEN: LCD Controller Horizontal and Vertical Synchronization Enable

0: No effect

1: Both horizontal and vertical synchronization (LCD_VSYNC and LCD_HSYNC) signals are generated.

DISPEN: LCD Controller DISP Signal Enable

0: No effect

1: LCD_DISP signal is generated

PWMEN: LCD Controller Pulse Width Modulation Enable

0: No effect

1: PWM is enabled

© 2017 Microchip Technology Inc.

43.7.9 LCD Controller Disable Register

Name: LCDC_LCDDIS

Address:0xF8038024

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	PWMRST	DISPRST	SYNCRST	CLKRST
7	6	5	4	3	2	1	0
–	–	–	–	PWMDIS	DISPDIS	SYNCDIS	CLKDIS

CLKDIS: LCD Controller Pixel Clock Disable

0: No effect.

1: Disable the pixel clock.

SYNCDIS: LCD Controller Horizontal and Vertical Synchronization Disable

0: No effect.

1: Disable the synchronization signals after the end of the frame.

DISPDIS: LCD Controller DISP Signal Disable

0: No effect.

1: Disable the DISP signal.

PWMDIS: LCD Controller Pulse Width Modulation Disable

0: No effect.

1: Disable the pulse width modulation signal.

CLKRST: LCD Controller Clock Reset

0: No effect.

1: Reset the pixel clock generator module. The pixel clock duty cycle may be violated.

SYNCRST: LCD Controller Horizontal and Vertical Synchronization Reset

0: No effect.

1: Reset the timing engine. Both Horizontal and vertical pulse width are violated.

DISPRST: LCD Controller DISP Signal Reset

0: No effect.

1: Reset the DISP signal.

PWMRST: LCD Controller PWM Reset

0: No effect.

1: Reset the PWM module, the duty cycle may be violated.

43.7.10 LCD Controller Status Register

Name: LCDC_LCDSR

Address:0xF8038028

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	SIPSTS	PWMSTS	DISPSTS	LCDSTS	CLKSTS

CLKSTS: Clock Status

0: Pixel Clock is disabled.

1: Pixel Clock is running.

LCDSTS: LCD Controller Synchronization status

0: Timing Engine is disabled.

1: Timing Engine is running.

DISPSTS: LCD Controller DISP Signal Status

0: DISP is disabled.

1: DISP signal is activated.

PWMSTS: LCD Controller PWM Signal Status

0: PWM is disabled.

1: PWM signal is activated.

SIPSTS: Synchronization In Progress

0: Clock domain synchronization is terminated.

1: A double domain synchronization is in progress, access to the LCDC_LCDEN and LCDC_LCDDIS registers has no effect.

© 2017 Microchip Technology Inc.

43.7.11 LCD Controller Interrupt Enable Register

Name: LCDC_LCDIER

Address:0xF803802C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	BASEIE
7	6	5	4	3	2	1	0
–	–	–	FIFOERRIE	–	DISPIE	DISIE	SOFIE

SOFIE: Start of Frame Interrupt Enable Register

0: No effect.

1: Enable the interrupt.

DISIE: LCD Disable Interrupt Enable Register

0: No effect.

1: Enable the interrupt.

DISPIE: Power UP/Down Sequence Terminated Interrupt Enable Register

0: No effect.

1: Enable the interrupt.

FIFOERRIE: Output FIFO Error Interrupt Enable Register

0: No effect.

1: Enable the interrupt.

BASEIE: Base Layer Interrupt Enable Register

0: No effect.

1: Enable the interrupt.

43.7.12 LCD Controller Interrupt Disable Register

Name: LCDC_LCDIDR

Address:0xF8038030

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	BASEID
7	6	5	4	3	2	1	0
–	–	–	FIFOERRID	–	DISPID	DISID	SOFID

SOFID: Start of Frame Interrupt Disable Register

0: No effect.

1: Disable the interrupt.

DISID: LCD Disable Interrupt Disable Register

0: No effect.

1: Disable the interrupt.

DISPID: Power UP/Down Sequence Terminated Interrupt Disable Register

0: No effect.

1: Disable the interrupt.

FIFOERRID: Output FIFO Error Interrupt Disable Register

0: No effect.

1: Disable the interrupt.

BASEID: Base Layer Interrupt Disable Register

0: No effect.

1: Disable the interrupt.

© 2017 Microchip Technology Inc.

43.7.13 LCD Controller Interrupt Mask Register

Name: LCDC_LCDIMR

Address:0xF8038034

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	BASEIM
7	6	5	4	3	2	1	0
–	–	–	FIFOERRIM	–	DISPIM	DISIM	SOFIM

SOFIM: Start of Frame Interrupt Mask Register

0: Interrupt source is disabled.

1: Interrupt source is enabled.

DISIM: LCD Disable Interrupt Mask Register

0: Interrupt source is disabled.

1: Interrupt source is enabled.

DISPIM: Power UP/Down Sequence Terminated Interrupt Mask Register

0: Interrupt source is disabled.

1: Interrupt source is enabled.

FIFOERRIM: Output FIFO Error Interrupt Mask Register

0: Interrupt source is disabled.

1: Interrupt source is enabled.

BASEIM: Base Layer Interrupt Mask Register

0: Interrupt source is disabled.

1: Interrupt source is enabled.

43.7.14 LCD Controller Interrupt Status Register

Name: LCDC_LCDISR

Address:0xF8038038

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	BASE
7	6	5	4	3	2	1	0
–	–	–	FIFOERR	–	DISP	DIS	SOF

SOF: Start of Frame Interrupt Status Register

When set to one this flag indicates that a start of frame event has been detected. This flag is reset after a read operation.

DIS: LCD Disable Interrupt Status Register

When set to one this flag indicates that the horizontal and vertical timing generator has been successfully disabled. This flag is reset after a read operation.

DISP: Power-up/Power-down Sequence Terminated Interrupt Status Register

When set to one this flag indicates whether the power-up sequence or power-down sequence has terminated. This flag is reset after a read operation.

FIFOERR: Output FIFO Error

When set to one this flag indicates that an underflow occurs in the output FIFO. This flag is reset after a read operation.

BASE: Base Layer Raw Interrupt Status Register

When set to one this flag indicates that a Base layer interrupt is pending. This flag is reset as soon as the BASEISR register is read.

© 2017 Microchip Technology Inc.

43.7.15 Base Layer Channel Enable Register

Name: LCDC_BASECHER

Address:0xF8038040

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	A2QEN	UPDATEEN	CHEN

CHEN: Channel Enable Register

0: No effect.

1: Enable the DMA channel.

UPDATEEN: Update Overlay Attributes Enable Register

0: No effect.

1: update windows attributes on the next start of frame.

A2QEN: Add Head Pointer Enable Register

Write this field to one to add the head pointer to the descriptor list. This field is reset by hardware as soon as the head register is added to the list.

43.7.16 Base Layer Channel Disable Register

Name: LCDC_BASECHDR

Address:0xF8038044

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	CHRST
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CHDIS

CHDIS: Channel Disable Register

When set to one this field disables the layer at the end of the current frame. The frame is completed.

CHRST: Channel Reset Register

When set to one this field resets the layer immediately. The frame is aborted.

© 2017 Microchip Technology Inc.

43.7.17 Base Layer Channel Status Register

Name: LCDC_BASECHSR

Address:0xF8038048

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	A2QSR	UPDATESR	CHSR

CHSR: Channel Status Register

When set to one this field disables the layer at the end of the current frame.

UPDATESR: Update Overlay Attributes In Progress

When set to one this bit indicates that the overlay attributes will be updated on the next frame.

A2QSR: Add To Queue Pending Register

When set to one this bit indicates that the head pointer is still pending.

43.7.18 Base Layer Interrupt Enable Register

Name: LCDC_BASEIER

Address:0xF803804C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	OVR	DONE	ADD	DSCR	DMA	–	–

DMA: End of DMA Transfer Interrupt Enable Register

0: No effect.

1: Interrupt source is enabled.

DSCR: Descriptor Loaded Interrupt Enable Register

0: No effect.

1: Interrupt source is enabled.

ADD: Head Descriptor Loaded Interrupt Enable Register

0: No effect.

1: Interrupt source is enabled.

DONE: End of List Interrupt Enable Register

0: No effect.

1: Interrupt source is enabled.

OVR: Overflow Interrupt Enable Register

0: No effect.

1: Interrupt source is enabled.

© 2017 Microchip Technology Inc.

43.7.19 Base Layer Interrupt Disable Register

Name: LCDC_BASEIDR

Address:0xF8038050

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	OVR	DONE	ADD	DSCR	DMA	–	–

DMA: End of DMA Transfer Interrupt Disable Register

0: No effect.

1: Interrupt source is disabled.

DSCR: Descriptor Loaded Interrupt Disable Register

0: No effect.

1: Interrupt source is disabled.

ADD: Head Descriptor Loaded Interrupt Disable Register

0: No effect.

1: Interrupt source is disabled.

DONE: End of List Interrupt Disable Register

0: No effect.

1: Interrupt source is disabled.

OVR: Overflow Interrupt Disable Register

0: No effect.

1: Interrupt source is disabled.

43.7.20 Base Layer Interrupt Mask Register

Name: LCDC_BASEIMR

Address:0xF8038054

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	OVR	DONE	ADD	DSCR	DMA	–	–

DMA: End of DMA Transfer Interrupt Mask Register

0: Interrupt source is disabled.

1: Interrupt source is enabled.

DSCR: Descriptor Loaded Interrupt Mask Register

0: Interrupt source is disabled.

1: Interrupt source is enabled.

ADD: Head Descriptor Loaded Interrupt Mask Register

0: Interrupt source is disabled.

1: Interrupt source is enabled.

DONE: End of List Interrupt Mask Register

0: Interrupt source is disabled.

1: Interrupt source is enabled.

OVR: Overflow Interrupt Mask Register

0: Interrupt source is disabled.

1: Interrupt source is enabled.

© 2017 Microchip Technology Inc.

43.7.21 Base Layer Interrupt Status Register

Name: LCDC_BASEISR

Address:0xF8038058

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	OVR	DONE	ADD	DSCR	DMA	–	–

DMA: End of DMA Transfer

When set to one this flag indicates that an End of Transfer has been detected. This flag is reset after a read operation.

DSCR: DMA Descriptor Loaded

When set to one this flag indicates that a descriptor has been loaded successfully. This flag is reset after a read operation.

ADD: Head Descriptor Loaded

When set to one this flag indicates that the descriptor pointed to by the head register has been loaded successfully. This flag is reset after a read operation.

DONE: End of List Detected

When set to one this flag indicates that an End of List condition has occurred. This flag is reset after a read operation.

OVR: Overflow Detected

When set to one this flag indicates that an overflow occurred. This flag is reset after a read operation.

© 2017 Microchip Technology Inc.

43.7.22 Base Layer Head Register

Name: LCDC_BASEHEAD

Address:0xF803805C

Access: Read/Write

31	30	29	28	27	26	25	24
HEAD							
23	22	21	20	19	18	17	16
HEAD							
15	14	13	12	11	10	9	8
HEAD							
7	6	5	4	3	2	1	0
HEAD						–	–

HEAD: DMA Head Pointer

The Head Pointer points to a new descriptor.

© 2017 Microchip Technology Inc.

43.7.23 Base Layer Address Register

Name: LCDC_BASEADDR

Address:0xF8038060

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							
23	22	21	20	19	18	17	16
ADDR							
15	14	13	12	11	10	9	8
ADDR							
7	6	5	4	3	2	1	0
ADDR							

ADDR: DMA Transfer Start Address

Frame buffer base address.

43.7.24 Base Layer Control Register

Name: LCDC_BASECTRL

Address:0xF8038064

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	DONEIEN	ADDIEN	DSCRIEN	DMAIEN	LFETCH	DFETCH

DFETCH: Transfer Descriptor Fetch Enable

0: Transfer Descriptor fetch is disabled.

1: Transfer Descriptor fetch is enabled.

LFETCH: Lookup Table Fetch Enable

0: Lookup Table DMA fetch is disabled.

1: Lookup Table DMA fetch is enabled.

DMAIEN: End of DMA Transfer Interrupt Enable

0: DMA transfer completed interrupt is enabled.

1: DMA transfer completed interrupt is disabled.

DSCRIEN: Descriptor Loaded Interrupt Enable

0: Transfer descriptor loaded interrupt is enabled.

1: Transfer descriptor loaded interrupt is disabled.

ADDIEN: Add Head Descriptor to Queue Interrupt Enable

0: Transfer descriptor added to queue interrupt is enabled.

1: Transfer descriptor added to queue interrupt is disabled.

DONEIEN: End of List Interrupt Enable

0: End of list interrupt is disabled.

1: End of list interrupt is enabled.

© 2017 Microchip Technology Inc.

43.7.25 Base Layer Next Register

Name: LCDC_BASENEXT

Address:0xF8038068

Access: Read/Write

31	30	29	28	27	26	25	24
NEXT							
23	22	21	20	19	18	17	16
NEXT							
15	14	13	12	11	10	9	8
NEXT							
7	6	5	4	3	2	1	0
NEXT							

NEXT: DMA Descriptor Next Address

DMA Descriptor next address, this address must be word aligned.

43.7.26 Base Layer Configuration 0 Register

Name: LCDC_BASECFG0

Address:0xF803806C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	DLBO
7	6	5	4	3	2	1	0
–	–	BLEN		–	–	–	–

BLEN: AHB Burst Length

Value	Name	Description
0	AHB_SINGLE	AHB Access is started as soon as there is enough space in the FIFO to store one 32-bit data. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts can be used. INCR is used for a burst of 2 and 3 beats.
1	AHB_INCR4	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of four 32-bit data. An AHB INCR4 Burst is preferred. SINGLE, INCR and INCR4 bursts can be used. INCR is used for a burst of 2 and 3 beats.
2	AHB_INCR8	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of eight 32-bit data. An AHB INCR8 Burst is preferred. SINGLE, INCR, INCR4 and INCR8 bursts can be used. INCR is used for a burst of 2 and 3 beats.
3	AHB_INCR16	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of sixteen 32-bit data. An AHB INCR16 Burst is preferred. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts can be used. INCR is used for a burst of 2 and 3 beats.

DLBO: Defined Length Burst Only For Channel Bus Transaction.

0: Undefined length INCR burst is used for a burst of 2 and 3 beats.

1: Only Defined Length burst is used (SINGLE, INCR4, INCR8 and INCR16).

© 2017 Microchip Technology Inc.

43.7.27 Base Layer Configuration 1 Register

Name: LCDC_BASECFG1

Address:0xF8038070

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	-	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–				–	–	CLUTMODE	
7	6	5	4	3	2	1	0
RGBMODE				–	–	–	CLUTEN

CLUTEN: Color Lookup Table Mode Enable

0: RGB mode is selected.

1: Color lookup table is selected.

RGBMODE: RGB Mode Input Selection

Value	Name	Description
0	12BPP_RGB_444	12 bpp RGB 444
1	16BPP_ARGB_4444	16 bpp ARGB 4444
2	16BPP_RGBA_4444	16 bpp RGBA 4444
3	16BPP_RGB_565	16 bpp RGB 565
4	16BPP_TRGB_1555	16 bpp TRGB 1555
5	18BPP_RGB_666	18 bpp RGB 666
6	18BPP_RGB_666_PACKED	18 bpp RGB 666 PACKED
7	19BPP_TRGB_1666	19 bpp TRGB 1666
8	19BPP_TRGB_PACKED	19 bpp TRGB 1666 PACKED
9	24BPP_RGB_888	24 bpp RGB 888
10	24BPP_RGB_888_PACKED	24 bpp RGB 888 PACKED
11	25BPP_TRGB_1888	25 bpp TRGB 1888
12	32BPP_ARGB_8888	32 bpp ARGB 8888
13	32BPP_RGBA_8888	32 bpp RGBA 8888

CLUTMODE: Color Lookup Table Mode Input Selection

Value	Name	Description
0	1BPP	color lookup table mode set to 1 bit per pixel
1	2BPP	color lookup table mode set to 2 bits per pixel
2	4BPP	color lookup table mode set to 4 bits per pixel
3	8BPP	color lookup table mode set to 8 bits per pixel

43.7.28 Base Layer Configuration 2 Register

Name: LCDC_BASECFG2

Address:0xF8038074

Access: Read/Write

31	30	29	28	27	26	25	24
XSTRIDE							
23	22	21	20	19	18	17	16
XSTRIDE							
15	14	13	12	11	10	9	8
XSTRIDE							
7	6	5	4	3	2	1	0
XSTRIDE							

XSTRIDE: Horizontal Stride

XSTRIDE represents the memory offset, in bytes, between two rows of the image memory.

© 2017 Microchip Technology Inc.

43.7.29 Base Layer Configuration 3 Register

Name: LCDC_BASECFG3

Address:0xF8038078

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
RDEF							
15	14	13	12	11	10	9	8
GDEF							
7	6	5	4	3	2	1	0
BDEF							

RDEF: Red Default

Default Red color when the Base DMA channel is disabled.

GDEF: Green Default

Default Green color when the Base DMA channel is disabled.

BDEF: Blue Default

Default Blue color when the Base DMA channel is disabled.

43.7.30 Base Layer Configuration 4 Register

Name: LCDC_BASECFG4

Address:0xF803807C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	REP	DMA
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

DMA: Use DMA Data Path

0: The default color is used on the Base Layer.

1: The DMA channel retrieves the pixels stream from the memory.

REP: Use Replication logic to expand RGB color to 24 bits

0: When the selected pixel depth is less than 24 bpp the pixel is shifted and least significant bits are set to 0.

1: When the selected pixel depth is less than 24 bpp the pixel is shifted and the least significant bit replicates the MSB.

© 2017 Microchip Technology Inc.

43.7.31 Base CLUT Register x Register

Name: LCDC_BASECLUTx [x=0..255]

Address: 0xF8038400

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
RCLUT							
15	14	13	12	11	10	9	8
GCLUT							
7	6	5	4	3	2	1	0
BCLUT							

BCLUT: Blue Color entry

This field indicates the 8-bit width Blue color of the color lookup table.

GCLUT: Green Color entry

This field indicates the 8-bit width Green color of the color lookup table.

RCLUT: Red Color entry

This field indicates the 8-bit width Red color of the color lookup table.

44. Advanced Encryption Standard (AES)

44.1 Description

The Advanced Encryption Standard (AES) is compliant with the American *FIPS (Federal Information Processing Standard) Publication 197* specification.

The AES supports all five confidentiality modes of operation for symmetrical key block cipher algorithms (ECB, CBC, OFB, CFB and CTR), as specified in the *NIST Special Publication 800-38A Recommendation*. It is compatible with all these modes via DMA Controller channels, minimizing processor intervention for large buffer transfers.

The 128-bit/192-bit/256-bit key is stored in four/six/eight 32-bit write-only AES Key Word Registers (AES_KEYWR0–3).

The 128-bit input data and initialization vector (for some modes) are each stored in four 32-bit write-only AES Input Data Registers (AES_IDATAR0–3) and AES Initialization Vector Registers (AES_IVR0–3).

As soon as the initialization vector, the input data and the key are configured, the encryption/decryption process may be started. Then the encrypted/decrypted data are ready to be read out on the four 32-bit AES Output Data Registers (AES_ODATAR0–3) or through the DMA channels.

44.2 Embedded Characteristics

- Compliant with *FIPS Publication 197, Advanced Encryption Standard (AES)*
- 128-bit/192-bit/256-bit Cryptographic Key
- 12/14/16 Clock Cycles Encryption/Decryption Processing Time with a 128-bit/192-bit/256-bit Cryptographic Key
- Double Input Buffer Optimizes Runtime
- Support of the Modes of Operation Specified in the *NIST Special Publication 800-38A*:
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC) including CBC-MAC
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - Counter (CTR)
- 8, 16, 32, 64 and 128-bit Data Sizes Possible in CFB Mode
- Last Output Data Mode Allows Optimized Message Authentication Code (MAC) Generation
- Connection to DMA Optimizes Data Transfers for all Operating Modes

44.3 Product Dependencies

44.3.1 Power Management

The AES may be clocked through the Power Management Controller (PMC), so the programmer must first to configure the PMC to enable the AES clock.

44.3.2 Interrupt

The AES interface has an interrupt line connected to the Interrupt Controller.

Handling the AES interrupt requires programming the Interrupt Controller before configuring the AES.

Table 44-1: Peripheral IDs

Instance	ID
AES	29

44.4 Functional Description

The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information.

Encryption converts data to an unintelligible form called ciphertext. Decrypting the ciphertext converts the data back into its original form, called plaintext. The CIPHER bit in the AES Mode Register (AES_MR) allows selection between the encryption and the decryption processes.

© 2017 Microchip Technology Inc.

The AES is capable of using cryptographic keys of 128/192/256 bits to encrypt and decrypt data in blocks of 128 bits. This 128-bit/192-bit/256-bit key is defined in the AES_KEYWRx.

The input to the encryption processes of the CBC, CFB, and OFB modes includes, in addition to the plaintext, a 128-bit data block called the initialization vector (IV), which must be set in the AES_IVRx. The initialization vector is used in an initial step in the encryption of a message and in the corresponding decryption of the message. The AES_IVRx are also used by the CTR mode to set the counter value.

44.4.1 AES Register Endianism

In Arm processor-based products, the system bus and processors manipulate data in little-endian form. The AES interface requires little-endian format words. However, in accordance with the protocol of the FIPS 197 specification, data is collected, processed and stored by the AES algorithm in big-endian form.

The following example illustrates how to configure the AES:

If the first 64 bits of a message (according to FIPS 197, i.e., big-endian format) to be processed is 0xcafedeca_01234567, then the AES_IDATAR0 and AES_IDATAR1 registers must be written with the following pattern:

- AES_IDATAR0 = 0xcadefeca
- AES_IDATAR1 = 0x67452301

44.4.2 Operation Modes

The AES supports the following modes of operation:

- ECB: Electronic Code Book
- CBC: Cipher Block Chaining
- OFB: Output Feedback
- CFB: Cipher Feedback
  - CFB8 (CFB where the length of the data segment is 8 bits)
  - CFB16 (CFB where the length of the data segment is 16 bits)
  - CFB32 (CFB where the length of the data segment is 32 bits)
  - CFB64 (CFB where the length of the data segment is 64 bits)
  - CFB128 (CFB where the length of the data segment is 128 bits)
- CTR: Counter

The data pre-processing, post-processing and data chaining for the concerned modes are automatically performed. Refer to the *NIST Special Publication 800-38A* for more complete information.

These modes are selected by setting the OPMOD field in the AES_MR.

In CFB mode, five data sizes are possible (8, 16, 32, 64 or 128 bits), configurable by means of the CFBS field in the AES_MR (Section 44.5.2 “AES Mode Register”).

In CTR mode, the size of the block counter embedded in the module is 16 bits. Therefore, there is a rollover after processing 1 megabyte of data. If the file to be processed is greater than 1 megabyte, this file must be split into fragments of 1 megabyte or less for the first fragment if the initial value of the counter is greater than 0. Prior to loading the first fragment into AES_IDATARx, AES_IVRx must be fully programmed with the initial counter value. For any fragment, after the transfer is completed and prior to transferring the next fragment, AES_IVRx must be programmed with the appropriate counter value.

If the initial value of the counter is greater than 0 and the data buffer size to be processed is greater than 1 megabyte, the size of the first fragment to be processed must be 1 megabyte minus 16x(initial value) to prevent a rollover of the internal 1-bit counter.

To have a sequential increment, the counter value must be programmed with the value programmed for the previous fragment + 2^{16} (or less for the first fragment).

All AES_IVRx fields must be programmed to take into account the possible carry propagation.

44.4.3 Double Input Buffer

The AES_IDATARx can be double-buffered to reduce the runtime of large files.

This mode allows writing a new message block when the previous message block is being processed. This is only possible when DMA accesses are performed (SMOD = 0x2).

The DUALBUFF bit in the AES_MR must be set to ‘1’ to access the double buffer.

© 2017 Microchip Technology Inc.

44.4.4 Start Modes

The SMOD field in the AES_MR allows selection of the encryption (or decryption) Start mode.

44.4.4.1 Manual Mode

The sequence order is as follows:

1. Write the AES_MR with all required fields, including but not limited to SMOD and OPMOD.
2. Write the 128-bit/192-bit/256-bit key in the AES_KEYWRx.
3. Write the initialization vector (or counter) in the AES_IVRx.

Note: The AES_IVRx concern all modes except ECB.

4. Set the bit DATRDY (Data Ready) in the AES Interrupt Enable Register (AES_IER), depending on whether an interrupt is required or not at the end of processing.
5. Write the data to be encrypted/decrypted in the authorized AES_IDATARx (see Table 44-2).
6. Set the START bit in the AES Control Register (AES_CR) to begin the encryption or the decryption process.
7. When processing completes, the DATRDY flag in the AES Interrupt Status Register (AES_ISR) is raised. If an interrupt has been enabled by setting the DATRDY bit in the AES_IER, the interrupt line of the AES is activated.
8. When software reads one of the AES_ODATARx, the DATRDY bit is automatically cleared.

Table 44-2: Authorized Input Data Registers

Operation Mode	Input Data Registers to Write
ECB	All
CBC	All
OFB	All
128-bit CFB	All
64-bit CFB	AES_IDATAR0 and AES_IDATAR1
32-bit CFB	AES_IDATAR0
16-bit CFB	AES_IDATAR0
8-bit CFB	AES_IDATAR0
CTR	All

Note 1: In 64-bit CFB mode, writing to AES_IDATAR2 and AES_IDATAR3 is not allowed and may lead to errors in processing.

2: In 32, 16, and 8-bit CFB modes, writing to AES_IDATAR1, AES_IDATAR2 and AES_IDATAR3 is not allowed and may lead to errors in processing.

44.4.4.2 Auto Mode

The Auto Mode is similar to the manual one, except that in this mode, as soon as the correct number of AES_IDATARx is written, processing is automatically started without any action in the AES_CR.

44.4.4.3 DMA Mode

The DMA Controller can be used in association with the AES to perform an encryption/decryption of a buffer without any action by software during processing.

The SMOD field in the AES_MR must be configured to 0x2 and the DMA must be configured with non-incremental addresses.

The start address of any transfer descriptor must be configured with the address of AES_IDATAR0.

The DMA chunk size configuration depends on the AES mode of operation and is listed in Table 44-3 “DMA Data Transfer Type for the Different Operation Modes”.

© 2017 Microchip Technology Inc.

When writing data to AES with a first DMA channel, data are first fetched from a memory buffer (source data). It is recommended to configure the size of source data to “words” even for CFB modes. On the contrary, the destination data size depends on the mode of operation. When reading data from the AES with the second DMA channel, the source data is the data read from AES and data destination is the memory buffer. In this case, the source data size depends on the AES mode of operation and is listed in Table 44-3.

Table 44-3: DMA Data Transfer Type for the Different Operation Modes

Operation Mode	Chunk Size	Destination/Source Data Transfer Type
ECB	4	Word
CBC	4	Word
OFB	4	Word
CFB 128-bit	4	Word
CFB 64-bit	1	Word
CFB 32-bit	1	Word
CFB 16-bit	1	Half-word
CFB 8-bit	1	Byte
CTR	4	Word

44.4.5 Last Output Data Mode

This mode is used to generate cryptographic checksums on data (MAC) by means of cipher block chaining encryption algorithm (CBC-MAC algorithm for example).

After each end of encryption/decryption, the output data are available either on the AES_ODATARx for Manual and Auto mode or at the address specified in the receive buffer pointer for DMA mode (see Table 44-4 “Last Output Data Mode Behavior versus Start Modes”).

The Last Output Data (LOD) bit in the AES_MR allows retrieval of only the last data of several encryption/decryption processes.

Therefore, there is no need to define a read buffer in DMA mode.

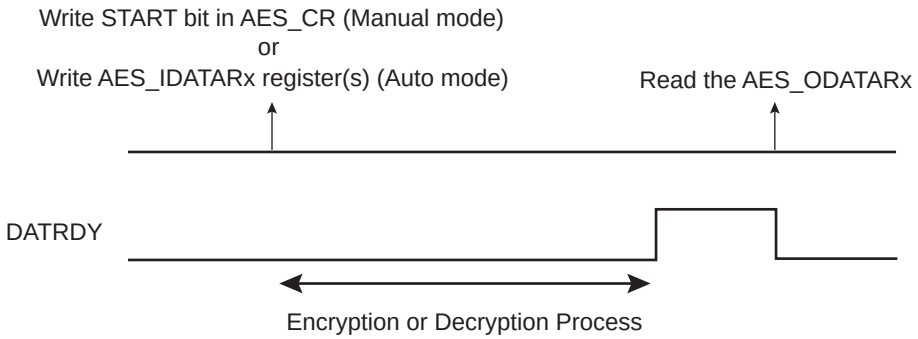
This data are only available on the AES_ODATARx.

44.4.5.1 Manual and Auto Modes

- If AES_MR.LOD = 0

The DATRDY flag is cleared when at least one of the AES_ODATARx is read (see Figure 44-1).

Figure 44-1: Manual and Auto Modes with AES_MR.LOD = 0



If the user does not want to read the AES_ODATARx between each encryption/decryption, the DATRDY flag will not be cleared. If the DATRDY flag is not cleared, the user cannot know the end of the following encryptions/decryptions.

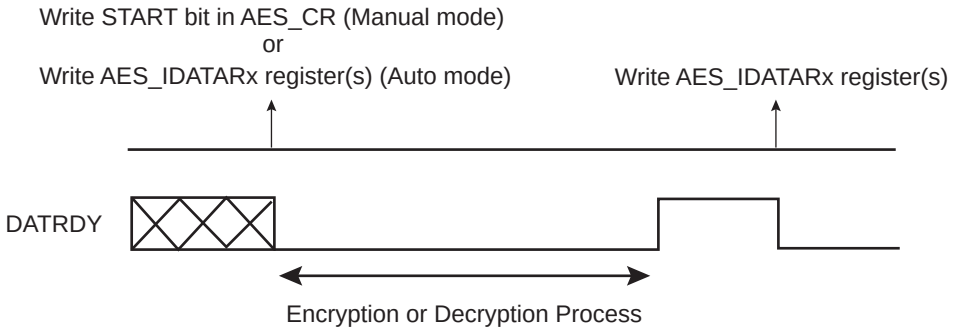
- If AES_MR.LOD = 1

This mode is optimized to process AES CPC-MAC operating mode.

The DATRDY flag is cleared when at least one AES_IDATAR is written (see Figure 44-2). No more AES_ODATAR reads are necessary between consecutive encryptions/decryptions.

© 2017 Microchip Technology Inc.

Figure 44-2: Manual and Auto Modes with AES_MR.LOD = 1



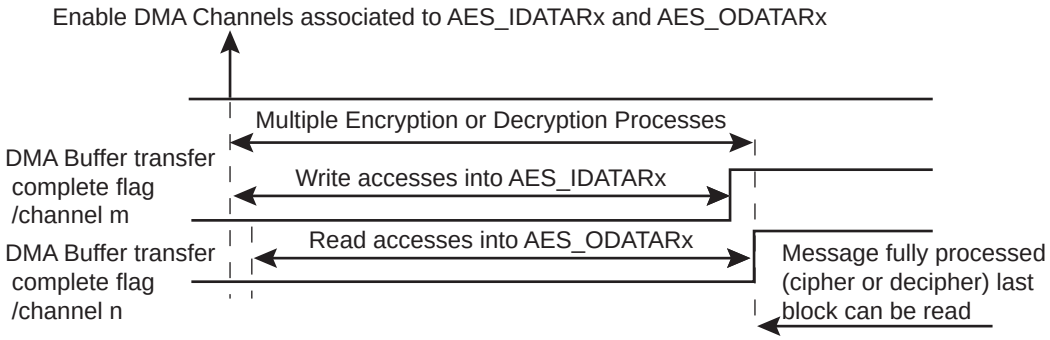
44.4.5.2 DMA Mode

- If AES_MR.LOD = 0

This mode may be used for all AES operating modes except CBC-MAC where AES_MR.LOD = 1 mode is recommended.

The end of the encryption/decryption is indicated by the end of DMA transfer associated to AES_ODATARx (see Figure 44-3). Two DMA channels are required: one for writing message blocks to AES_IDATARx and one to obtain the result from AES_ODATARx.

Figure 44-3: DMA Transfer with AES_MR.LOD = 0



- If AES_MR.LOD = 1

This mode is optimized to process AES CBC-MAC operating mode.

The user must first wait for the DMA buffer transfer complete flag, then for the flag DATRDY to rise to ensure that the encryption/decryption is completed (see Figure 44-4).

In this case, no receive buffers are required.

The output data are only available on the AES_ODATARx.

Figure 44-4: DMA Transfer with AES_MR.LOD = 1

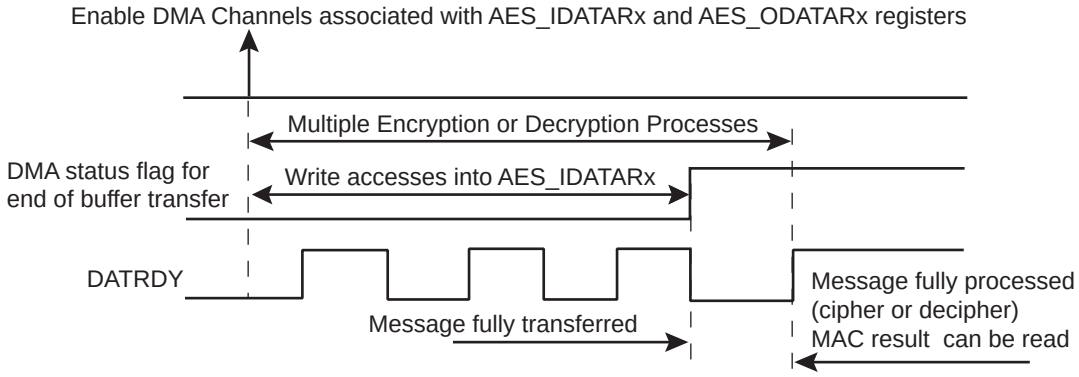


Table 44-4 summarizes the different cases.

© 2017 Microchip Technology Inc.

Table 44-4: Last Output Data Mode Behavior versus Start Modes

Sequence	Manual and Auto Modes		DMA Transfer	
	AES_MR.LOD = 0	AES_MR.LOD = 1	AES_MR.LOD = 0	AES_MR.LOD = 1
DATRDY Flag Clearing Condition(1)	At least one AES_ODATAR must be read	At least one AES_IDATAR must be written	Not used	Managed by the DMA
End of Encryption/ Decryption Notification	DATRDY	DATRDY	2 DMA Buffer transfer complete flags (channel m and channel n)	DMA buffer transfer complete flag, then AES DATRDY flag
Encrypted/Decrypted Data Result Location	In the AES_ODATARx	In the AES_ODATARx	At the address specified in the Channel Buffer Transfer Descriptor	In the AES_ODATARx

Note 1: Depending on the mode, there are other ways of clearing the DATRDY flag. See Section 44.5.6 “AES Interrupt Status Register”.

Warning: In DMA mode, reading the AES_ODATARx before the last data transfer may lead to unpredictable results.

44.4.6 Security Features

44.4.6.1 Unspecified Register Access Detection

When an unspecified register access occurs, the URAD flag in the AES_ISR is raised. Its source is then reported in the Unspecified Register Access Type (URAT) field. Only the last unspecified register access is available through the URAT field.

Several kinds of unspecified register accesses can occur:

- Input Data Register written during the data processing when SMOD = IDATAR0_START
- Output Data Register read during data processing
- Mode Register written during data processing
- Output Data Register read during sub-keys generation
- Mode Register written during sub-keys generation
- Write-only register read access

The URAD bit and the URAT field can only be reset by the SWRST bit in the AES_CR.

44.5 Advanced Encryption Standard (AES) User Interface

Table 44-5: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	AES_CR	Write-only	–
0x04	Mode Register	AES_MR	Read/Write	0x0
0x08–0x0C	Reserved	–	–	–
0x10	Interrupt Enable Register	AES_IER	Write-only	–
0x14	Interrupt Disable Register	AES_IDR	Write-only	–
0x18	Interrupt Mask Register	AES_IMR	Read-only	0x0
0x1C	Interrupt Status Register	AES_ISR	Read-only	0x0
0x20	Key Word Register 0	AES_KEYWR0	Write-only	–
0x24	Key Word Register 1	AES_KEYWR1	Write-only	–
0x28	Key Word Register 2	AES_KEYWR2	Write-only	–
0x2C	Key Word Register 3	AES_KEYWR3	Write-only	–
0x30	Key Word Register 4	AES_KEYWR4	Write-only	–
0x34	Key Word Register 5	AES_KEYWR5	Write-only	–
0x38	Key Word Register 6	AES_KEYWR6	Write-only	–
0x3C	Key Word Register 7	AES_KEYWR7	Write-only	–
0x40	Input Data Register 0	AES_IDATAR0	Write-only	–
0x44	Input Data Register 1	AES_IDATAR1	Write-only	–
0x48	Input Data Register 2	AES_IDATAR2	Write-only	–
0x4C	Input Data Register 3	AES_IDATAR3	Write-only	–
0x50	Output Data Register 0	AES_ODATAR0	Read-only	0x0
0x54	Output Data Register 1	AES_ODATAR1	Read-only	0x0
0x58	Output Data Register 2	AES_ODATAR2	Read-only	0x0
0x5C	Output Data Register 3	AES_ODATAR3	Read-only	0x0
0x60	Initialization Vector Register 0	AES_IVR0	Write-only	–
0x64	Initialization Vector Register 1	AES_IVR1	Write-only	–
0x68	Initialization Vector Register 2	AES_IVR2	Write-only	–
0x6C	Initialization Vector Register 3	AES_IVR3	Write-only	–
0x70–0xAC	Reserved	–	–	–
0xB0–0xFC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

44.5.1 AES Control Register

Name: AES_CR

Address:0xF000C000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	SWRST
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	START

START: Start Processing

0: No effect.

1: Starts manual encryption/decryption process.

SWRST: Software Reset

0: No effect.

1: Resets the AES. A software-triggered hardware reset of the AES interface is performed.

44.5.2 AES Mode Register

Name: AES_MR

Address:0xF000C004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CKEY				–	CFBS		
15	14	13	12	11	10	9	8
LOD	OPMOD			KEYSIZE		SMOD	
7	6	5	4	3	2	1	0
PROCDLY				DUALBUFF	–	–	CIPHER

CIPHER: Processing Mode

0: Decrypts data.

1: Encrypts data.

DUALBUFF: Dual Input Buffer

Value	Name	Description
0	INACTIVE	AES_IDATARx cannot be written during processing of previous block.
1	ACTIVE	AES_IDATARx can be written during processing of previous block when SMOD = 2. It speeds up the overall runtime of large files.

PROCDLY: Processing Delay

Processing Time = N × (PROCDLY + 1)

where

N = 10 when KEYSIZE = 0

N = 12 when KEYSIZE = 1

N = 14 when KEYSIZE = 2

The processing time represents the number of clock cycles that the AES needs in order to perform one encryption/decryption.

Note: The best performance is achieved with PROCDLY equal to 0.

SMOD: Start Mode

Value	Name	Description
0	MANUAL_START	Manual Mode
1	AUTO_START	Auto Mode
2	IDATAR0_START	AES_IDATAR0 access only Auto Mode (DMA)

Values which are not listed in the table must be considered as “reserved”.

If a DMA transfer is used, configure SMOD to 0x2. Refer to Section 44.4.4.3 “DMA Mode” for more details.

© 2017 Microchip Technology Inc.

KEYSIZE: Key Size

Value	Name	Description
0	AES128	AES Key Size is 128 bits
1	AES192	AES Key Size is 192 bits
2	AES256	AES Key Size is 256 bits

Values which are not listed in the table must be considered as “reserved”.

OPMOD: Operation Mode

Value	Name	Description
0	ECB	ECB: Electronic Code Book mode
1	CBC	CBC: Cipher Block Chaining mode
2	OFB	OFB: Output Feedback mode
3	CFB	CFB: Cipher Feedback mode
4	CTR	CTR: Counter mode (16-bit internal counter)

Values which are not listed in the table must be considered as “reserved”.

For CBC-MAC operating mode, set OPMOD to CBC and LOD to 1.

LOD: Last Output Data Mode

0: No effect.

After each end of encryption/decryption, the output data are available either on the output data registers (Manual and Auto modes) or at the address specified in the Channel Buffer Transfer Descriptor for DMA mode.

In Manual and Auto modes, the DATRDY flag is cleared when at least one of the Output Data registers is read.

1: The DATRDY flag is cleared when at least one of the Input Data Registers is written.

No more Output Data Register reads is necessary between consecutive encryptions/decryptions (see Section 44.4.5 “Last Output Data Mode”).

Warning: In DMA mode, reading to the Output Data registers before the last data encryption/decryption process may lead to unpredictable results.

CFBS: Cipher Feedback Data Size

Value	Name	Description
0	SIZE_128BIT	128-bit
1	SIZE_64BIT	64-bit
2	SIZE_32BIT	32-bit
3	SIZE_16BIT	16-bit
4	SIZE_8BIT	8-bit

Values which are not listed in table must be considered as “reserved”.

CKEY: Key

Value	Name	Description
0xE	PASSWD	This field must be written with 0xE the first time the AES_MR is programmed. For subsequent programming of the AES_MR, any value can be written, including that of 0xE. Always reads as 0.

44.5.3 AES Interrupt Enable Register

Name: AES_IER

Address:0xF000C010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

DATRDY: Data Ready Interrupt Enable

URAD: Unspecified Register Access Detection Interrupt Enable

© 2017 Microchip Technology Inc.

44.5.4 AES Interrupt Disable Register

Name: AES_IDR

Address:0xF000C014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

DATRDY: Data Ready Interrupt Disable

URAD: Unspecified Register Access Detection Interrupt Disable

44.5.5 AES Interrupt Mask Register

Name: AES_IMR

Address:0xF000C018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

DATRDY: Data Ready Interrupt Mask

URAD: Unspecified Register Access Detection Interrupt Mask

© 2017 Microchip Technology Inc.

44.5.6 AES Interrupt Status Register

Name: AES_ISR

Address:0xF000C01C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
URAT				–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

DATRDY: Data Ready (cleared by setting bit START or bit SWRST in AES_CR or by reading AES_ODATARx)

0: Output data not valid.

1: Encryption or decryption process is completed.

Note: If AES_MR.LOD = 1: In Manual and Auto mode, the DATRDY flag can also be cleared by writing at least one AES_IDATARx.

URAD: Unspecified Register Access Detection Status (cleared by writing SWRST in AES_CR)

0: No unspecified register access has been detected since the last SWRST.

1: At least one unspecified register access has been detected since the last SWRST.

URAT: Unspecified Register Access (cleared by writing SWRST in AES_CR)

Value	Name	Description
0	IDR_WR_PROCESSING	Input Data Register written during the data processing when SMOD = 0x2 mode.
1	ODR_RD_PROCESSING	Output Data Register read during the data processing.
2	MR_WR_PROCESSING	Mode Register written during the data processing.
3	ODR_RD_SUBKGEN	Output Data Register read during the sub-keys generation.
4	MR_WR_SUBKGEN	Mode Register written during the sub-keys generation.
5	WOR_RD_ACCESS	Write-only register read access.

Only the last Unspecified Register Access Type is available through the URAT field.

44.5.7 AES Key Word Register x

Name: AES_KEYWRx [x=0..7]

Address:0xF000C020

Access: Write-only

31	30	29	28	27	26	25	24
KEYW							

23	22	21	20	19	18	17	16
KEYW							

15	14	13	12	11	10	9	8
KEYW							

7	6	5	4	3	2	1	0
KEYW							

KEYW: Key Word

The four/six/eight 32-bit Key Word Registers set the 128-bit/192-bit/256-bit cryptographic key used for AES encryption/decryption. AES_KEYWR0 corresponds to the first word of the key and respectively AES_KEYWR3/AES_KEYWR5/AES_KEYWR7 to the last one. These registers are write-only to prevent the key from being read by another application.

© 2017 Microchip Technology Inc.

44.5.8 AES Input Data Register x

Name: AES_IDATARx [x=0..3]

Address:0xF000C040

Access: Write-only

31	30	29	28	27	26	25	24
IDATA							
23	22	21	20	19	18	17	16
IDATA							
15	14	13	12	11	10	9	8
IDATA							
7	6	5	4	3	2	1	0
IDATA							

IDATA: Input Data Word

The four 32-bit Input Data registers set the 128-bit data block used for encryption/decryption.

AES_IDATAR0 corresponds to the first word of the data to be encrypted/decrypted, and AES_IDATAR3 to the last one.

These registers are write-only to prevent the input data from being read by another application.

44.5.9 AES Output Data Register x

Name: AES_ODATARx [x=0..3]

Address:0xF000C050

Access: Read-only

31	30	29	28	27	26	25	24
ODATA							

23	22	21	20	19	18	17	16
ODATA							

15	14	13	12	11	10	9	8
ODATA							

7	6	5	4	3	2	1	0
ODATA							

ODATA: Output Data

The four 32-bit Output Data registers contain the 128-bit data block that has been encrypted/decrypted.

AES_ODATAR0 corresponds to the first word, AES_ODATAR3 to the last one.

© 2017 Microchip Technology Inc.

44.5.10 AES Initialization Vector Register x

Name: AES_IVRx [x=0..3]

Address:0xF000C060

Access: Write-only

31	30	29	28	27	26	25	24
IV							
23	22	21	20	19	18	17	16
IV							
15	14	13	12	11	10	9	8
IV							
7	6	5	4	3	2	1	0
IV							

IV: Initialization Vector

The four 32-bit Initialization Vector Registers set the 128-bit Initialization Vector data block that is used by some modes of operation as an additional initial input.

AES_IVR0 corresponds to the first word of the Initialization Vector, AES_IVR3 to the last one.

These registers are write-only to prevent the Initialization Vector from being read by another application.

For CBC, OFB and CFB modes, the IV input value corresponds to the initialization vector.

For CTR mode, the IV input value corresponds to the initial counter value.

Note: These registers are not used in ECB mode and must not be written.

45. Secure Hash Algorithm (SHA)

45.1 Description

The Secure Hash Algorithm (SHA) is compliant with the American *FIPS (Federal Information Processing Standard) Publication 180-2* specification.

The 512-bit block of message is respectively stored in 16 x 32-bit registers, (SHA_IDATARx/SHA_ODATARx) which are write-only.

As soon as the input data is written, the hash processing may be started. The registers comprising the block of a padded message must be entered consecutively. Then the message digest is ready to be read out on the 5 up to 8 x 32-bit output data registers (SHA_ODATARx) or through the DMA channels.

45.2 Embedded Characteristics

- Supports Secure Hash Algorithm (SHA1, SHA224, SHA256,)
- Compliant with *FIPS Publication 180-2*
- Configurable Processing Period:
  - 85 Clock Cycles to obtain a fast SHA1 runtime or 209 Clock Cycles for Maximizing Bandwidth of Other Applications
  - 72 Clock Cycles to obtain a fast SHA224, SHA256 runtime or 194 Clock Cycles for Maximizing Bandwidth of Other Applications
- Connection to DMA Channel Capabilities Optimizes Data Transfers
- Double Input Buffer Optimizes Runtime

45.3 Product Dependencies

45.3.1 Power Management

The SHA may be clocked through the Power Management Controller (PMC), so the programmer must first configure the PMC to enable the SHA clock.

45.3.2 Interrupt Sources

The SHA interface has an interrupt line connected to the Interrupt Controller.

Handling the SHA interrupt requires programming the interrupt controller before configuring the SHA.

Table 45-1: Peripheral IDs

Instance	ID
SHA	27

45.4 Functional Description

The Secure Hash Algorithm (SHA) module requires a padded message according to FIPS180-2 specification. The first block of the message must be indicated to the module by a specific command. The SHA module produces an N-bit message digest each time a block is written and processing period ends, where N is 160 for SHA1, 224 for SHA224,256 for SHA256.

45.4.1 SHA Algorithm

The SHA can process SHA1, SHA224, SHA256 by configuring the ALGO field in the SHA Mode register (SHA_MR).

45.4.2 Processing Period

The processing period can be configured.

The short processing period allocates bandwidth to the SHA module, whereas the long processing period allocates more bandwidth on the system bus to other applications. An example is DMA channels not associated with SHA.

In SHA1 mode, the shortest processing period is 85 clock cycles + 2 clock cycles for start command synchronization. The longest period is 209 clock cycles + 2 clock cycles.

In SHA256 and SHA224 mode, the shortest processing period is 72 clock cycles + 2 clock cycles for start command synchronization. The longest period is 194 clock cycles + 2 clock cycles.

© 2017 Microchip Technology Inc.

45.4.3 Double Input Buffer

The SHA Input Data registers (SHA_IDATARx) can be double-buffered to reduce the runtime of large files.

Double-buffering allows a new message block to be written while the previous message block is being processed. This is only possible when DMA accesses are performed (SMOD = 2).

The DUALBUFF bit in the SHA_MR must be set to have double input buffer access.

45.4.4 Start Modes

The SMOD field in the SHA_MR is used to select the Hash Processing Start mode.

45.4.4.1 Manual Mode

In Manual mode, the sequence is as follows:

1. Set the bit DATRDY (Data Ready) in the SHA_IER, depending on whether an interrupt is required at the end of processing.
2. For the first block of a message, the FIRST command must be set by writing a 1 into the corresponding bit of the SHA Control Register (SHA_CR). For the other blocks, there is nothing to write.
3. Write the block to be processed in the SHA_IDATARx.
4. To begin processing, set the START bit in the SHA_CR.
5. When processing is completed, the bit DATRDY in the Interrupt Status register (SHA_ISR) raises. If an interrupt has been enabled by setting the bit DATRDY in SHA_IER, the interrupt line of the SHA is activated.
6. Repeat the write procedure for each block, start procedure and wait for the interrupt procedure up to the last block of the entire message. Each time the start procedure is complete, the DATRDY flag is cleared.
7. After the last block is processed (DATRDY flag is set, if an interrupt has been enabled by setting the bit DATRDY in SHA_IER, the interrupt line of the SHA is activated), read the message digest in the Output Data Registers. The DATRDY flag is automatically cleared when reading the SHA_ODATARx registers.

45.4.4.2 Auto Mode

In Auto mode, processing starts as soon as the correct number of SHA_IDATARx is written. No action in the SHA_CR is necessary.

45.4.4.3 DMA Mode

The DMA can be used in association with the SHA to perform the algorithm on a complete message without any action by the software during processing.

The SMOD field in SHA_MR must be configured to 2.

The DMA must be configured with non-incremental addresses.

The start address of any transfer descriptor must be set to point to the SHA_IDATAR0.

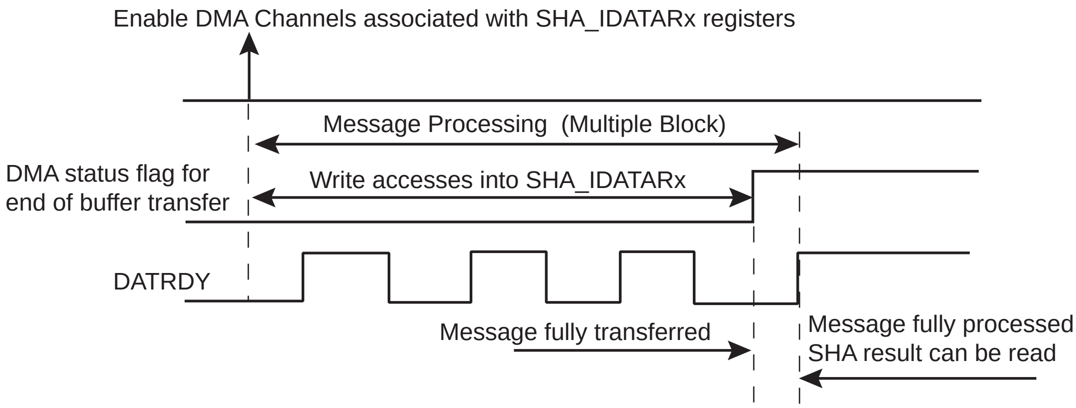
The DMA chunk size must be set to transfer, for each trigger request, 16 words of 32 bits.

The FIRST bit of the SHA_CR must be set before starting the DMA when the first block is transferred.

The DMA generates an interrupt when the end of buffer transfer is completed but the SHA processing is still in progress. The end of SHA processing is indicated by the flag DATRDY in the SHA_SR.

The end of SHA processing requires two interrupts to be verified. The DMA end of transfer interrupt must be verified first, then the SHA DATRDY interrupt must be enabled and verified (see Figure 45-1).

Figure 45-1: interrupts Processing with DMA



45.4.4.4 SHA Register Endianism

In Arm processor-based products, the system bus and processors manipulate data in little-endian form. The SHA interface requires little-endian format words. However, in accordance with the protocol of FIPS 180-2 specification, data is collected, processed and stored by the SHA algorithm in big-endian form.

The following example illustrates how to configure the SHA:

If the first 64 bits of a message (according to FIPS 180-2, i.e., big-endian format) to be processed is 0xcafedeca_01234567, then the SHA_IDATAR0 and SHA_IDATAR1 registers must be written with the following pattern:

- SHA_IDATAR0 = 0xcadefeca
- SHA_IDATAR1 = 0x67452301

In a little-endian system, the message (according to FIPS 180-2) starting with pattern 0xcafedeca_01234567 is stored into memory as follows:

- 0xca stored at initial offset (for example 0x00),
- then 0xfe stored at initial offset + 1 (i.e., 0x01),
- 0xde stored at initial offset + 2 (i.e., 0x02),
- 0xca stored at initial offset + 3 (i.e., 0x03).

If the message is received through a serial-to-parallel communication channel, the first received character is 0xca and it is stored at the first memory location (initial offset). The second byte, 0xfe, is stored at initial offset + 1.

When reading on a 32-bit little-endian system bus, the first word read back from system memory is 0xcadefeca.

When the SHA_ODATARx registers are read, the hash result is organized in little-endian format, allowing system memory storage in the same format as the message.

Taking an example from the FIPS 180-2 specification Appendix B.1, the endianism conversion can be observed.

For this example, the 512-bit message is:

0x61626380000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000018

and the expected SHA-256 result is:

0xba7816bf_8f01cfea_414140de_5dae2223_b00361a3_96177a9c_b410ff61_f20015ad

If the message has not already been stored in the system memory, the first step is to convert the input message to little-endian before writing to the SHA_IDATARx registers. This would result in a write of:

SHA_IDATAR0 = 0x80636261...... SHA_IDATAR15 = 0x18000000

The data in the output message digest registers, SHA_ODATARx, contain SHA_ODATAR0 = 0xbf1678ba... SHA_ODATAR7 = 0xad1500f2 which is the little-endian format of 0xba7816bf,..., 0xf20015ad.

Reading SHA_ODATAR0 to SHA_ODATAR1 and storing into a little-endian memory system forces hash results to be stored in the same format as the message.

When the output message is read, the user can convert back to big-endian for a resulting message value of:

0xba7816bf_8f01cfea_414140de_5dae2223_b00361a3_96177a9c_b410ff61_f20015ad

© 2017 Microchip Technology Inc.

45.4.5 Security Features

When an unspecified register access occurs, the URAD bit in the SHA_ISR is set. Its source is then reported in the Unspecified Register Access Type field (URAT). Only the last unspecified register access is available through the URAT field.

Several kinds of unspecified register accesses can occur:

- SHA_IDATARx written during data processing in DMA mode
- SHA_ODATARx read during data processing
- SHA_MR written during data processing
- Write-only register read access

The URAD bit and the URAT field can only be reset by the SWRST bit in the SHA_CR.

45.5 Secure Hash Algorithm (SHA) User Interface

Table 45-2: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	SHA_CR	Write-only	–
0x04	Mode Register	SHA_MR	Read/Write	0x0000100
0x08–0x0C	Reserved	–	–	–
0x10	Interrupt Enable Register	SHA_IER	Write-only	–
0x14	Interrupt Disable Register	SHA_IDR	Write-only	–
0x18	Interrupt Mask Register	SHA_IMR	Read-only	0x0
0x1C	Interrupt Status Register	SHA_ISR	Read-only	0x0
0x20–0x3C	Reserved	–	–	–
0x40	Input Data 0 Register	SHA_IDATAR0	Write-only	–
...	...	...	...	...
0x7C	Input Data 15 Register	SHA_IDATAR15	Write-only	–
0x80	Output Data 0 Register	SHA_ODATAR0	Read-only	0x0
...	...	...	...	...
0x9C	Output Data 7 Register	SHA_ODATAR7	Read-only	0x0
0xA0–0xFC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

45.5.1 SHA Control Register

Name: SHA_CR

Address:0xF0014000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	SWRST
7	6	5	4	3	2	1	0
–	–	–	FIRST	–	–	–	START

START: Start Processing

0: No effect.

1: Starts manual hash algorithm process.

FIRST: First Block of a Message

0: No effect.

1: Indicates that the next block to process is the first one of a message.

SWRST: Software Reset

0: No effect.

1: Resets the SHA. A software-triggered hardware reset of the SHA interface is performed.

45.5.2 SHA Mode Register

Name: SHA_MR

Address:0xF0014004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	DUALBUFF
15	14	13	12	11	10	9	8
–	–	–	–	ALGO			
7	6	5	4	3	2	1	0
–	–	–	PROCDLY	–	–	SMOD	

SMOD: Start Mode

Value	Name	Description
0	MANUAL_START	Manual Mode
1	AUTO_START	Auto Mode
2	IDATAR0_START	SHA_IDATAR0 access only Auto Mode

Values not listed in the table must be considered as “reserved”.

If a DMA transfer is used, configure the SMOD value with 1 or 2. Refer to Section 45.4.4.3 “DMA Mode” for more details.

PROCDLY: Processing Delay

Value	Name	Description
0	SHORTEST	SHA processing runtime is the shortest one
1	LONGEST	SHA processing runtime is the longest one (reduces the SHA bandwidth requirement, reduces the system bus overload)

When SHA1 algorithm is processed, runtime period is either 85 or 209 clock cycles.

When SHA256 or SHA224 algorithm is processed, runtime period is either 72 or 194 clock cycles.

ALGO: SHA Algorithm

Value	Name	Description
0	SHA1	SHA1 algorithm processed
1	SHA256	SHA256 algorithm processed
2	Reserved	–
3	Reserved	–
4	SHA224	SHA224 algorithm processed

Values not listed in the table must be considered as “reserved”.

DUALBUFF: Dual Input Buffer

Value	Name	Description
0	INACTIVE	SHA_IDATARx and SHA_ODATARx cannot be written during processing of previous block.
1	ACTIVE	SHA_IDATARx and SHA_ODATARx can be written during processing of previous block when SMOD value = 2. It speeds up the overall runtime of large files.

© 2017 Microchip Technology Inc.

45.5.3 SHA Interrupt Enable Register

Name: SHA_IER

Address:0xF0014010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

DATRDY: Data Ready Interrupt Enable

URAD: Unspecified Register Access Detection Interrupt Enable

45.5.4 SHA Interrupt Disable Register

Name: SHA_IDR

Address:0xF0014014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

DATRDY: Data Ready Interrupt Disable

URAD: Unspecified Register Access Detection Interrupt Disable

© 2017 Microchip Technology Inc.

45.5.5 SHA Interrupt Mask Register

Name: SHA_IMR

Address:0xF0014018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

DATRDY: Data Ready Interrupt Mask

URAD: Unspecified Register Access Detection Interrupt Mask

45.5.6 SHA Interrupt Status Register

Name: SHA_ISR

Address:0xF001401C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
URAT				–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

DATRDY: Data Ready (cleared by writing a 1 to bit SWRST or START in SHA_CR, or by reading SHA_IODATARx)

0: Output data is not valid.

1: 512-bit block process is completed.

DATRDY is cleared when one of the following conditions is met:

- Bit START in SHA_CR is set.
- Bit SWRST in SHA_CR is set.
- The hash result is read.

URAD: Unspecified Register Access Detection Status (cleared by writing a 1 to SWRST bit in SHA_CR)

0: No unspecified register access has been detected since the last SWRST.

1: At least one unspecified register access has been detected since the last SWRST.

URAT: Unspecified Register Access Type (cleared by writing a 1 to SWRST bit in SHA_CR)

Value	Description
0	SHA_IDATAR0 to SHA_IDATAR15 written during the data processing in DMA mode (URAD = 1 and URAT = 0 can occur only if DUALBUFF is cleared in SHA_MR).
1	Output Data Register read during the data processing.
2	SHA_MR written during the data processing.
3	Write-only register read access.

Only the last Unspecified Register Access Type is available through the URAT field.

© 2017 Microchip Technology Inc.

45.5.7 SHA Input Data x Register

Name: SHA_IDATARx [x=0..15]

Address:0xF0014040

Access: Write-only

31	30	29	28	27	26	25	24
IDATA							
23	22	21	20	19	18	17	16
IDATA							
15	14	13	12	11	10	9	8
IDATA							
7	6	5	4	3	2	1	0
IDATA							

IDATA: Input Data

The 32-bit Input Data registers allow to load the data block used for hash processing.

These registers are write-only to prevent the input data from being read by another application.

SHA_IDATAR0 corresponds to the first word of the block, SHA_IDATAR15 to the last word of the last block in case SHA algorithm is set to SHA1, SHA224, SHA256.

45.5.8 SHA Output Data Register x

Name: SHA_ODATARx [x=0..15]

Address:0xF0014080

Access: Read only

31	30	29	28	27	26	25	24
ODATA							

23	22	21	20	19	18	17	16
ODATA							

15	14	13	12	11	10	9	8
ODATA							

7	6	5	4	3	2	1	0
ODATA							

ODATA: Output Data

These registers can be used to read the resulting message digest.

When SHA processing is in progress, these registers return 0x0000.

SHA_ODATAR0 corresponds to the first word of the message digest; SHA_ODATAR4 to the last one in SHA1 mode, SHA_ODATAR6 in SHA224, SHA_ODATAR7 in SHA256, .

When SHA224 is selected, the content of SHA_ODATAR7 must be ignored.

© 2017 Microchip Technology Inc.

46. True Random Number Generator (TRNG)

46.1 Description

The True Random Number Generator (TRNG) passes the American *NIST Special Publication 800-22* (*A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*) and the *Diehard Suite of Tests*.

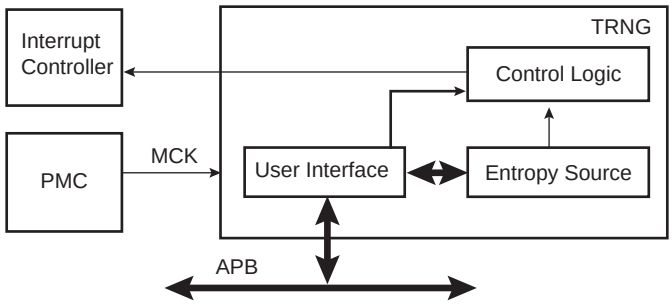
The TRNG may be used as an entropy source for seeding an NIST approved DRNG (Deterministic RNG) as required by FIPS PUB 140-2 and 140-3.

46.2 Embedded Characteristics

- Passes *NIST Special Publication 800-22* Test Suite
- Passes *Diehard Suite of Tests*
- May be used as Entropy Source for seeding an NIST approved DRNG (Deterministic RNG) as required by FIPS PUB 140-2 and 140-3
- Provides a 32-bit Random Number Every 84 Clock Cycles

46.3 Block Diagram

Figure 46-1: TRNG Block Diagram



46.4 Product Dependencies

46.4.1 Power Management

The TRNG interface may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the TRNG user interface clock. The user interface clock is independent from any clock that may be used in the entropy source logic circuitry. The source of entropy can be enabled before enabling the user interface clock.

46.4.2 Interrupt Sources

The TRNG interface has an interrupt line connected to the Interrupt Controller. In order to handle interrupts, the Interrupt Controller must be programmed before configuring the TRNG.

Table 46-1: Peripheral IDs

Instance	ID
TRNG	30

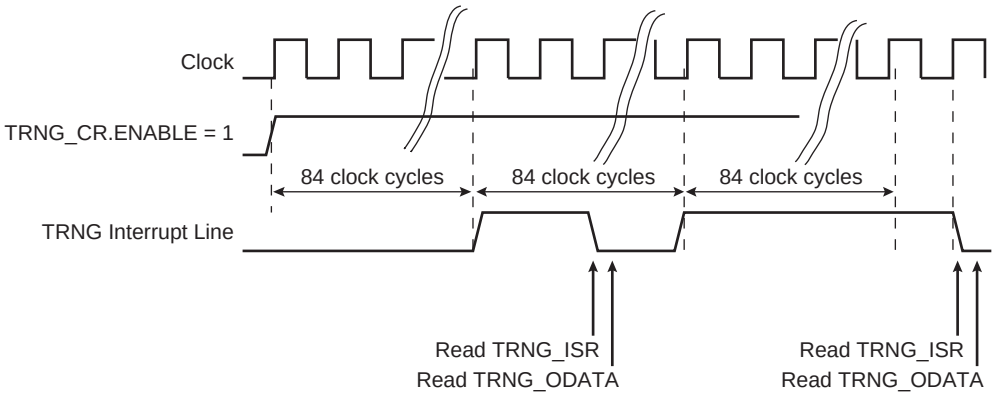
46.5 Functional Description

As soon as the TRNG is enabled in the control register (TRNG_CR), the generator provides one 32-bit value every 84 clock cycles. The TRNG interrupt line can be enabled in the TRNG_IER (respectively disabled in the TRNG_IDR). This interrupt is set when a new random value is available and is cleared when the status register (TRNG_ISR) is read. The flag DATRDY of the (TRNG_ISR) is set when the random data is ready to be read out on the 32-bit output data register (TRNG_ODATA).

The normal mode of operation checks that the status register flag equals 1 before reading the output data register when a 32-bit random value is required by the software application.

© 2017 Microchip Technology Inc.

Figure 46-2: TRNG Data Generation Sequence



© 2017 Microchip Technology Inc.

46.6 True Random Number Generator (TRNG) User Interface

Table 46-2: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	TRNG_CR	Write-only	–
0x04–0x0C	Reserved	–	–	–
0x10	Interrupt Enable Register	TRNG_IER	Write-only	–
0x14	Interrupt Disable Register	TRNG_IDR	Write-only	–
0x18	Interrupt Mask Register	TRNG_IMR	Read-only	0x0000_0000
0x1C	Interrupt Status Register	TRNG_ISR	Read-only	0x0000_0000
0x20–0x4C	Reserved	–	–	–
0x50	Output Data Register	TRNG_ODATA	Read-only	0x0000_0000
0x54–0xFC	Reserved	–	–	–

46.6.1 TRNG Control Register

Name:TRNG_CR

Address:0xF8048000

Access: Write-only

31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	ENABLE

ENABLE: Enables the TRNG to Provide Random Values

0: Disables the TRNG.

1: Enables the TRNG if 0x524E47 (“RNG” in ASCII) is written in KEY field at the same time.

KEY: Security Key

Value	Name	Description
0x524E47	PASSWD	Writing any other value in this field aborts the write operation.

© 2017 Microchip Technology Inc.

46.6.2 TRNG Interrupt Enable Register

Name: TRNG_IER

Address:0xF8048010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

DATRDY: Data Ready Interrupt Enable

0: No effect.

1: Enables the corresponding interrupt.

46.6.3 TRNG Interrupt Disable Register

Name: TRNG_IDR

Address:0xF8048014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

DATRDY: Data Ready Interrupt Disable

0: No effect.

1: Disables the corresponding interrupt.

© 2017 Microchip Technology Inc.

46.6.4 TRNG Interrupt Mask Register

Name: TRNG_IMR

Address:0xF8048018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

DATRDY: Data Ready Interrupt Mask

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

46.6.5 TRNG Interrupt Status Register

Name: TRNG_ISR

Address:0xF804801C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
		–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

DATRDY: Data Ready

0: Output data is not valid or TRNG is disabled.

1: New random value is completed.

DATRDY is cleared when this register is read.

© 2017 Microchip Technology Inc.

46.6.6 TRNG Output Data Register

Name: TRNG_ODATA

Address:0xF8048050

Access: Read-only

31	30	29	28	27	26	25	24
ODATA							
23	22	21	20	19	18	17	16
ODATA							
15	14	13	12	11	10	9	8
ODATA							
7	6	5	4	3	2	1	0
ODATA							

ODATA: Output Data

The 32-bit Output Data register contains the 32-bit random data.

47. Electrical Characteristics

47.1 Absolute Maximum Ratings

Table 47-1: Absolute Maximum Ratings*

Junction Temperature.....125°C
Storage Temperature.....-60°C to +150°C
Voltage on Input Pins with Respect to Ground....-0.3V to VDDIO + 0.3V (+4V max)
Maximum Operating Voltage (VDDCORE and VDDPLL) 1.2V
(VDDIOM, VDDIOPx, VDDOSC, VDDANA, VDDNF, VDDUSB, VDDFUSE, and VDDBU) 4.0V
Total DC Output Current on all I/O lines.....350 mA

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

47.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to +85°C, unless otherwise specified.

Table 47-2: DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T_A	Operating Temperature (Industrial)		-40	–	+85	°C
V_{DDCORE}	DC Supply Core		0.9	1.0	1.1	V
V_{DDBU}	DC Supply Backup		1.8	–	3.6	V
V_{DDPLL}	DC Supply PLL		0.9	1.0	1.1	V
V_{DDOSC}	DC Supply Oscillator		1.65	–	3.6	V
V_{DDIOM}	DC Supply EBI I/Os		1.65/3.0	1.8/3.3	1.95/3.6	V
V_{DDNF}	DC Supply NAND Flash I/Os		1.65/3.0	1.8/3.3	1.95/3.6	V
V_{DDIOP0}	DC Supply Peripheral I/Os		1.65	–	3.6	V
V_{DDIOP1}	DC Supply Peripheral I/Os		1.65	–	3.6	V
V_{DDANA}	DC Supply Analog		3.0	3.3	3.6	V
V_{DDUSB}	DC Supply USB		3.0	3.3	3.6	V
V_{DDFUSE}	DC Supply Fuse Box		3.0	–	3.6	V
V_{IL}	Input Low-level Voltage	V_{DDIO} 3.0–3.6V	-0.3	–	0.8	V
		V_{DDIO} 1.65–1.95V	-0.3	–	$0.3 \times V_{DDIO}$	V
V_{IH}	Input High-level Voltage	V_{DDIO} 3.0–3.6V	2	–	V_{DDIO} + 0.3	V
		V_{DDIO} 1.65–1.95V	$0.7 \times V_{DDIO}$	–	V_{DDIO} + 0.3	V

© 2017 Microchip Technology Inc.

Table 47-2: DC Characteristics (Continued)

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{OL}	Output Low-level Voltage	I_O Max, V_{DDIO} 3.0–3.6V		–	–	0.4	V
		CMOS (I_O < 0.3 mA), V_{DDIO} 1.65–1.95V		–	–	0.1	V
		TTL (I_O Max), V_{DDIO} 1.65–1.95V		–	–	0.4	V
V_{OH}	Output High-level Voltage	I_O Max, V_{DDIO} 3.0–3.6V		V_{DDIO} - 0.4	–	–	V
		CMOS (I_O < 0.3 mA), V_{DDIO} 1.65–1.95V		V_{DDIO} - 0.1	–	–	V
		TTL (I_O Max), V_{DDIO} 1.65–1.95V		V_{DDIO} - 0.4	–	–	V
V_{T-}	Schmitt trigger Negative going threshold Voltage	I_O Max, V_{DDIO} 3.0–3.6V		0.8	1.1	–	V
		TTL (I_O Max), V_{DDIO} 1.65–1.95V		–	–	$0.3 \times V_{DDIO}$	V
V_{T+}	Schmitt trigger Positive going threshold Voltage	I_O Max, V_{DDIO} 3.0–3.6V		–	1.6	2.0	V
		TTL (I_O Max), V_{DDIO} 1.65–1.95V		$0.3 \times V_{DDIO}$	–	–	V
V_{hys}	Schmitt trigger Hysteresis	V_{DDIO} 3.0–3.6V		0.5	–	0.75	V
		V_{DDIO} 1.65–1.95V		0.28	–	0.6	V
R_{PULLUP}	Pull-up Resistance	PA0–PA31 PB0–PB31 PD0–PD31 PE0–PE31 NTRST and NRST		40	75	190	kΩ
		PC0–PC31 V_{DDIOM1} in 1.8V range		240	–	1000	
		PC0–PC31 V_{DDIOM1} in 3.3V range		120	–	350	
I_O	Output Current	PA0–PA31 PB0–PB31 PD0–PD31 PE0–PE31		–	–	8	mA
		PC0–PC31 V_{DDIOM1} in 1.8V range		–	–	2	
		PC0–PC31 V_{DDIOM1} in 3.3V range		–	–	4	
I_{SC}	Static Current	On V_{DDCORE} = 1.0V, MCK = 0 Hz, excluding POR	T_A = 25°C	–	11	–	mA
		All inputs driven TMS, TDI, TCK, NRST = 1	T_A = 85°C	–	–	25	
		On V_{DDBU} = 3.3V, Logic cells consumption, excluding POR	T_A = 25°C	–	8	–	µA
		All inputs driven WKUP = 0	T_A = 85°C	–	–	15	

© 2017 Microchip Technology Inc.

47.3 Power Consumption

- Typical power consumption of PLLs, Slow Clock and Main Oscillator
- Power consumption of power supply in four different modes: Active, Idle, Ultra Low-power and Backup
- Power consumption by peripheral: calculated as the difference in current measurement after having enabled then disabled the corresponding clock

47.3.1 Power Consumption versus Modes

The values in Table 47-3 and Table 47-4 represent the power consumption estimated on the power supplies with operating conditions as follows:

- V_{DDIOM} = 1.8V
- V_{DDIOP0} and V_{DDIOP1} = 3.3V
- V_{DDPLL} = 1.0V
- V_{DDCORE} = 1.0V
- V_{DDBU} = 3.3V
- T_A = 25 °C
- There is no consumption on the I/Os of the device

Figure 47-1: Measures Schematics

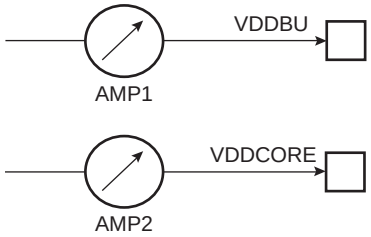


Table 47-3: Power Consumption for Different Modes

Mode	Conditions	Consumption	Unit
Active	Arm Core clock is 400 MHz. MCK is 133 MHz. All peripheral clocks activated. onto AMP2	103	mA
Idle	Idle state, waiting an interrupt. All peripheral clocks de-activated. onto AMP2	33	mA
Ultra Low-power	Arm Core clock is 500 Hz. All peripheral clocks de-activated. onto AMP2	7	mA
Backup	Device only V_{DDBU} powered onto AMP1	8	µA

© 2017 Microchip Technology Inc.

Table 47-4: Power Consumption by Peripheral in Active Mode

Peripheral	Consumption	Unit
PIO Controller	6	µA/MHz
USART	6	
ADC	5	
TWI	2	
SPI	3	
UART	3	
UHP	5	
UDP	5	
LCDC	3	
PWM	6	
HSMCI	3	
SSC	5	
Timer Counter Channels	12	
DMA	1	
AES	4	
SHA	3	
TRNG	1	

47.4 Clock Characteristics

47.4.1 Processor Clock

Table 47-5: Processor Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPPCK})$	Processor Clock Frequency	V_{DDCORE} min	250(1)	400	MHz

Note 1: With DDR2 usage. There are no limitations for LPDDR, SDRAM and mobile SDRAM.

47.4.2 System Clock

The system clock is the maximum clock at which the system is able to run. It is given by the smallest value of the internal bus clock and EBI clock.

Table 47-6: System Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPMCK})$	System Clock Frequency	V_{DDCORE} min	125(1)	133	MHz

Note 1: With DDR2 usage. There are no limitations for LPDDR, SDRAM and mobile SDRAM.

© 2017 Microchip Technology Inc.

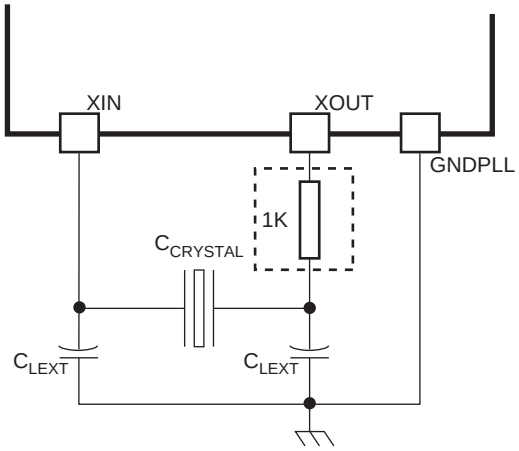
47.4.3 Main Oscillator Characteristics

Table 47-7: Main Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPMAIN})$	Crystal Oscillator Frequency		8	16	20	MHz
$C_{CRYSTAL}$ (1)	Crystal Load Capacitance		12.5	–	17.5	pF
C_{INT} (1)	Internal Load Capacitance		1.85	2.1	2.35	pF
C_{LEXT}	External Load Capacitance	$C_{CRYSTAL}$ = 12.5 pF (1)	–	20.8	–	pF
		$C_{CRYSTAL}$ = 17.5 pF (1)	–	30.8	–	pF
	Duty Cycle		–	–	–	%
t_{START}	Startup Time	@ 3 MHz @ 8 MHz @ 16 MHz @ 20 MHz	–	–	20 4 2 2	ms
I_{DDST}	Standby Current Consumption	Standby mode	–	–	1	µA
P_{ON}	Drive Level	@ 3 MHz @ 8 MHz @ 16 MHz @ 20 MHz	–	–	15 30 50 50	µW
$I_{DD\ ON}$	Current Dissipation	@ 3 MHz @ 8 MHz @ 16 MHz @ 20 MHz	–	280 380 500 580	380 510 630 750	µA

Note 1: The $C_{CRYSTAL}$ value is specified by the crystal manufacturer. In our case, $C_{CRYSTAL}$ must be between 12.5 pF and 17.5 pF. All parasitic capacitance, package and board, **must be calculated** in order to reach 12.5 pF (minimum targeted load for the oscillator) by taking into account the internal load C_{INT}. So, to target the minimum oscillator load of 12.5 pF, external capacitance must be 12.5 pF - 2.1 pF = 10.4 pF, which means that 20.8 pF is the target value (20.8 pF from XIN to GND and 20.8 pF from XOUT to GND). If 17.5 pF load is targeted, the sum of pad, package, board and external capacitances must be 17.5 pF - 2.1 pF = 15.4 pF, which means 30.8 pF (30.8 pF from XIN to GND and 30.8 pF from XOUT to GND).

Figure 47-2: Main Oscillator Schematic



© 2017 Microchip Technology Inc.

47.4.4 Crystal Oscillator Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

Table 47-8: Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs	Fundamental @ 3 MHz Fundamental @ 8 MHz Fundamental @ 16 MHz Fundamental @ 20 MHz	–	–	200 100 80 50	Ω
C_m	Motional Capacitance		–	–	8	fF
C_{SHUNT}	Shunt Capacitance		–	–	7	pF

47.4.5 XIN Clock Characteristics

Table 47-9: XIN Clock Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
1/(t_{CPXIN})	XIN Clock Frequency		–	50	MHz
t_{CPXIN}	XIN Clock Period		20	–	ns
t_{CHXIN}	XIN Clock High Half-period		$0.4 \times t_{CPXIN}$	$0.6 \times t_{CPXIN}$	ns
t_{CLXIN}	XIN Clock Low Half-period		$0.4 \times t_{CPXIN}$	$0.6 \times t_{CPXIN}$	ns
C_{IN}	XIN Input Capacitance	Main Oscillator in Bypass mode, i.e., when CKGR_MOR.MOSCEN = 0 and CKGR_MOR.OSCBYPASS = 1 (see Section 21.13.7 "PMC Clock Generator Main Oscillator Register")	–	25	pF
R_{IN}	XIN Pull-down Resistor		–	500	kΩ
V_{XINLOW}	XIN Low Voltage		-0.3V	$0.3 \times V_{DDOSC}$(1)	V
$V_{XINHIGH}$	XIN High Voltage		$0.7 \times V_{DDOSC}$	V_{DDOSC} + 0.3(1)	V

Note 1: Do not exceed 3.6V

47.5 12 MHz RC Oscillator Characteristics

Table 47-10: 12 MHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
F0	Nominal Frequency		8.4	12	15.6	MHz
Duty	Duty Cycle		45	50	55	%
$I_{DD\ ON}$	Power Consumption Oscillation		86 86	–	140 125	µA
t_{START}	Startup time		6	–	10	µs
$I_{DD\ STDBY}$	Standby consumption		–	–	22	µA

47.6 32 kHz Oscillator Characteristics

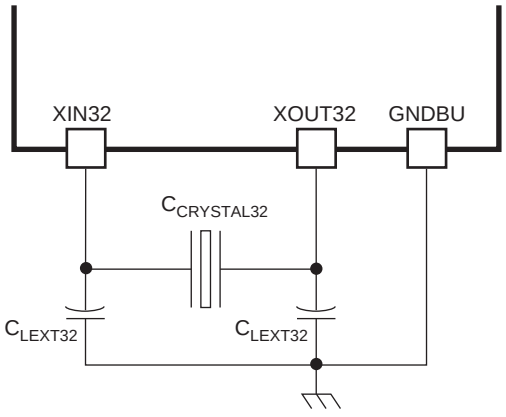
Table 47-11: 32 kHz Oscillator Characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
1/($t_{CP32KHz}$)	Crystal Oscillator Frequency			–	32.768	–	kHz
$C_{CRYSTAL32}$	Load Capacitance	Crystal @ 32.768 kHz		6	–	12.5	pF
C_{LEXT32}[2]	External Load Capacitance	$C_{CRYSTAL32}$ = 6 pF		–	6	–	pF
		$C_{CRYSTAL32}$ = 12.5 pF		–	19	–	pF
	Duty Cycle			40	50	60	%
t_{START}	Startup Time	R_S = 50 kΩ[1]	$C_{CRYSTAL32}$ = 6 pF	–	–	400	ms
			$C_{CRYSTAL32}$ = 12.5 pF	–	–	900	ms
		R_S = 100 kΩ[1]	$C_{CRYSTAL32}$ = 6 pF	–	–	600	ms
			$C_{CRYSTAL32}$ = 12.5 pF	–	–	1200	ms

Note 1: R_S is the equivalent series resistance.

2: C_{LEXT32} is determined by taking into account internal, parasitic and package load capacitance.

Figure 47-3: 32 kHz Oscillator Schematic



© 2017 Microchip Technology Inc.

47.6.1 32 kHz Crystal Characteristics

Table 47-12: 32 kHz Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor R_S	Crystal @ 32.768 kHz	–	50	100	kΩ
C_m	Motional Capacitance	Crystal @ 32.768 kHz	0.6	–	3	fF
C_{SHUNT}	Shunt Capacitance	Crystal @ 32.768 kHz	0.6	–	2	pF
$I_{DD\ ON}$	Current dissipation	R_S = 50 kΩ(1) $C_{CRYSTAL32}$ = 6 pF	–	0.55	1.3	µA
		R_S = 50 kΩ(1) $C_{CRYSTAL32}$ = 12.5pF	–	0.85	1.6	µA
		R_S = 100 kΩ(1) $C_{CRYSTAL32}$ = 6 pF	–	0.7	2.0	µA
		R_S = 100 kΩ(1) $C_{CRYSTAL32}$ = 12.5 pF	–	1.1	2.2	µA
$I_{DD\ STDBY}$	Standby consumption		–	–	0.3	µA

Note 1: R_S is the equivalent series resistance.

47.6.2 XIN32 Clock Characteristics

Table 47-13: XIN32 Clock Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
1/($t_{CPXIN32}$)	XIN32 Clock Frequency		–	44	kHz
$t_{CPXIN32}$	XIN32 Clock Period		22	–	µs
$t_{CHXIN32}$	XIN32 Clock High Half-period		11	–	µs
$t_{CLXIN32}$	XIN32 Clock Low Half-period		11	–	µs
t_{CLCH32}	XIN32 Clock Rise time		400	–	ns
t_{CLCL32}	XIN32 Clock Fall time		400	–	ns
C_{IN32}	XIN32 Input Capacitance	32.768 kHz Oscillator in Bypass mode, i.e., when RCEN = 0, OSC32EN = 0, OSC32BYP = 1 and OSCSEL = 1 in "Slow Clock Controller Configuration Register" (SCKC_CR) . See Section 19.4.2 "Bypassing the 32.768 kHz Crystal Oscillator".	–	6	pF
R_{IN32}	XIN32 Pull-down Resistor		–	4	MΩ
V_{IN32}	XIN32 Voltage		V_{DDBU}	V_{DDBU}	V
V_{INIL32}	XIN32 Input Low Level Voltage		-0.3	$0.3 \times V_{DDBU}$	V
V_{INIH32}	XIN32 Input High Level Voltage		$0.7 \times V_{DDBU}$	V_{DDBU} + 0.3	V

47.7 32 kHz RC Oscillator Characteristics

Table 47-14: 32 kHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
1/(t_{CPRCz})	Crystal Oscillator Frequency		20	32	44	kHz
	Duty Cycle		45	–	55	%
t_{START}	Startup Time		–	–	75	µs
$I_{DD\ ON}$	Power Consumption Oscillation	After startup time	–	1.1	2.1	µA
$I_{DD\ STDBY}$	Standby consumption		–	–	0.4	µA

© 2017 Microchip Technology Inc.

47.8 PLL Characteristics

Table 47-15: PLLA Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output Frequency	Refer to Table 47-16	400	–	800	MHz
f_{IN}	Input Frequency		2	–	32	MHz
I_{PLL}	Current Consumption	Active mode	–	3.6	4.5	mA
		Standby mode	–	–	1	µA
t_{START}	Startup Time		–	–	50	µs

PMC_PLLICPR.ICPLLA and CKGR_PLLAR.OUTA must be configured for each PLLA frequency range as shown in Table 47-16.

Table 47-16: PLLA Frequency Configuration with PMC_PLLICPR.ICPLLA and CKGR_PLLAR.OUTA

PLL Frequency Range (MHz)	ICPLLA	OUTA	
745–800	0	0	0
695–750	0	0	1
645–700	0	1	0
595–650	0	1	1
545–600	1	0	0
495–550	1	0	1
445–500	1	1	0
400–450	1	1	1

Table 47-17: PLLB Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output Frequency	Field CKGR_PLLBR.OUTB = 00	30	–	100	MHz
f_{IN}	Input Frequency		2	–	32	MHz
I_{PLL}	Current Consumption	Active mode @ 100 MHz	–	–	1.2	mA
		Standby mode	–	–	1	µA
t_{START}	Startup Time		–	–	100	µs

47.9 I/Os

Criteria used to define the maximum frequency of the I/Os:

- Output duty cycle (40%–60%)
- Minimum output swing: 100 mV to VDDIO - 100 mV

© 2017 Microchip Technology Inc.

- Addition of rising and falling time inferior to 75% of the period

Table 47-18: I/O Characteristics

Symbol	Parameter	Conditions		Min	Max	Unit
FreqMax	VDDIOP powered pins frequency	3.3V domain	CCFG_EBICSA.EBI_DRIVE = LOW, C_{LOAD} = 20 pF(1)	–	66	MHz
			CCFG_EBICSA.EBI_DRIVE = HIGH, C_{LOAD} = 40 pF(1)	–	66	
		1.8V domain	CCFG_EBICSA.EBI_DRIVE = LOW, C_{LOAD} = 20 pF(1)	–	66	MHz
			CCFG_EBICSA.EBI_DRIVE = HIGH, C_{LOAD} = 30 pF(1)	–	66	

Note 1: C_{LOAD} = maximum external capacitance

47.10 Analog-to-Digital Converter (ADC)

Table 47-19: Channel Conversion Time and ADC Clock

Parameter	Conditions	Min	Typ	Max	Unit
ADC Clock Frequency	10-bit resolution mode	–	–	13.2	MHz
Startup Time	Return from Idle Mode	–	–	40	µs
Track and Hold Acquisition Time (TTH)	ADC Clock = 13.2 MHz(1)	0.5	–		µs
Conversion Time (TCT)	ADC Clock = 13.2 MHz(1) ADC Clock = 5 MHz(1)	–	–	1.74 4.6	µs
Throughput Rate	ADC Clock = 13.2 MHz(1) ADC Clock = 5 MHz(1)	–	–	440 192	ksps

Note 1: The Track-and-Hold Acquisition Time is given by: TTH (ns) = 500 + (0.12 × Z_{IN}) (Ω)
The ADC internal clock is divided by 2 in order to generate a clock with a duty cycle of 75%. So the maximum conversion time is give by:

$$TCT(\mu s) = \frac{23}{fclk}(MHz)$$

The full speed is obtained for an input source impedance of < 50 Ω maximum, or TTH = 500 ns.
In order to make the ADC work properly, the TRACKTIM field in the ADC Mode Register is to be calculated according to this Track and Hold Acquisition Time, also called Sampled and Hold Time.

Table 47-20: External Voltage Reference Input

Parameter	Conditions	Min	Typ	Max	Unit
ADVREF Input Voltage Range		2.4	–	VDDANA	V
ADVREF Average Current		–	–	600	µA
Current Consumption on VDDANA		–	–	600	µA

© 2017 Microchip Technology Inc.

Table 47-21: Analog Inputs

Parameter	Conditions	Min	Typ	Max	Unit
Input Voltage Range		0	–	ADVREF	V
Input Peak Current		–	–	2.5	mA
Input Capacitance		–	7	10	pF
Input Impedance		–	50		Ω

Table 47-22: Transfer Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
	Resolution		–	10	–	bit
INL	Integral Non-linearity		–	–	±2	LSB
DNL	Differential Non-linearity	ADC Clock = 13.2 MHz ADC Clock = 5 MHz	–	–	±2 ±0.9	LSB
E_O	Offset Error		–	–	±10	LSB
E_G	Gain Error	ADC Clock = 13.2 MHz ADC Clock = 5 MHz	–	–	±3 ±2	LSB

47.11 USB Transceiver Characteristics

Table 47-23: USB Electrical Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Input Levels						
V_{IL}	Low Level		–	–	0.8	V
V_{IH}	High Level		2.0	–	–	V
V_{DI}	Differential Input Sensitivity	|(D+) - (D-)|	0.2	–	–	V
V_{CM}	Differential Input Common Mode Range		0.8	–	2.5	V
C_{IN}	Transceiver capacitance	Capacitance to ground on each line	–	–	9.18	pF
I_{lkg}	Hi-Z State Data Line Leakage	$0V < V_{IN} < 3.3V$	- 10	–	+ 10	µA
R_{EXT}	Recommended External USB Series Resistor	In series with each USB pin with ±5%	–	27	–	Ω
Output Levels						
V_{OL}	Low Level Output	Measured with R_L of 1.425 kΩ tied to 3.6V	0.0	–	0.3	V
V_{OH}	High Level Output	Measured with R_L of 14.25 kΩ tied to GND	2.8	–	3.6	V
V_{CRS}	Output Signal Crossover Voltage	Measure conditions described in Figure 47-22	1.3	–	2.0	V
Pull-up and Pull-down Resistor						
R_{PUI}	Bus Pull-up Resistor on Upstream Port (idle bus)		0.900	–	1.575	kΩ
R_{PUA}	Bus Pull-up Resistor on Upstream Port (upstream port receiving)		1.425	–	3.090	kΩ
R_{PD}	Bus Pull-down resistor		14.25	–	24.8	kΩ

© 2017 Microchip Technology Inc.

47.12 Core Power Supply POR Characteristics

Table 47-24: Power-On-Reset Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{T+}	Threshold Voltage Rising	Minimum Slope of +2.0V/30ms	0.5	0.7	0.89	V
V_{T-}	Threshold Voltage Falling		0.4	0.6	0.85	V
t_{RST}	Reset Time		30	70	130	µs
I_{DD}	Current consumption	After t_{RST}	–	3	7	µA

47.13 SMC Timings

47.13.1 Timing Conditions

Timings are given assuming a capacitance load on data, control and address pads.

Table 47-25: Capacitance Load

	Corner	
Supply	Max	Min
3.3V	50 pF	5 pF
1.8V	30 pF	5 pF

In the following tables, t_{CPMCK} is MCK period.

47.13.2 Timing Extraction

47.13.2.1 Zero Hold Mode Restrictions

Table 47-26: Zero Hold Mode Use Maximum System Clock Frequency (MCK)

		Min		
Symbol	Parameter	1.8V VDDIOM Supply	3.3V VDDIOM Supply	Unit
f_{max}	MCK frequency	66	66	MHz

47.13.2.2 Read Timings

Table 47-27: SMC Read Signals - NRD Controlled (READ_MODE = 1)

Symbol	Parameter	Min		Unit
		1.8V VDDIOM Supply	3.3V VDDIOM Supply	
NO HOLD SETTINGS (nrd hold = 0)				
SMC_1	Data Setup before NRD High	13.7	11.8	ns
SMC_2	Data Hold after NRD High	0	0	ns
HOLD SETTINGS (nrd hold ≠ 0)				
SMC_3	Data Setup before NRD High	10.7	8.8	ns
SMC_4	Data Hold after NRD High	0	0	ns
HOLD or NO HOLD SETTINGS (nrd hold ≠ 0, nrd hold = 0)				
SMC_5	NBS0/A0, NBS1, NBS2/A1, NBS3, A2–A25 Valid before NRD High	(nrd setup + nrd pulse) × t_{CPMCK} - 5.3	(nrd setup + nrd pulse) × t_{CPMCK} - 5.1	ns
SMC_6	NCS low before NRD High	(nrd setup + nrd pulse - ncs rd setup) × t_{CPMCK} -4.8	(nrd setup + nrd pulse - ncs rd setup) × t_{CPMCK} - 4.9	ns
SMC_7	NRD Pulse Width	nrd pulse × t_{CPMCK} - 3.4	nrd pulse × t_{CPMCK} - 3.5	ns

47.13.2.3 Write Timings

Table 47-28: SMC Read Signals - NCS Controlled (READ_MODE = 0)

Symbol	Parameter	Min		Unit
		1.8V VDDIOM Supply	3.3V VDDIOM Supply	
NO HOLD SETTINGS (ncs rd hold = 0)				
SMC_8	Data Setup before NCS High	26.7	24.7	ns
SMC_9	Data Hold after NCS High	0	0	ns
HOLD SETTINGS (ncs rd hold ≠ 0)				
SMC_{10}	Data Setup before NCS High	12.4	10.4	ns
SMC_{11}	Data Hold after NCS High	0	0	ns
HOLD or NO HOLD SETTINGS (ncs rd hold ≠ 0, ncs rd hold = 0)				
SMC_{12}	NBS0/A0, NBS1, NBS2/A1, NBS3, A2–A25 valid before NCS High	(ncs rd setup + ncs rd pulse) × t_{CPMCK} - 18.1	(ncs rd setup + ncs rd pulse) × t_{CPMCK} - 18.2	ns
SMC_{13}	NRD low before NCS High	(ncs rd setup + ncs rd pulse - nrd setup) × t_{CPMCK} - 2.8	(ncs rd setup + ncs rd pulse - nrd setup) × t_{CPMCK} - 2.9	ns
SMC_{14}	NCS Pulse Width	ncs rd pulse length × t_{CPMCK} - 4.0	ncs rd pulse length × t_{CPMCK} - 4.0	ns

© 2017 Microchip Technology Inc.

Table 47-29: SMC Write Signals - NWE Controlled (WRITE_MODE = 1)

Symbol	Parameter	Min		Unit
		1.8V VDDIOM Supply	3.3V VDDIOM Supply	
HOLD or NO HOLD SETTINGS (nwe hold ≠ 0, nwe hold = 0)				
SMC_{15}	Data Out Valid before NWE High	nwe pulse × t_{CPMCK} - 4.1	nwe pulse × t_{CPMCK} - 4.0	ns
SMC_{16}	NWE Pulse Width	nwe pulse × t_{CPMCK} - 3.0	nwe pulse × t_{CPMCK} - 3.1	ns
SMC_{17}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 valid before NWE low	nwe setup × t_{CPMCK} - 4.2	nwe setup × t_{CPMCK} - 4.1	ns
SMC_{18}	NCS low before NWE high	(nwe setup - ncs rd setup + nwe pulse) × t_{CPMCK} - 3.8	(nwe setup - ncs rd setup + nwe pulse) × t_{CPMCK} - 3.7	ns
HOLD SETTINGS (nwe hold ≠ 0)				
SMC_{19}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 change	nwe hold × t_{CPMCK} - 4.0	nwe hold × t_{CPMCK} - 3.1	ns
SMC_{20}	NWE High to NCS Inactive (1)	(nwe hold - ncs wr hold) × t_{CPMCK} - 2.8	(nwe hold - ncs wr hold) × t_{CPMCK} - 2.0	ns
NO HOLD SETTINGS (nwe hold = 0)				
SMC_{21}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25, NCS change(1)	1.6	1.4	ns

Note 1: hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs wr hold length” or “NWE hold length”.

Table 47-30: SMC Write NCS Controlled (WRITE_MODE = 0)

Symbol	Parameter	Min		Unit
		1.8V VDDIOM Supply	3.3V VDDIOM Supply	
SMC_{22}	Data Out Valid before NCS High	ncs wr pulse × t_{CPMCK} - 4.3	ncs wr pulse × t_{CPMCK} - 4.5	ns
SMC_{23}	NCS Pulse Width	ncs wr pulse × t_{CPMCK} - 4.0	ncs wr pulse × t_{CPMCK} - 4.0	ns
SMC_{24}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 valid before NCS low	ncs wr setup × t_{CPMCK} - 3.6	ncs wr setup × t_{CPMCK} - 3.5	ns
SMC_{25}	NWE low before NCS high	(ncs wr setup - nwe setup + ncs pulse) × t_{CPMCK} - 3.9	(ncs wr setup - nwe setup + ncs pulse) × t_{CPMCK} - 3.9	ns
SMC_{26}	NCS High to Data Out, NBS0/A0, NBS1, NBS2/A1, NBS3, A2–A25 change	ncs wr hold × t_{CPMCK} - 6.1	ncs wr hold × t_{CPMCK} - 5.2	ns
SMC_{27}	NCS High to NWE Inactive	(ncs wr hold - nwe hold) × t_{CPMCK} - 4.8	(ncs wr hold - nwe hold) × t_{CPMCK} - 4.4	ns

© 2017 Microchip Technology Inc.

Figure 47-4: SMC Timings - NCS Controlled Read and Write

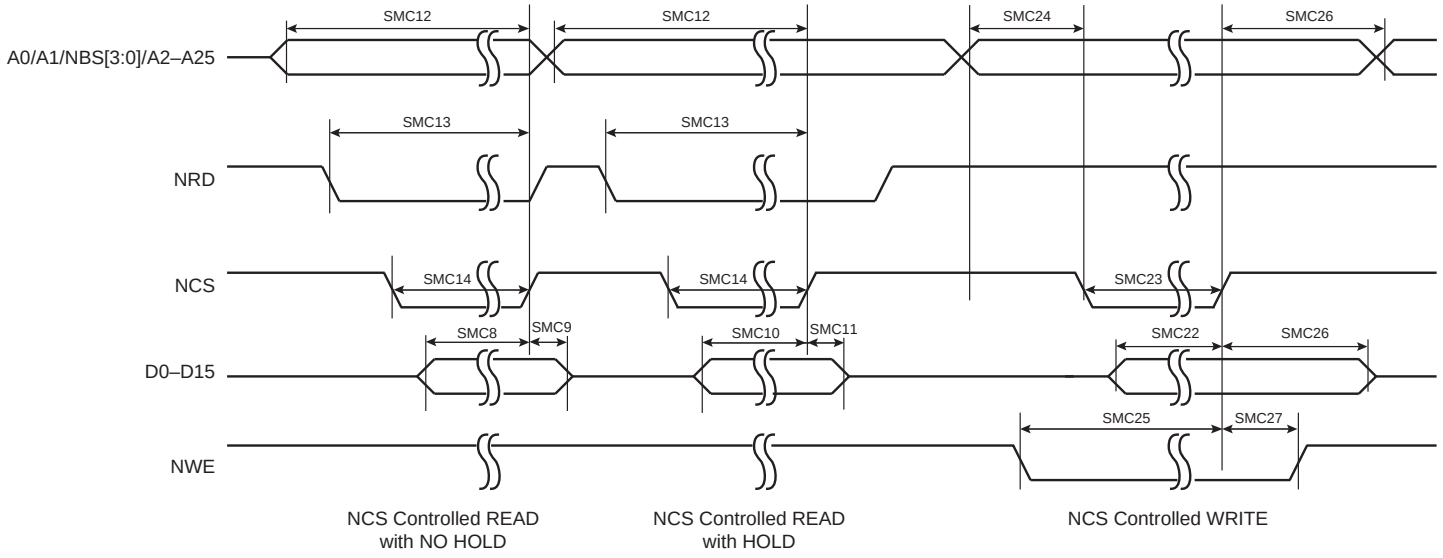
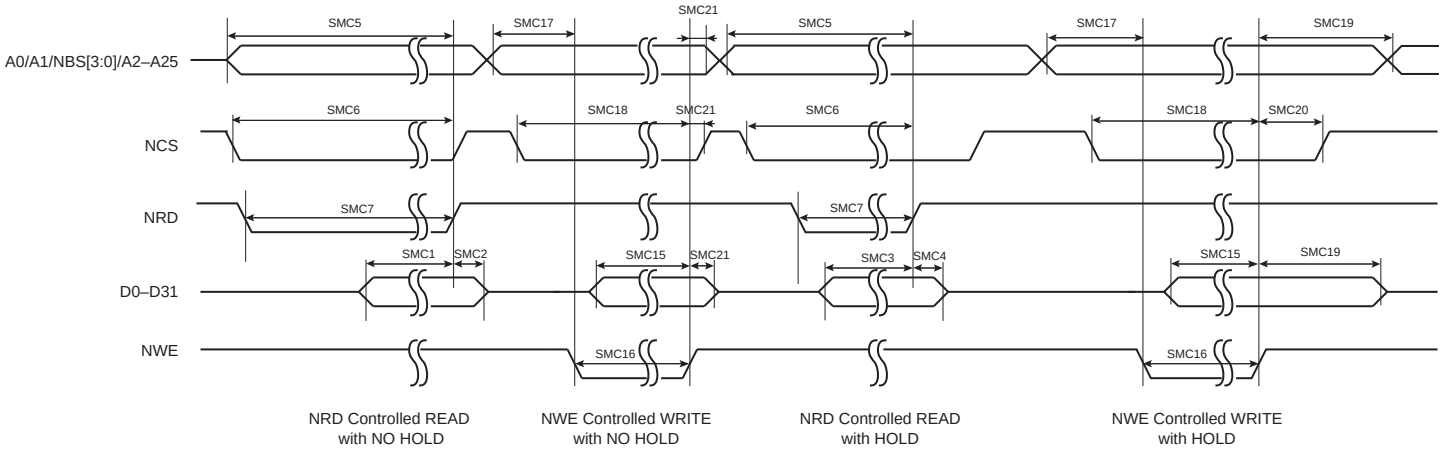


Figure 47-5: SMC Timings - NRD Controlled Read and NWE Controlled Write



© 2017 Microchip Technology Inc.

47.14 DDRSDRC Timings

The DDRSDRC controller satisfies the timings of standard DDR2, LP-DDR, SDR and LP-SDR modules.

DDR2, LP-DDR and SDR timings are specified by the JEDEC standard.

Supported speed grade limitations:

- DDR2-400 limited at 133 MHz clock frequency (1.8V, 30 pF on data/control, 10 pF on CK/CK#)
- LP-DDR (1.8V, 30 pF on data/control, 10pF on CK)
  t_{cyc} = 5.0 ns, f_{max} = 125 MHz
  t_{cyc} = 6.0 ns, f_{max} = 110 MHz
  t_{cyc} = 7.5 ns, f_{max} = 95 MHz
- SDR-100 (3.3V, 50 pF on data/control, 10 pF on CK)
- SDR-133 (3.3V, 50 pF on data/control, 10 pF on CK)
- LP-SDR-133 (1.8V, 30 pF on data/control, 10 pF on CK)

47.15 Peripheral Timings

47.15.1 SPI

47.15.1.1 Maximum SPI Frequency

The following formulas give maximum SPI frequency in Master read and write modes and in Slave read and write modes.

- Master Write Mode

  The SPI only sends data to a slave device such as an LCD, for example. The limit is given by SPI_2 (or SPI_5) timing. Since it gives a maximum frequency above the maximum pad speed (see Section 47.9 “I/Os”), the maximum SPI frequency is defined by the pin FreqMax value.

- Master Read Mode

  $$f_{SPCK}Max = \frac{1}{SPI_0(orSPI_3) + t_{valid}}$$

  t_{valid} is the slave time response to output data after deleting an SPCK edge. F or a non-volatile memory with t_{valid} (or t_v) = 12 ns Max, f_{SPCK}Max = 47.1 MHz @ V_{DDIO} = 3.3V.

- Slave Read Mode

  In slave mode, SPCK is the input clock for the SPI. The max SPCK frequency is given by setup and hold timings SPI_7/SPI_8(or SPI_{10}/SPI_{11}). Since this gives a frequency well above the pad limit, the limit in slave read mode is given by SPCK pad.

- Slave Write Mode

  $$f_{SPCK}Max = \frac{1}{SPI_6(orSPI_9) + t_{setup}}$$

  t_{setup} is the setup time from the master before sampling data (12 ns).

  This gives f_{SPCK}Max = 44.6 MHz @ V_{DDIO} = 3.3V.

47.15.1.2 Timing Conditions

Timings are given assuming a capacitance load on MISO, SPCK and MOSI.

Table 47-31: Capacitance Load for MISO, SPCK and MOSI (product dependent)

Supply	Corner	
	Max	Min
3.3V	40 pF	5 pF
1.8V	20 pF	5 pF

47.15.1.3 Timing Extraction

Figure 47-6: SPI Master mode 1 and 2

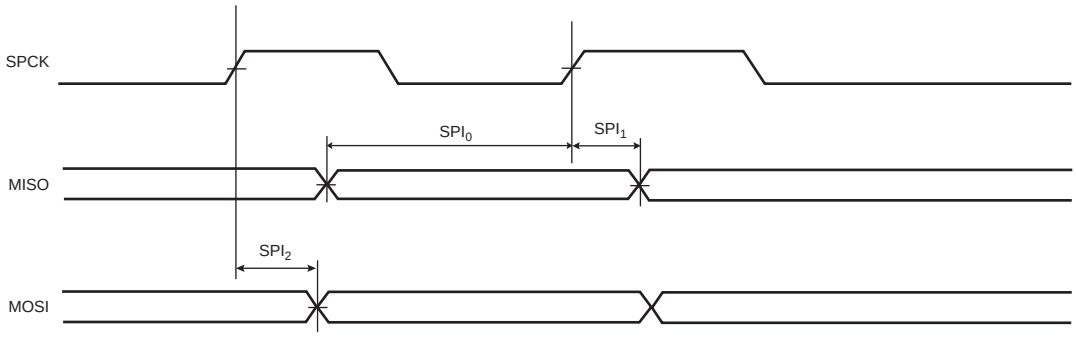


Figure 47-7: SPI Master mode 0 and 3

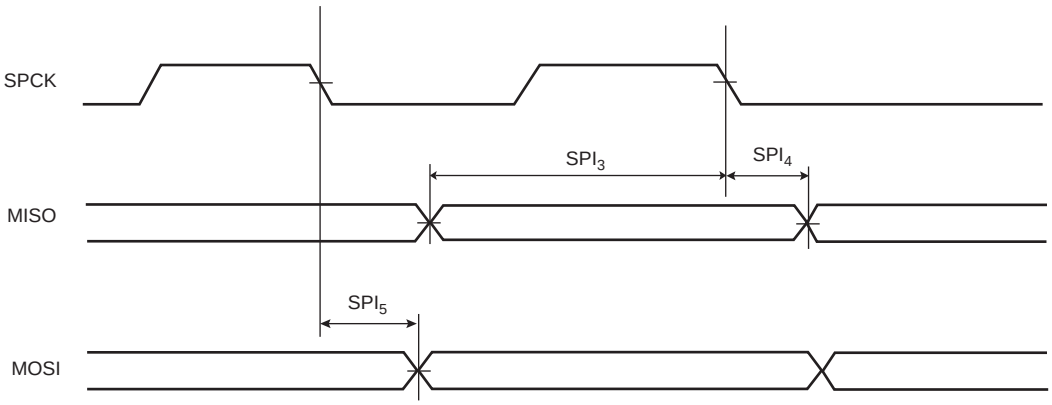
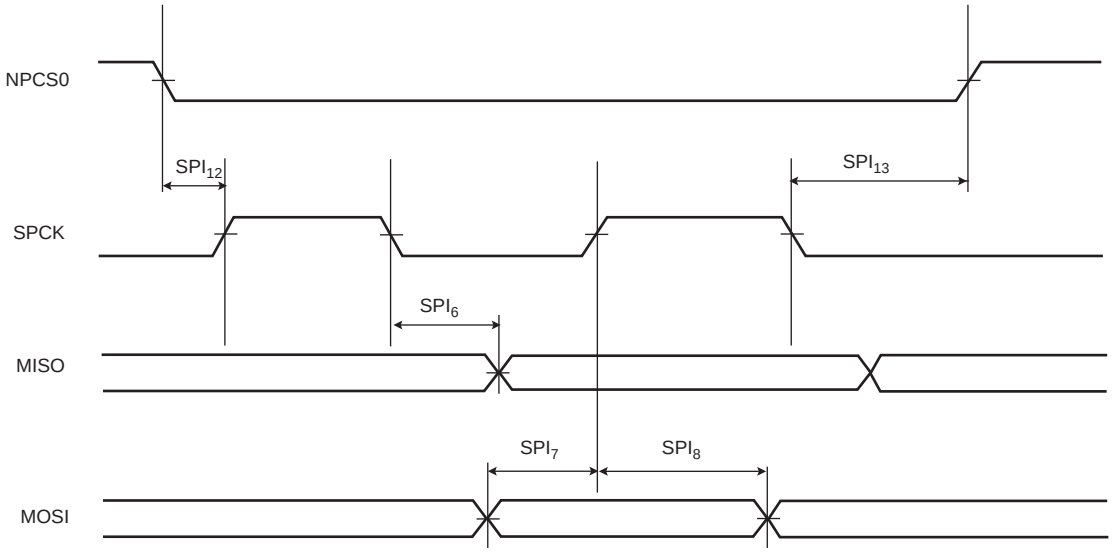


Figure 47-8: SPI Slave mode 0 and 3



© 2017 Microchip Technology Inc.

Figure 47-9: SPI Slave mode 1 and 2

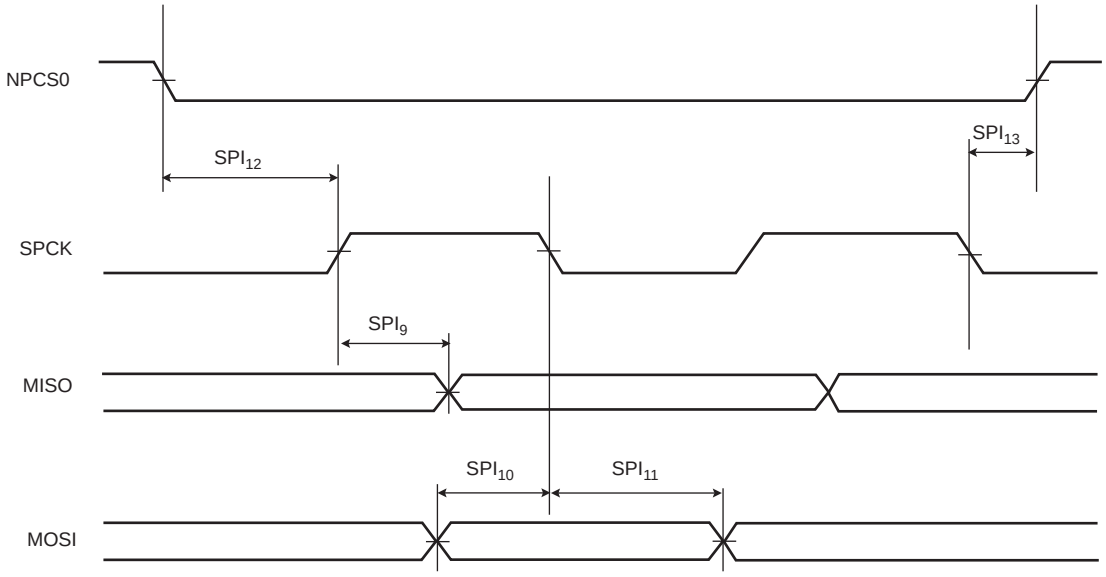


Figure 47-10: SPI Slave mode - NPCS timings

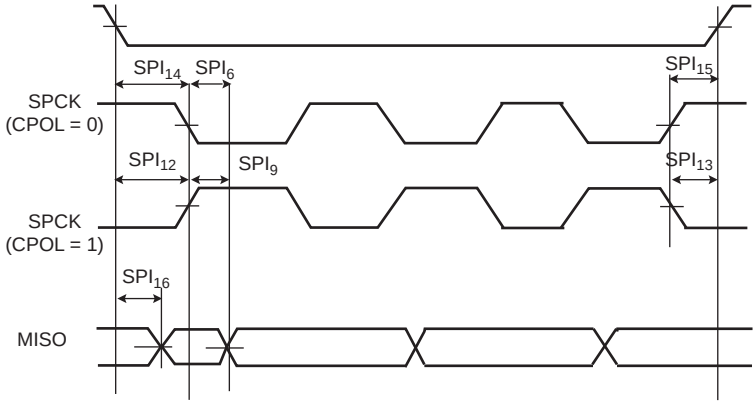


Table 47-32: SPI Timings with 3.3V Peripheral Supply

Symbol	Parameter	Conditions	Min	Max	Unit
Master Mode					
SPI_{SPCK}	SPI Clock		–	66	MHz
SPI_0	MISO Setup time before SPCK rises		13.7	–	ns
SPI_1	MISO Hold time after SPCK rises		0	–	ns
SPI_2	SPCK rising to MOSI		0	7.6	ns
SPI_3	MISO Setup time before SPCK falls		13.2	–	ns
SPI_4	MISO Hold time after SPCK falls		0	–	ns
SPI_5	SPCK falling to MOSI		0	7.7	ns
Slave Mode					
SPI_6	SPCK falling to MISO		2.7	14.1	ns

Table 47-32: SPI Timings with 3.3V Peripheral Supply (Continued)

Symbol	Parameter	Conditions	Min	Max	Unit
SPI_7	MOSI Setup time before SPCK rises		2.7	–	ns
SPI_8	MOSI Hold time after SPCK rises		0.2	–	ns
SPI_9	SPCK rising to MISO		2.5	13.8	ns
SPI_{10}	MOSI Setup time before SPCK falls		2.2	–	ns
SPI_{11}	MOSI Hold time after SPCK falls		0.6	–	ns
SPI_{12}	NPCS0 setup to SPCK rising		4.3	–	ns
SPI_{13}	NPCS0 hold after SPCK falling		0	–	ns
SPI_{14}	NPCS0 setup to SPCK falling		3.8	–	ns
SPI_{15}	NPCS0 hold after SPCK rising		0	–	ns
SPI_{16}	NPCS0 falling to MISO valid		–	14.5	ns

Table 47-33: SPI Timings with 1.8V Peripheral Supply

Symbol	Parameter	Conditions	Min	Max	Unit
Master Mode					
SPI_{SPCK}	SPI Clock		–	66	MHz
SPI_0	MISO Setup time before SPCK rises		16.3	–	ns
SPI_1	MISO Hold time after SPCK rises		0	–	ns
SPI_2	SPCK rising to MOSI		0	6.9	ns
SPI_3	MISO Setup time before SPCK falls		15.1	–	ns
SPI_4	MISO Hold time after SPCK falls		0	–	ns
SPI_5	SPCK falling to MOSI		0	7.0	ns
Slave Mode					
SPI_6	SPCK falling to MISO		3.5	16.8	ns
SPI_7	MOSI Setup time before SPCK rises		2.9	–	ns
SPI_8	MOSI Hold time after SPCK rises		0.3	–	ns
SPI_9	SPCK rising to MISO		3.3	16.4	ns
SPI_{10}	MOSI Setup time before SPCK falls		2.4	–	ns
SPI_{11}	MOSI Hold time after SPCK falls		0.7	–	ns
SPI_{12}	NPCS0 setup to SPCK rising		4.5	–	ns
SPI_{13}	NPCS0 hold after SPCK falling		0	–	ns
SPI_{14}	NPCS0 setup to SPCK falling		3.9	–	ns
SPI_{15}	NPCS0 hold after SPCK rising		0	–	ns
SPI_{16}	NPCS0 falling to MISO valid		–	17.3	ns

© 2017 Microchip Technology Inc.

47.15.2 SSC

47.15.2.1 Timing Conditions

1.8V domain: V_{DDIO} from 1.65V to 1.95V, maximum external capacitor = 20 pF

3.3V domain: V_{DDIO} from 3.0V to 3.6V, maximum external capacitor = 30 pF

Timings are given assuming a capacitance load as specified in Table 47-34.

Table 47-34: Capacitance Load

Supply	Corner	
	Max	Min
3.3V	30 pF	5 pF
1.8V	20 pF	5 pF

47.15.2.2 Timing Extraction

Figure 47-11: SSC Transmitter, TK and TF in output

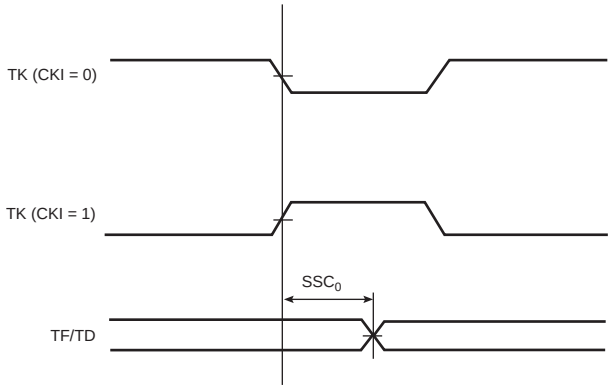


Figure 47-12: SSC Transmitter, TK in input and TF in output

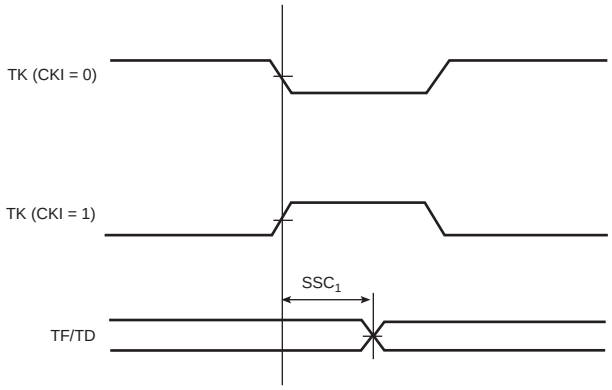


Figure 47-13: SSC Transmitter, TK in output and TF in input

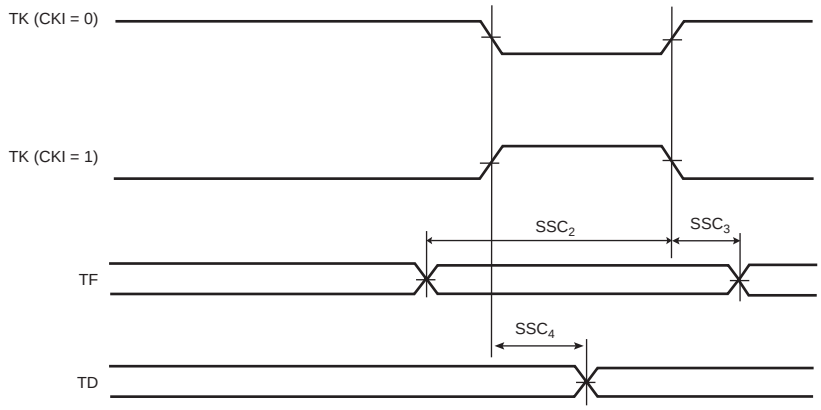


Figure 47-14: SSC Transmitter, TK and TF in Input

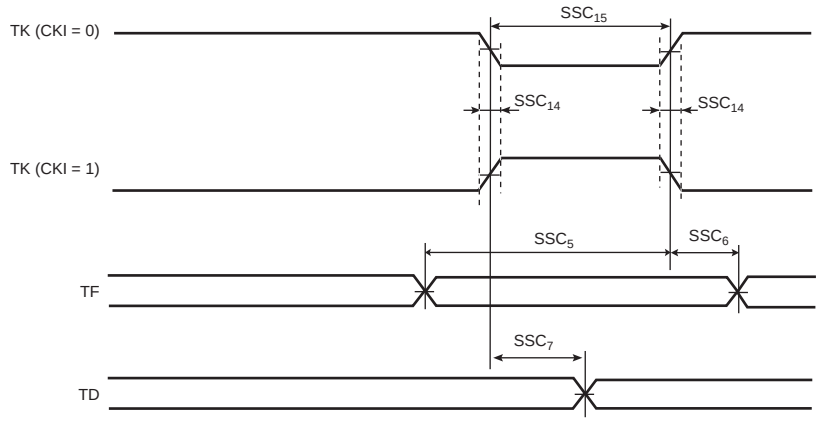
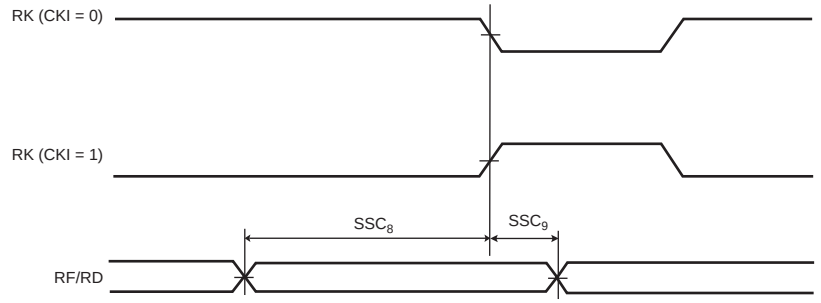


Figure 47-15: SSC Receiver RK and RF in Input



© 2017 Microchip Technology Inc.

Figure 47-16: SSC Receiver, RK in input and RF in Output

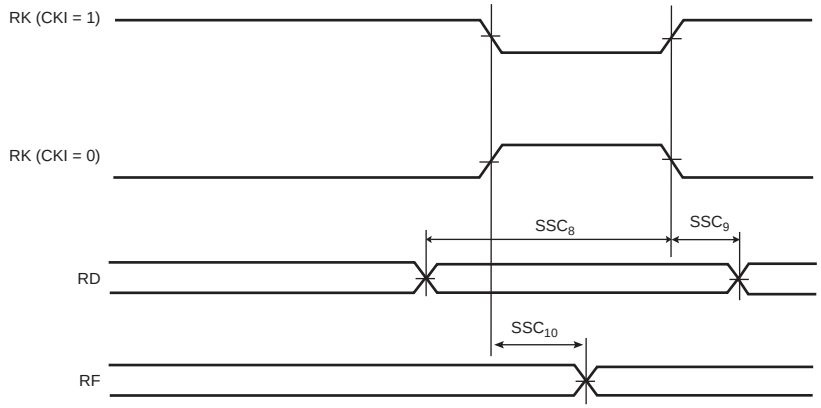


Figure 47-17: SSC Receiver, RK and RF in Output

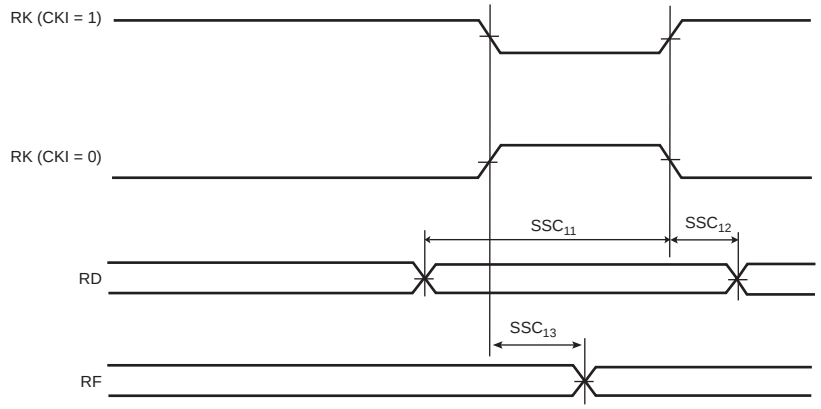


Figure 47-18: SSC Receiver, RK in Output and RF in Input

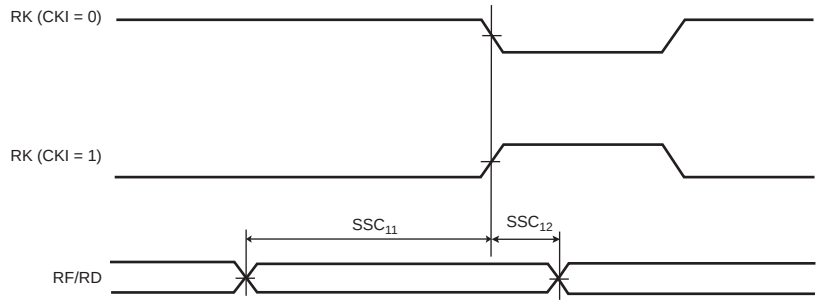


Table 47-35: SSC Timings 3.3V Domain

Symbol	Parameter	Conditions	Min	Max	Unit
		Transmitter			
SSC_0	TK edge to TF/TD (TK output, TF output)		2.1(1)	13.2(1)	ns
SSC_1	TK edge to TF/TD (TK input, TF output)		2.1(1)	11.1(1)	ns
SSC_2	TF setup time before TK edge (TK output)		10.6 - t_{CPMCK}	–	ns
SSC_3	TF hold time after TK edge (TK output)		t_{CPMCK} - 2.0	–	ns

Table 47-35: SSC Timings 3.3V Domain (Continued)

Symbol	Parameter	Conditions	Min	Max	Unit
SSC_4	TK edge to TF/TD (TK output, TF input)		2.0	13.2	ns
		STTDLY = 0 START = 4, 5 or 7	2.0 + (2 × t_{CPMCK})	13.2 + (2 × t_{CPMCK})	
SSC_5	TF setup time before TK edge (TK input)		0	–	ns
SSC_6	TF hold time after TK edge (TK input)		t_{CPMCK}	–	ns
SSC_7	TK edge to TF/TD (TK input, TF input)		2.1	11.1	ns
		STTDLY = 0 START = 4, 5 or 7	2.1 + (3 × t_{CPMCK})	11.1 + (3 × t_{CPMCK})	
Receiver					
SSC_8	RF/RD setup time before RK edge (RK input)		0	–	ns
SSC_9	RF/RD hold time after RK edge (RK input)		t_{CPMCK}	–	ns
SSC_{10}	RK edge to RF (RK input)		2.1	10.8	ns
SSC_{11}	RF/RD setup time before RK edge (RK output)		10.4 - t_{CPMCK}	–	ns
SSC_{12}	RF/RD hold time after RK edge (RK output)		t_{CPMCK} - 1.9	–	ns
SSC_{13}	RK edge to RF (RK output)		2.0	13.2	ns
SSC_{14}[(1)]	TK rise time or fall time	10 to 90%		10	ns
SSC_{15}[(1)]	TK low or high time	$V_{TK}>V_{IH}$ or $V_{TK}<V_{IL}$	3.4 + (3 x t_{CPMCK})	–	ns

Note 1: Timings SSC_{14} and SSC_{15} also apply when RK is used instead of TK (SSC_TCMR.CKS = RK).

Table 47-36: SSC Timings 1.8V Domain

Symbol	Parameter	Conditions	Min	Max	Unit
Transmitter					
SSC_0	TK edge to TF/TD (TK output, TF output)		2.1	13.5	ns
SSC_1	TK edge to TF/TD (TK input, TF output)		2.8	15.4	ns
SSC_2	TF setup time before TK edge (TK output)		14.6 - t_{CPMCK}	–	ns
SSC_3	TF hold time after TK edge (TK output)		t_{CPMCK} - 2.7	–	ns
SSC_4	TK edge to TF/TD (TK output, TF input)		2.1	13.5	ns
		STTDLY = 0 START = 4, 5 or 7	2.1 + (2 × t_{CPMCK})	13.5 + (2 × t_{CPMCK})	
SSC_5	TF setup time before TK edge (TK input)		0	–	ns
SSC_6	TF hold time after TK edge (TK input)		t_{CPMCK}	–	ns
SSC_7	TK edge to TF/TD (TK input, TF input)		2.8	15.4	ns
		STTDLY = 0 START = 4, 5 or 7	2.8 + (3 × t_{CPMCK})	15.4 + (3 × t_{CPMCK})	
Receiver					
SSC_8	RF/RD setup time before RK edge (RK input)		0	–	ns
SSC_9	RF/RD hold time after RK edge (RK input)		t_{CPMCK}	–	ns
SSC_{10}	RK edge to RF (RK input)		2.8	15.1	ns
SSC_{11}	RF/RD setup time before RK edge (RK output)		14.5 - t_{CPMCK}	–	ns
SSC_{12}	RF/RD hold time after RK edge (RK output)		t_{CPMCK} - 2.7	–	ns
SSC_{13}	RK edge to RF (RK output)		2.1	13.5	ns
SSC_{14}[(1)]	TK rise time or fall time	10 to 90%	–	10	ns

© 2017 Microchip Technology Inc.

Table 47-36: SSC Timings 1.8V Domain (Continued)

Symbol	Parameter	Conditions	Min	Max	Unit
SSC_{15}[(1)]	TK low or high time	$V_{TK}>V_{IH}$ or $V_{TK}<V_{IL}$	3.6 + (3 x t_{CPMCK})	–	ns

Note 1: Timings SSC_{14} and SSC_{15} also apply when RK is used instead of TK (SSC_TCMR.CKS = RK).

47.15.3 HSMCI

The High Speed MultiMedia Card Interface (HSMCI) supports the MultiMedia Card (MMC) Specification V4.3, the SD Memory Card Specification V2.0, the SDIO V2.0 specification and CE-ATA V1.1.

47.15.4 USART in SPI Mode Timings

47.15.4.1 Timing Conditions

1.8V domain: V_{DDIO} from 1.65V to 1.95V, maximum external capacitor = 20 pF

3.3V domain: V_{DDIO} from 3.0V to 3.6V, maximum external capacitor = 40 pF

Timings are given assuming a capacitance load as specified in Table 47-34.

Table 47-37: Capacitance Load

Supply	Corner	
	Max	Min
3.3V	40 pF	5 pF
1.8V	20 pF	5 pF

47.15.4.2 Timing Extraction

Figure 47-19: USART SPI Master Mode

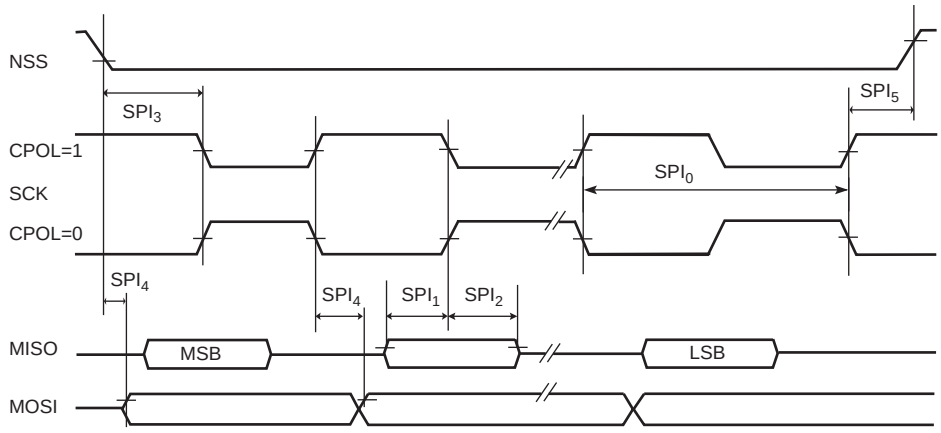


Figure 47-20: USART SPI Slave mode: (Mode 1 or 2)

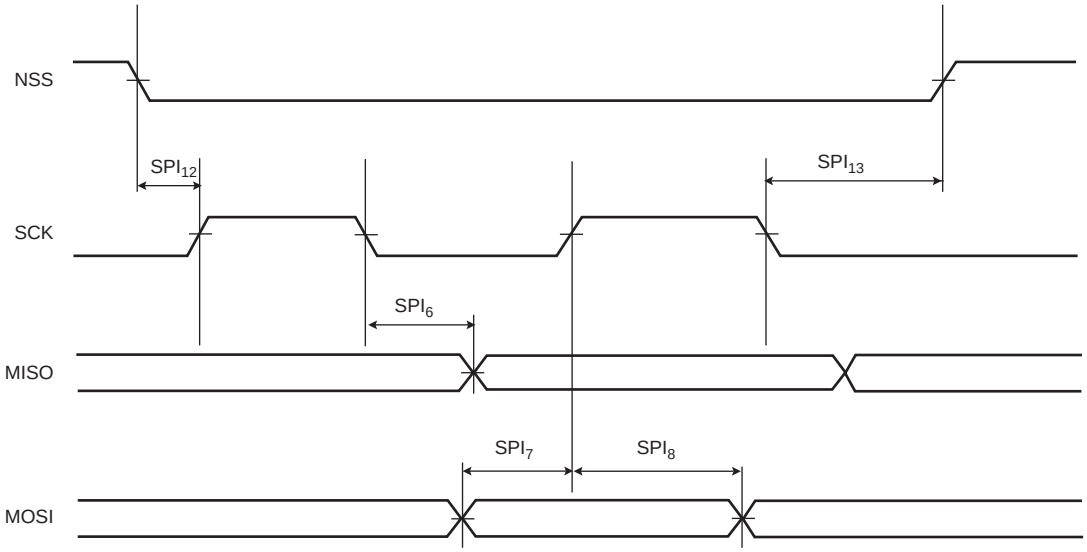


Figure 47-21: USART SPI Slave mode: (Mode 0 or 3)

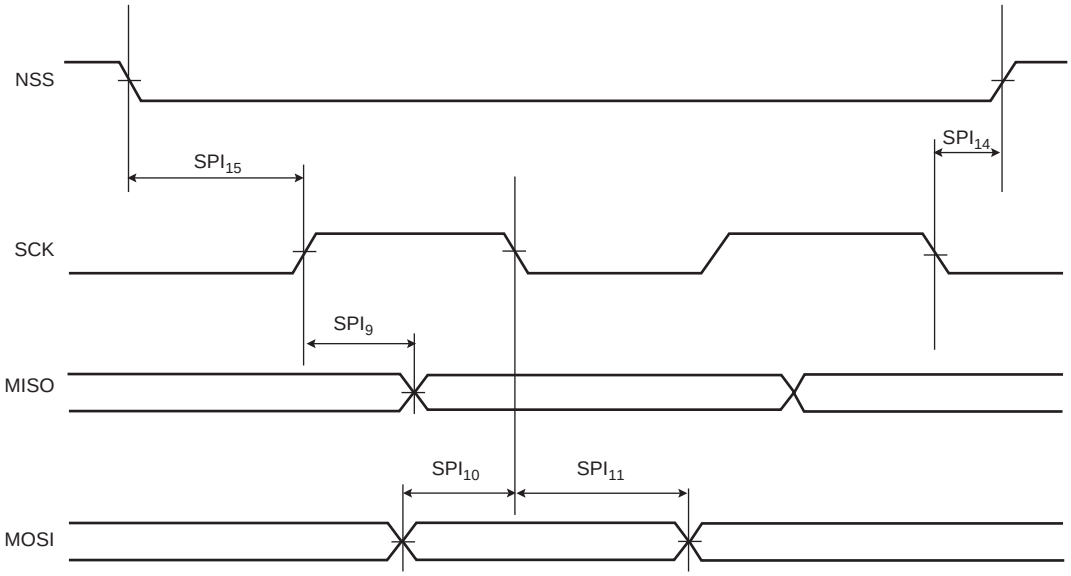


Table 47-38: USART SPI Timings

Symbol	Parameter	Conditions	Min	Max	Unit
Master Mode					
SPI_0	SCK Period	1.8v domain 3.3v domain	MCK/6	–	ns
SPI_1	Input Data Setup Time	1.8v domain 3.3v domain	0.5 × MCK + 3.5 0.5 × MCK + 3.3	–	ns
SPI_2	Input Data Hold Time	1.8v domain 3.3v domain	1.5 × MCK + 1.1 1.5 × MCK + 0.8	–	ns

© 2017 Microchip Technology Inc.

Table 47-38: USART SPI Timings (Continued)

Symbol	Parameter	Conditions	Min	Max	Unit
SPI_3	Chip Select Active to Serial Clock	1.8v domain 3.3v domain	1.5 × SCK - 1.9 1.5 × SCK - 2.5	–	ns
SPI_4	Output Data Setup Time	1.8v domain 3.3v domain	0 0	7.6 8.4	ns
SPI_5	Serial Clock to Chip Select Inactive	1.8v domain 3.3v domain	1 × SCK - 7.1 1 × SCK - 7.8	–	ns
Slave Mode					
SPI_6	SCK falling to MISO	1.8v domain 3.3v domain	3.8 3.0	19.5 16.6	ns
SPI_7	MOSI Setup time before SCK rises	1.8v domain 3.3v domain	2 × MCK + 3.0 2 × MCK + 2.7	–	ns
SPI_8	MOSI Hold time after SCK rises	1.8v domain 3.3v domain	1.5 1.3	–	ns
SPI_9	SCK rising to MISO	1.8v domain 3.3v domain	3.6 2.9	19.1 16.6	ns
SPI_{10}	MOSI Setup time before SCK falls	1.8v domain 3.3v domain	2 × MCK + 2.6 2 × MCK + 2.4	–	ns
SPI_{11}	MOSI Hold time after SCK falls	1.8v domain 3.3v domain	1.5 1.2	–	ns
SPI_{12}	NPCS0 setup to SCK rising	1.8v domain 3.3v domain	2.5 × MCK + 1.4 2.5 × MCK + 1.1	–	ns
SPI_{13}	NPCS0 hold after SCK falling	1.8v domain 3.3v domain	1.5 × MCK + 2.4 1.5 × MCK + 2.1	–	ns
SPI_{14}	NPCS0 setup to SCK falling	1.8v domain 3.3v domain	2.5 × MCK + 1.1 2.5 × MCK + 1.0	–	ns
SPI_{15}	NPCS0 hold after SCK rising	1.8v domain 3.3v domain	1.5 × MCK + 1.8 1.5 × MCK + 1.6	–	ns

47.15.5 UDP

Figure 47-22: USB Data Signal Rise and Fall Times

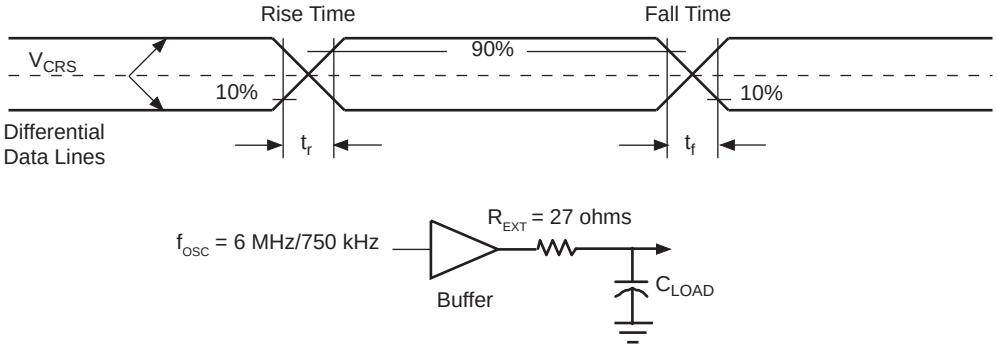


Table 47-39: In Full Speed

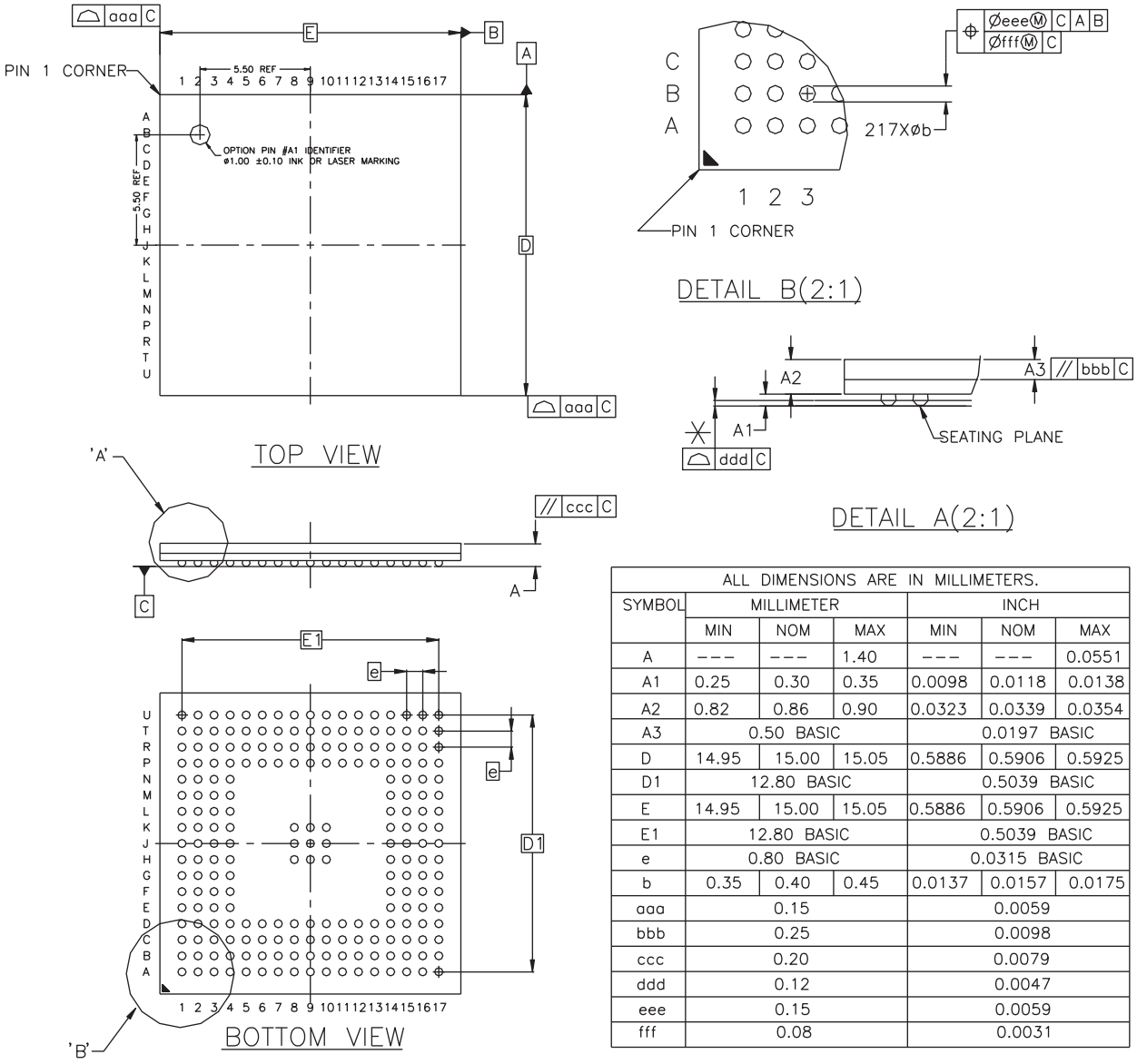
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_r	Transition Rise Time	C_{LOAD} = 50 pF	4	–	20	ns
t_f	Transition Fall Time	C_{LOAD} = 50 pF	4	–	20	ns
t_{rfm}	Rise/Fall Time Matching		90	–	111.11	%

© 2017 Microchip Technology Inc.

48. Mechanical Characteristics

48.1 217-ball BGA Mechanical Characteristics

Figure 48-1: 217-ball BGA Package Drawing



ALL DIMENSIONS ARE IN MILLIMETERS.						
SYMBOL	MILLIMETER			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	---	---	1.40	---	---	0.0551
A1	0.25	0.30	0.35	0.0098	0.0118	0.0138
A2	0.82	0.86	0.90	0.0323	0.0339	0.0354
A3	0.50 BASIC			0.0197 BASIC		
D	14.95	15.00	15.05	0.5886	0.5906	0.5925
D1	12.80 BASIC			0.5039 BASIC		
E	14.95	15.00	15.05	0.5886	0.5906	0.5925
E1	12.80 BASIC			0.5039 BASIC		
e	0.80 BASIC			0.0315 BASIC		
b	0.35	0.40	0.45	0.0137	0.0157	0.0175
aaa	0.15			0.0059		
bbb	0.25			0.0098		
ccc	0.20			0.0079		
ddd	0.12			0.0047		
eee	0.15			0.0059		
fff	0.08			0.0031		

Table 48-1: BGA217 Ball Information

Ball pitch	0.8 mm +/- 0.05
Ball Diameter	0.4 mm +/- 0.05

Table 48-2: BGA217 Soldering Information

Ball Land	0.43 mm +/- 0.05
Solder Mask Opening	0.30 mm +/- 0.05

Table 48-3: Device and BGA217 Package Maximum Weight

450	mg

Table 48-4: BGA217 Package Characteristics

Moisture Sensitivity Level	3

Table 48-5: BGA217 Package Reference

JEDEC Drawing Reference	MO-205
JESD97 Classification	e1

Table 48-6: BGA217 – Recommended Soldering Profile from J-STD-20

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3°C/sec. max.
Preheat Temperature 175°C ±25°C	180 sec. max.
Temperature Maintained Above 217°C	60 sec. to 150 sec.
Time within 5°C of Actual Peak Temperature	20 sec. to 40 sec.
Peak Temperature Range	260 +0°C
Ramp-down Rate	6°C/sec. max.
Time 25°C to Peak Temperature	8 min. max.

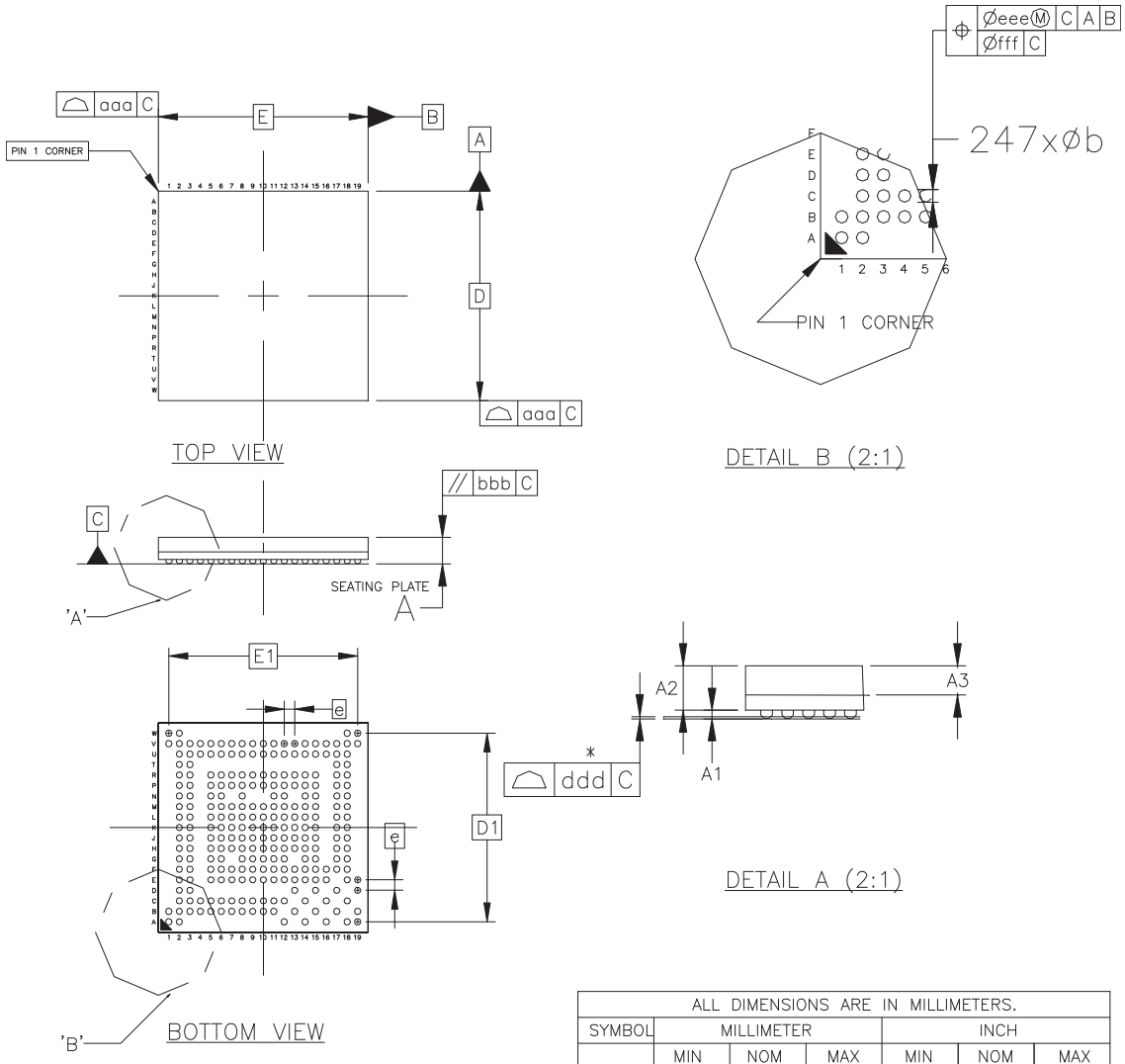
Note: It is recommended to apply a soldering temperature higher than 250°C.

A maximum of three reflow passes is allowed per component.

© 2017 Microchip Technology Inc.

48.2 247-ball BGA Mechanical Characteristics

Figure 48-2: 247-ball BGA Package Drawing



ALL DIMENSIONS ARE IN MILLIMETERS.						
SYMBOL	MILLIMETER			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	---	---	1.37	---	---	0.0539
A1	0.16	0.21	0.26	0.0063	0.0083	0.0102
A2	1.02	1.06	1.10	0.0402	0.0417	0.0433
A3	0.70 BASIC			0.0276 BASIC		
D	9.90	10.00	10.10	0.3898	0.3937	0.3976
D1	9 BASIC			0.3543BASIC		
E	9.90	10.00	10.10	0.3898	0.3937	0.3976
E1	9 BASIC			0.3543BASIC		
e	0.5 BASIC			0.0197 BASIC		
b	0.25	0.3	0.35	0.0098	0.0118	0.0138
aaa	0.10			0.0039		
bbb	0.20			0.0079		
ddd	0.08			0.0031		
eee	0.15			0.0059		
fff	0.05			0.0020		

Table 48-7: BGA247 Ball Information

Ball pitch	0.5 mm +/- 0.05
Ball Diameter	0.3 mm +/- 0.05

Table 48-8: BGA247 Soldering Information

Ball Land	0.35 mm +/- 0.05
Solder Mask Opening	0.27 mm +/- 0.05

Table 48-9: Device and BGA247 Package Maximum Weight

177	mg

Table 48-10: BGA247 Package Characteristics

Moisture Sensitivity Level	3

Table 48-11: BGA247 Package Reference

JEDEC Drawing Reference	none
JESD97 Classification	e1

Table 48-12: 247-ball TFBGA – Recommended Soldering Profile from J-STD-20

Profile Feature	**Green Package**
Average Ramp-up Rate (217°C to Peak)	3°C/sec. max.
Preheat Temperature 175°C ±25°C	180 sec. max.
Temperature Maintained Above 217°C	60 sec. to 150 sec.
Time within 5°C of Actual Peak Temperature	20 sec. to 40 sec.
Peak Temperature Range	260 +0°C
Ramp-down Rate	6°C/sec. max.
Time 25°C to Peak Temperature	8 min. max.

Note: It is recommended to apply a soldering temperature higher than 250°C.

A maximum of three reflow passes is allowed per component.

© 2017 Microchip Technology Inc.

49. Marking

All devices are marked with the company logo and the ordering code.

Additional marking is as follows:



where

- “YY”: Manufactory year
- “WW”: Manufactory week
- “C”: Assembly country code (optional)
- “V”: Revision
- “XXXXXXX”: Lot number

50. Ordering Information

Table 50-1: SAM9N12/CN11/CN12 Ordering Information

Ordering Code	MRL	Package	Carrier Type	Operating Temperature Range
AT91SAM9CN12-CU	A	BGA217	Tray	Industrial -40°C to 85°C
AT91SAM9CN11-CU(1)	A	BGA217	Tray	Industrial -40°C to 85°C
AT91SAM9N12-CU	A	BGA217	Tray	Industrial -40°C to 85°C
AT91SAM9CN12-CFU	A	BGA247	Tray	Industrial -40°C to 85°C
AT91SAM9CN11-CFU(1)	A	BGA247	Tray	Industrial -40°C to 85°C
AT91SAM9N12-CFU	A	BGA247	Tray	Industrial -40°C to 85°C
AT91SAM9CN12B-CU	B	BGA217	Tray	Industrial -40°C to 85°C
AT91SAM9CN12B-CUR	B	BGA217	Tape & Reel	Industrial -40°C to 85°C
AT91SAM9CN12B-CFU	B	BGA247	Tray	Industrial -40°C to 85°C
AT91SAM9CN12B-CFUR	B	BGA247	Tape & Reel	Industrial -40°C to 85°C

Note 1: The SAM9CN11 device is for evaluation only.

© 2017 Microchip Technology Inc.

51. Errata

Errata is gathered in two sections:

Section 51.1 “Errata: Revision A Parts”

Section 51.2 “Errata: Revision B Parts”

51.1 Errata: Revision A Parts

51.1.1 BootROM - SAM9CN12 - Rev. A

51.1.1.1 Boot from SPI Serial Flash Devices (xx25xxx) Is not Working

The boot from SPI serial Flash series is not working. SAM9CN12 supports only booting from AT45 DataFlash series.

Problem Fix/Workaround

AT45 DataFlash series must be used. Use one of the following Data/Serial Flashes: AT45DB321, AT45DB642.

51.1.2 16 MHz Main Crystal - SAM9CN12 - Rev. A

51.1.2.1 16 MHz Main Crystal Detection

The 16 MHz crystal may not be detected properly in harsh conditions (high temperature, low voltage).

The code uses only ±3% tolerance drift for the internal 12 MHz RC, used for main crystal detection, instead of ±5%.

This leads to a bad detection of the main crystal, if 16 MHz value is used, and to a lock of the boot.

This issue does not apply if the main crystal is 12 MHz.

Problem Fix/Workaround

Use the 12 MHz value for the main crystal.

51.1.3 LCD Controller (LCDC) - SAM9N12 / SAM9CN11 / SAM9CN12 - Rev. A

51.1.3.1 LCDC: LCDC PWM Is Not Usable

When slow clock is selected as the source clock to feed PWM with (CLKPWMSEL in LCDC_LCDCFG0), the output waveform generated is corrupted. When the MCK is selected the prescaler (PWMPS in LCDC_LCDCFG6) is not sized to generate the PWM output in a range of 200 Hz–1 kHz.

Problem Fix/Workaround

Use standalone PWM output instead of LCDC embedded PWM.

51.1.4 12 MHz RC Oscillator - SAM9N12 / SAM9CN11 / SAM9CN12 - Rev. A

51.1.4.1 Reset Hangs with 12 MHz RC Disabled

After power-up, the 12 MHz RC is enabled automatically. It is not enabled after a reset, even if it is used during the boot process.

The user must not disable the 12 MHz RC to avoid any system freezing on reset event.

Problem Fix/Workaround

None

51.1.5 Reset Controller (RSTC) - SAM9N12 / SAM9CN11 - Rev. A

51.1.5.1 RSTC: Reset during SDRAM Accesses

When a Reset (User reset, watchdog, software reset) occurs during SDRAM read access, the SDRAM clock is turned off while data is ready to be read on the data bus. The SDRAM maintains the data until the clock restarts.

This leads to a data bus conflict and adversely affects the boot memories connected on the EBI:

- NAND Flash boot functionality, if the system boots out of internal ROM.
- NOR Flash boot, if the system boots on an external memory connected on the EBI CS0.

© 2017 Microchip Technology Inc.

Problem Fix/Workaround

1. Boot from Serial Flash or Data Flash on SPI
2. Connect the NAND Flash on D16–D23 and set NFD0_ON_D16 to 1 in CCFG_EBICSA register.
   Warning! This prohibits connecting another device on the EBI.

51.1.6 Universal Synchronous Asynchronous Receiver Transmitter (USART) - SAM9N12 / SAM9CN11 / SAM9CN12 - Rev. A

51.1.6.1 Bad frame detection issue

If a bad frame is received (incorrect baud rate) with the last data bit being sampled at 1, there is no detection of frame error.

Problem Fix/Workaround

There is no general fix. When performing baud rate detection with receive part, the transmit frame must be sent with a parity bit set to 0.

51.2 Errata: Revision B Parts

51.2.1 Boot ROM - SAM9CN12 - Rev. B

51.2.1.1 Boot from SPI Data/Serial Flash Devices do not work with All Memories

The boot from SPI Data/Serial Flash series does not work with all memories.

Problem Fix/Workaround

Use one of the following Data/Serial Flashes: AT45DB321, AT45DB642, AT25DF161, AT25DF512B, AT25DF021, AT25DF041A, AT25DF321, AT25DF641A, M25P05, M25P10, M25P20, M25P80, M25P16, M25P32, M25P64.

51.2.2 Reset Controller (RSTC) - SAM9N12 / SAM9CN11 - Rev. B

51.2.2.1 RSTC: Reset during SDRAM Accesses

When a Reset (User reset, watchdog, software reset) occurs during SDRAM read access, the SDRAM clock is turned off while data is ready to be read on the data bus. The SDRAM maintains the data until the clock restarts.

This leads to a data bus conflict and adversely affects the boot memories connected on the EBI:

- NAND Flash boot functionality, if the system boots out of internal ROM.
- NOR Flash boot, if the system boots on an external memory connected on the EBI CS0.

Problem Fix/Workaround

1. Boot from Serial Flash or Data Flash on SPI
2. Connect the NAND Flash on D16–D23 and set NFD0_ON_D16 to 1 in CCFG_EBICSA register.
   Warning! This prohibits connecting another device on the EBI.

51.2.3 LCD Controller (LCDC) - SAM9N12 / SAM9CN11 / SAM9CN12 - Rev. B

51.2.3.1 LCDC: LCDC PWM Is Not Usable

When slow clock is selected as the source clock to feed PWM with (CLKPWMSEL in LCDC_LCDCFG0), the output waveform generated is corrupted. When the MCK is selected the prescaler (PWMPS in LCDC_LCDCFG6) is not sized to generate the PWM output in a range of 200 Hz–1 kHz.

Problem Fix/Workaround

Use standalone PWM output instead of LCDC embedded PWM.

© 2017 Microchip Technology Inc.

51.2.4 12 MHz RC Oscillator - SAM9N12 / SAM9CN11 / SAM9CN12 - Rev. B

51.2.4.1 Reset Hangs with 12 MHz RC Disabled

After power-up, the 12 MHz RC is enabled automatically. It is not enabled after a reset, even if it is used during the boot process.

The user must not disable the 12 MHz RC to avoid any system freezing on reset event.

Problem Fix/Workaround

None

51.2.5 Universal Synchronous Asynchronous Receiver Transmitter (USART) - SAM9N12 / SAM9CN11 / SAM9CN12 - Rev. B

51.2.5.1 Bad frame detection issue

If a bad frame is received (incorrect baud rate) with the last data bit being sampled at 1, there is no detection of frame error.

Problem Fix/Workaround

There is no general fix. When performing baud rate detection with receive part, the transmit frame must be sent with a parity bit set to 0.

52. Revision History

In the tables that follow, the most recent version of the document appears first.

Issue Date	Changes
Oct-17	General - Template update: Moved from Atmel to Microchip template. - The datasheet is assigned a new document number (DS60001517) and revision letter is reset to A. --- Document number DS60001517 revision A corresponds to what would have been 11063 revision M. - ISBN number assigned.
	Note “SAM9CN11 is for evaluation only” added throughout the document.
	Section 47. “Electrical Characteristics” Removed Figure 47-11 "Min and Max access time for SPI output signal" and Figure 47-20 "Min and Max Access Time of Output Signals" Table 47-35 ”SSC Timings 3.3V Domain” and Table 47-36 ”SSC Timings 1.8V Domain”: removed footnote “For output signals (TF, TD, RF), Min and Max access times are defined, etc.” Updated the following with SSC_{14} and SSC_{15} content: - Figure 47-14 "SSC Transmitter, TK and TF in Input" - Table 47-35 "SSC Timings 3.3V Domain” - Table 47-36 ”SSC Timings 1.8V Domain”
	Section 51. “Errata” Added: - Section 51.1.6 “Universal Synchronous Asynchronous Receiver Transmitter (USART) - SAM9N12 / SAM9CN11 / SAM9CN12 - Rev. A”: “Bad frame detection issue” - Section 51.2.5 “Universal Synchronous Asynchronous Receiver Transmitter (USART) - SAM9N12 / SAM9CN11 / SAM9CN12 - Rev. B”: “Bad frame detection issue”

© 2017 Microchip Technology Inc.

Doc. Rev. 11063L	Comments
	General editorial and formatting changes throughout Selected instances of “AT91SAM9” updated to “SAM9”
	Section 1. ”Block Diagram” Figure 1-1 "SAM9N12/CN11/CN12 Block Diagram": updated ADC signal names
	Section 2. ”Signal Description” Table 2-1 ”Signal Description List”: updated ADC signal names
	Section 3. ”Package and Pinout” Updated first sentence Table 3-1 ”BGA217 Pin Description”: - for GNDFUSE signal, defined direction as ‘I’ - for BMS signal, power rail ‘VDDCORE’ changed to ‘VDDIOP0’; reset state ‘I, PD, ST’ changed to ‘I, PU, ST’ Table 3-2 ”BGA247 Pin Description”: - for signals D0–D15, reset state ‘I, PD’ changed to ‘O, PD’ - for BMS signal, power rail ‘VDDCORE’ changed to ‘VDDIOP0’ - for JTAGSEL signal, reset state ‘I, PD, ST’ changed to ‘I, PD’ - for TDO signal, reset state f ‘O, ST’ changed to ‘O’ - for RTCK signal, reset state ‘O, ST’ changed to ‘O’ - for NRST signal, reset state ‘O, PU, ST’ changed to ‘I, PU, ST’
28-Oct-15	Section 4. ”Power Considerations” Added Section 4.2 ”Power Sequence Requirements” (includes Section 4.2.1 ”Power-Up Sequence” and Section 4.2.2 ”Power-Down Sequence”)
	Section 5. ”Memories” Figure 5-1 "SAM9N12/CN11/CN12 Memory Map": updated selected acronyms under “Peripheral Mapping” and “System Controller Mapping”
	Section 6. ”System Controller” Figure 6-1 "SAM9N12/CN11/CN12 System Controller Block Diagram": removed two instances of rstc_irq signal; Debug Unit input “MCK” renamed to “peripheral clock”; Periodic Interval Timer input “MCK” renamed to “peripheral clock”; “BSCR” corrected to “BSC_CR” Updated Section 6.4 ”Backup Section”
	Section 7. ”Peripherals” Table 7-1 ”SAM9N12/CN11/CN12 Peripheral Identifiers”: updated selected instance descriptions Added Section 7.5.3 ”Fuse Mapping”
	Section 10. ”Advanced Interrupt Controller (AIC)” Harmonized instances of AIC_FVR description (is “FIQ Vector Register”) Section 10.2 ”Embedded Characteristics”: “Write Protection Registers” renamed to “Register Write Protection” Section 10.8.8 ”Register Write Protection”: updated title (was “Write Protection Registers”) and updated content Section 10.9.13 ”AIC End of Interrupt Command Register”: added ENDIT bit Section 10.9.15 ”AIC Debug Control Register”: “GMSK: General Mask” corrected to “GMSK: General Interrupt Mask”

© 2017 Microchip Technology Inc.

Doc. Rev. 11063L	Comments (Continued)
28-Oct-15	Section 10. "Advanced Interrupt Controller (AIC)" (cont'd) Updated Section 10.9.19 "AIC Write Protection Mode Register" and Section 10.9.20 "AIC Write Protection Status Register"
	Section 11. "Boot Strategies" Section 11.1 "SAM9CN12 Only": in last sentence, updated name of referenced application note Updated Section 11.2.4.1 "NVM Boot Sequence" Section 11.2.4.2 "NVM Bootloader Program Description": in last paragraph, deleted sentence "All the calls to functions are PC relative and do not use absolute addresses" Section 11.2.5.3 "USB Device Port": deleted reference to "Windows XP"
	Section 12. "Boot Sequence Controller (BSC)" Register name "Boot Sequence Configuration Register" corrected to "Boot Sequence Controller Configuration Register" Section 12.4.1 "Boot Sequence Controller Configuration Register": updated BOOT field description
	Section 13. "Reset Controller (RSTC)" Figure 13-1 "Reset Controller Block Diagram": deleted signal "rstc_irq" Section 13.4 "Functional Description": deleted subsection "Reset Controller Status Register" (redundant with Section 13.5.2 "Reset Controller Status Register") Table 13-1 "Register Mapping": corrected RSTC_SR reset value and replaced single footnote with two footnotes Section 13.5.2 "Reset Controller Status Register": updated descriptions of bits URSTS, NRSTL, and SRCMP
	Section 14. "Real-time Clock (RTC)" Updated Section 14.1 "Description" Updated Section 14.2 "Embedded Characteristics" Updated Section 14.5 "Functional Description" Section 14.5.5 "Updating Time/Calendar": updated second paragraph Section 14.6.1 "RTC Control Register": added sentence on register write protection; updated description of bits UPDTIM and UPDCAL Section 14.6.2 "RTC Mode Register": added sentence on register write protection Section 14.6.3 "RTC Time Register": deleted "All non-significant bits read zero" Section 14.6.4 "RTC Calendar Register": deleted "All non-significant bits read zero"; updated description of CENT field Section 14.6.5 "RTC Time Alarm Register": added sentence on register write protection Section 14.6.6 "RTC Calendar Alarm Register": added sentence on register write protection
	Section 15. "Periodic Interval Timer (PIT)" Section 15.2 "Embedded Characteristics": removed bullet "AMBA™-compliant Interface"
	Section 16. "Watchdog Timer (WDT)" Section 16.4 "Functional Description": inserted paragraph "When setting the WDDIS bit, and while it is set, the fields WDV and WDD must not be modified." Figure 16-2 "Watchdog Behavior": "WDT_CR = WDRSTT" corrected to "WDT_CR.WDRSTT=1" Section 16.5.1 "Watchdog Timer Control Register": added note on modification of WDT_CR values Section 16.5.2 "Watchdog Timer Mode Register": updated second note below bitmap; updated WDDIS bit description Section 16.5.3 "Watchdog Timer Status Register": updated descriptions of bits WDUNF and WDERR
	Section 17. "Shutdown Controller (SHDWC)" Updated Section 17.2 "Embedded Characteristics" Section 17.6 "Functional Description": inserted heading "Wake-up Inputs"

© 2017 Microchip Technology Inc.

Doc. Rev. 11063L	Comments (Continued)
28-Oct-15	Section 18. "General Purpose Backup Registers (GPBR)" Updated Section 18.1 "Description" and Section 18.2 "Embedded Characteristics" Table 18-1 "Register Mapping": added reset value 0x00000000 for registers SYS_GPBRx Section 18.3.1 "General Purpose Backup Register x": inserted sentence "These registers are reset at first power-up and on each loss of VDDBU"
	Section 19. "Slow Clock Controller (SCKC)" Section 19.1 "Description" Inserted heading Section 19.4 "Functional Description" and updated content Harmonized description of SCKC_CR throughout (is "Slow Clock Controller Configuration Register")
	Section 20. "Clock Generator" Section 20.2 "Embedded Characteristics": "One 16 MHz Main Oscillator" changed to "3 to 20 MHz Main Oscillator" Figure 20-1 "Clock Generator Block Diagram": "16M Main Oscillator" changed to "3–20 MHz Main Oscillator" Figure 20-2 "Main Clock Block Diagram": "MAINRDY" corrected to "MAINFRDY" Removed section "Slow Clock Selection" (including subsections) Section 20.4 "Main Clock": register name "Slow Clock Control Register (SCKCR)" corrected to "Slow Clock Controller Configuration Register (SCKC_CR)" Added Section 20.4.4 "Bypass the Main Crystal Oscillator"
	Section 21. "Power Management Controller (PMC)" Section 21.6 "USB Device and Host Clocks": at end of first paragraph, deleted "depending on the USBDIV bit in CKGR_PLLBR" Figure 21-2 "USB Clock Controller": removed "/1,/2,/4" from Divider block Section 21.10 "Programming Sequence": revised description of Step 1; in Step 3, in sentence preceding the code example, changed "the additional divider by 1, 2 or 4, which generates the USB clock(s)" to "the additional divider which generates the USB clock(s)" Added Section 21.12 "Register Write Protection" Table 21-3 "Register Mapping": PMC_PLLICPR access "Write-only" changed to "Read/Write"; defined offset range 0x007C–0x00FC as reserved Section 21.13.1 "PMC System Clock Enable Register": added sentence about disabling write protection Section 21.13.2 "PMC System Clock Disable Register": added sentence about disabling write protection Section 21.13.8 "PMC Clock Generator Main Clock Frequency Register": added sentence about disabling write protection; updated MAINFRDY bit description Section 21.13.9 "PMC Clock Generator PLLA Register": added sentence about disabling write protection; changed name of bit 29 from '1' to 'ONE' and added bit description Section 21.13.11 "PMC Master Clock Register": added sentence about disabling write protection Section 21.13.13 "PMC Programmable Clock Register": added sentence about disabling write protection Added bit configuration values in Section 21.13.14 "PMC Interrupt Enable Register", Section 21.13.15 "PMC Interrupt Disable Register" and Section 21.13.17 "PMC Interrupt Mask Register" Section 21.13.18 "PLL Charge Pump Current Register": access "Write-only" changed to "Read/Write"; added sentence about disabling write protection; updated ICPLLA bit description Section 21.13.19 "PMC Write Protection Mode Register": deleted reset value (reset values are provided in Table 21-3 "Register Mapping"); updated field descriptions Section 21.13.20 "PMC Write Protection Status Register": deleted reset value (reset values are provided in Table 21-3 "Register Mapping"); updated WPVSRC field description

Doc. Rev. 11063L	Comments (Continued)
28-Oct-15	Section 22. "Parallel Input/Output Controller (PIO)" Instances of "MCK", "master clock", "PIO clock", and "PIO controller clock" changed to "peripheral clock"; instances of "div_slclk" changed to "div_slck"; instances of "slow_clock" changed to "slck" Section 22.2 "Embedded Characteristics": "Write Protect Registers" renamed to "Register Write Protection" Section 22.3 "Block Diagram": updated Figure 22-1 "Block Diagram"; deleted figure "Application Block Diagram" Figure 22-2 "I/O Line Control Logic": updated connectivity between clocks and glitch/debouncing filter block; renamed "Resynchronization Stage" to "Peripheral Clock Resynchronization Stage" Section 22.5.1 "Pull-up and Pull-down Resistor Control": updated information to specify that pull-up or pull-down can be set Section 22.5.3 "Peripheral A or B or C or D Selection": - added two sentences on products that do not have A, B, C or D peripherals, beginning with "If the software selects a peripheral A,B,C or D which does not exist for a pin..." - in fourth bullet, "bit at level zero in PIO_ABCDSR2" corrected to "bit at level one in PIO_ABCDSR2" Section 22.5.10 "Input Edge/Level Interrupt": reorganized example of interrupt generation (configuration subsections migrated into Table 22-2 "Configuration for Example Interrupt Generation") Section 22.5.15 "Register Write Protection": in list of registers, "PIO Pad Pull-Down Status Register" replaced by "PIO Pad Pull-Down Enable Register" Table 22-4 "Register Mapping": reserved offset range 0x00EC–0x00F8 corrected to 0x00EC–0x00FC) Section 22.6.32 "PIO Pad Pull-Down Status Register": deleted sentence about disabling write protection Section 22.6.38 "PIO Additional Interrupt Modes Mask Register": modified P0–P31 bit description Section 22.6.45 "PIO Write Protection Mode Register": deleted "Reset" line; deleted redundant sentence "For more information on write-protecting registers, refer to Section 23.5.14 "Register Write Protection" Deleted "Reset" line in Section 22.6.46 "PIO Write Protection Status Register", Section 22.6.47 "PIO Schmitt Trigger Register", Section 22.6.48 "PIO I/O Delay Register", Section 22.6.49 "PIO I/O Drive Register 1" and Section 22.6.50 "PIO I/O Drive Register 2"
	Section 23. "Debug Unit (DBGU)" "Master clock" or "MCK" replaced by "peripheral clock" Updated Section 23-1 "Debug Unit Functional Block Diagram" Section 23.6.10 "Debug Unit Chip ID Register": updated EPROC and ARCH field descriptions
	Section 25. "Bus Matrix (MATRIX)" Section 25.10.5.1 "EBI Chip Select Assignment Register": updated EBI_DRIVE bit description
	Section 26. "External Bus Interface (EBI)" Table 26-3 "EBI Pins and External Static Device Connections": updated controller names ("DDRC" and "SDRAMC" changed to "DDRSDRC"; "NFC" changed to "SMC")
	Section 27. "Programmable Multibit Error Correction Code Controller (PMECC)" Table 27-4 "Register Mapping": removed reset value from write-only registers; defined offset range 0x02C–0x03C as reserved Removed reset value from register description sections (reset values are provided in Table 27-4 "Register Mapping") Section 27.6.10 "PMECC Interrupt Mask Register": description "ERRIM: Error Interrupt Enable" changed to "ERRIM: Error Interrupt Mask" Section 27.6.11 "PMECC Interrupt Status Register": description "ERRIS: Error Interrupt Status Register" changed to "ERRIS: Error Interrupt Status"
	Section 29. "Static Memory Controller (SMC)" Updated Section 29.9.1.3 "Read Cycle" and Section 29.9.3.3 "Write Cycle"

© 2017 Microchip Technology Inc.

Doc. Rev. 11063L	Comments (Continued)
28-Oct-15	Section 29. "Static Memory Controller (SMC)" (cont'd) Section 29.9 "Standard Read and Write Protocols": renamed, updated and moved subsection "Write Protected Registers" to Section 29.15 "Register Write Protection" Section 29.14.2 "Byte Access Type in Page Mode": instance of "SMC_REGISTER" corrected to "SMC Mode Register (SMC_MODE)" Table 29-9 "Register Mapping": removed registers SMC_DELAY1–SMC_DELAY8 (offset range 0xC0–0xDC now reserved) Added sentence about disabling write protection in Section 29.16.1 "SMC Setup Register", Section 29.16.2 "SMC Pulse Register" and Section 29.16.3 "SMC Cycle Register" Removed section "SMC DELAY I/O Register" Section 29.16.4 "SMC Mode Register": added sentence about disabling write protection; updated descriptions of fields READ_MODE, WRITE_MODE, EXNW_MODE, BAT, DBW, and PS Updated Section 29.16.5 "SMC Write Protection Mode Register" and Section 29.16.6 "SMC Write Protection Status Register"
	Section 30. "DDR SDR SDRAM Controller (DDRSDRC)" Removed instances of or references to "temperature compensated self refresh", "TCR" field, and acronym "TCSR" Section 30.5.1 "SDRAM Controller Write Cycle": added note defining TWRD Section 30.5.4 "Power Management": added note specifying that possible SDRAM constraint of 4K cycles of burst auto-refresh is not supported Figure 30-12 "Single Read Access, Row Closed, Latency = 3, DDR2-SDRAM Device": corrected "Latency = 2" to "Latency = 3" and inserted third cycle Figure 30-16 "Burst Read Access, Latency = 2, SDR-SDRAM Devices": removed DQS[1:0] waveform Updated Section 30.5.6 "Register Write Protection" Section 30.6.3 "SDR-SDRAM Address Mapping for 32-bit Memory Data Bus Width": updated footnote 2 Table 30-16 "Register Mapping": added reserved offset 0x28 Removed "Reset" line from individual register descriptions (reset values are provided in Table 30-16 "Register Mapping") Section 30.7.7 "DDRSDRC Low-power Register": updated TIMEOUT field description Section 30.7.10 "DDRSDRC High Speed Register": updated DIS_ANTICIP_READ bit description Section 30.7.11 "DDRSDRC Write Protection Mode Register": updated WPEN bit description
	Section 31. "DMA Controller (DMAC)" Section 31.2 "Embedded Characteristics": added bullet "Register Write Protection" Added Section 31.5 "Product Dependencies" Section 31.6.3.1 "Software Handshaking": instance of "last transaction register" corrected to "Software Last Transfer Flag Register" Section 31.6.4.3 "Ending Multi-buffer Transfers": in second paragraph, "automatic mode is disabled by writing a '1' in DMAC_CTRLBx.AUTO bit" corrected to "automatic mode is disabled by clearing the DMAC_CTRLBx.AUTO bit" Section 31.6.6 "Disabling a Channel Prior to Transfer Completion": in last paragraph, "by writing a '1' to the DMAC_CHER.RESx field register" changed to "by setting the DMAC_CHDR.RESx bit" Section 31.6.6.1 "Abnormal Transfer Termination": - in first sentence, "channel enable bit, DMAC_CHDR.ENAx" corrected to "channel enable bit, DMAC_CHER.ENAx" - in second paragraph, "global enable bit in the DMAC Configuration Register (DMAC_EN.ENABLE bit)" corrected to "general enable bit in the DMAC Enable Register (DMAC_EN.ENABLE)" Section 31.6.7 "Register Write Protection": updated title (was "Write Protection Registers") and content

Doc. Rev. 11063L	Comments (Continued)
28-Oct-15	Section 31. "DMA Controller (DMAC)" (cont'd) Section 31.7 "DMAC Software Requirements": deleted bullet "When hardware handshaking interface protocol is fully implemented..." Removed reset values from individual register description sections (reset values are provided in Table 31-4 "Register Mapping") Section 31.8.16 "DMAC Channel x [x = 0..7] Control A Register": updated SCSIZE and DCSIZE field descriptions Updated Section 31.8.21 "DMAC Write Protection Mode Register" and Section 31.8.22 "DMAC Write Protection Status Register"
	Section 32. "USB Device Port (UDP)" Figure 32-1 "Block Diagram": appended caption "udp_int" with "(interrupt line)" Section 32.2 "Embedded Characteristics": added bullet "Integrated Pull-up on DDP" Section 32.4 "Product Dependencies": added paragraph "One I/O line may be used by the application to check that VBUS...then remove the pull-up." Updated Section 32.4.2 "Power Management" Updated Figure 32-2 "Board Schematic to Interface Device Peripheral" Updated Section 32.5.1 "USB Device Transceiver" and Section 32.5.2 "VBUS Monitoring" Table 32-5 "USB Transfer Events": restructured table and reorganized contents Section 32.6.2.3 "Data OUT Transaction": "FIFO Bank 0" changed to "FIFO Bank 1" in step 12 under "Using Endpoints With Ping-pong Attributes") Section 32.6.3 "Controlling Device States": in third paragraph, "may not consume more than 500 µA" changed to "must not consume more than 2.5 mA" Updated Section 32.6.3.2 "Entering Attached State" Section 32.6.3.6 "Entering in Suspend State": in second paragraph, "must drain less than 500uA" changed to "must drain no more than 2.5 mA" Figure 32-15 "Board Schematic to Drive a K State": "DM" changed to "DDM" Updated Figure 32.5.1 "USB Device Transceiver" Table 32-6 "Register Mapping": updated first footnote; replaced UDP_FDR0–UDP_FDR5 reset values with a footnote Section 32.7.7 "UDP Interrupt Status Register": in EPxINT bit configuration values, "Endpoint0 Interrupt" changed to "Endpointx Interrupt" Section 32.7.10 "UDP Endpoint Control and Status Register (CONTROL_BULK)": below warning, added text and updated code; updated EPTYPE field configuration values Section 32.7.11 "UDP Endpoint Control and Status Register (ISOCHRONOUS)": updated EPTYPE field configuration values
	Section 33. "USB Host Port (UHP)" Section 33.1 "Description": updated link to Hewlett-Packard website
	Section 34. "High Speed Multimedia Card Interface (HSMCI)" Section 34.1 "Description": in fourth paragraph, deleted sentence "Only one slot can be selected at a time (slots are multiplexed)" Section 34.14.12 "HSMCI Status Register": updated means of clearing a bit in relevant bit descriptions
	Section 35. "Serial Peripheral Interface (SPI)" Instances of "MCK" changed to "peripheral clock" Updated Section 35.2 "Embedded Characteristics" and Figure 35-1 "Block Diagram" Section 35.7.1 "Modes of Operation": updated text relating to Slave mode Updated Figure 35-3 "SPI Transfer Format (NCPHA = 1, 8 bits per transfer)" and Figure 35-4 "SPI Transfer Format (NCPHA = 0, 8 bits per transfer)" Updated Section 35.7.3 "Master Mode Operations"

© 2017 Microchip Technology Inc.

Doc. Rev. 11063L	Comments (Continued)
	Section 35. "Serial Peripheral Interface (SPI)" (cont'd)
	Section 35.7.3.2 "Master Mode Flow Diagram": updated Figure 35-7 "Master Mode Flow Diagram" and paragraph text
	Section 35.7.3.5 "Peripheral Selection": in last paragraph, replaced "If LASTXFER is used, the command must be issued before writing the last character" with "If LASTXFER is used, the command must be issued after writing the last character"
	Section 35.7.3.8 "Peripheral Deselection without DMA": in last paragraph, replaced "the Last Transfer (LASTXFER) bit in the SPI_MR must be set to 1 before writing the last data to transmit into the SPI_TDR" with "the Last Transfer (LASTXFER) bit in SPI_CR must be set after writing the last data to transmit into SPI_TDR"
	Section 35.7.4 "SPI Slave Mode": at end of section, inserted sentence "In this case the Underrun Error Status Flag (UNDES) is set in the SPI_SR."
	Section 35.7.5 "Register Write Protection": updated title (was "Write Protected Registers") and text
	Table 35-5 "Register Mapping": for SPI_CSRx, replaced fixed offset with equation; SPI_SR reset value 0x000000F0 changed to 0x0
	Section 35.8.1 "SPI Control Register": modified description of bit SPIDIS; added bits FIFODIS, FIFOEN, RXFCLR, TXFCLR and REQCLR
	Section 35.8.2 "SPI Mode Register": updated DLYBCS field description
	Section 35.8.5 "SPI Status Register": updated bit descriptions (SPIENS bit unchanged); added UNDES bit
	Added UNDES bit in Section 35.8.6 "SPI Interrupt Enable Register", Section 35.8.7 "SPI Interrupt Disable Register" and Section 35.8.8 "SPI Interrupt Mask Register"
	Section 35.8.9 "SPI Chip Select Register": updated descriptions of fields CSNAAT, SCBR, DLYBS, and DLYBCT
	Section 35.8.10 "SPI Write Protection Mode Register": removed reset line and updated WPEN bit description
	Section 35.8.11 "SPI Write Protection Status Register": removed reset line and updated WPVSRC field description
28-Oct-15	Section 36. "Timer Counter (TC)"
	Instances of "Master clock" or "MCK" changed to "peripheral clock"
	Updated Section 36.1 "Description" and Section 36.2 "Embedded Characteristics"
	Transferred Table 36-1 "Timer Counter Clock Assignment" from Section 36.1 "Description" to Section 36.3 "Block Diagram" and updated table footnote
	Updated Section 36.6.2 "32-bit Counter"
	Section 36.6.3 "Clock Selection": updated names of internal clock signals
	Section 36.6.12.1 "WAVSEL = 00": updated first paragraph
	Updated Figure 36-10 "WAVSEL = 10 without Trigger": and Figure 36-11 "WAVSEL = 10 with Trigger"
	Section 36.6.12.3 "WAVSEL = 01": updated first paragraph
	Updated Figure 36-14 "WAVSEL = 11 without Trigger" and Figure 36-15 "WAVSEL = 11 with Trigger"
	Added Section 36.6.15 "2-bit Gray Up/Down Counter for Stepper Motor" and Section 36.6.16 "Register Write Protection"
	Table 36-6 "Register Mapping": added Stepper Motor Mode Register and Write Protection Mode Register
	Section 36.7.2 "TC Channel Mode Register: Capture Mode": in 'Name' line, "(WAVE = 0)" changed to "(CAPTURE_MODE)"; inserted sentence on disabling write protection; updated TCCLKS field description
	Section 36.7.3 "TC Channel Mode Register: Waveform Mode": in 'Name' line, "(WAVE = 1)" changed to "(WAVEFORM_MODE)"; inserted sentence on disabling write protection; updated TCCLKS field description
	Added Section 36.7.4 "TC Stepper Motor Mode Register"
	Added sentence on disabling write protection in Section 36.7.7 "TC Register A", Section 36.7.8 "TC Register B", and Section 36.7.9 "TC Register C"
	Section 36.7.10 "TC Status Register": updated bit descriptions (CLKSTA bit unchanged)
	Section 36.7.15 "TC Block Mode Register": inserted sentence on disabling write protection; corrected TC2XC2S field configuration values (for value 2, TIOA1 corrected to TIOA0; for value 3, TIOA2 corrected to TIOA1)
	Added Section 36.7.16 "TC Write Protection Mode Register"

Doc. Rev. 11063L	Comments (Continued)
28-Oct-15	Section 38. "Two-wire Interface (TWI)" Instances of "shift register" changed to "internal shifter" Section 38.1 "Description": deleted sentence "Arbitration of the bus is performed internally and puts the TWIHS in Slave mode automatically if the bus arbitration is lost." Table 38-1 "Microchip TWI Compatibility with I2C Standard": "Clock Stretching" changed to "Clock Stretching/ Synchronization" Section 38.2 "Embedded Characteristics": added bullet "Register Write Protection" Updated Figure 38-1 "Block Diagram" Removed section "Application Block Diagram" Updated Section 38.5 "I/O Lines Description" Figure 38-21 "Multi-master Flowchart": deleted "TWI_CR = STOP" from "Stop Transfer" block Figure 38-26 "Clock Synchronization in Write Mode": replaced "SCL is stretched" with "TWCK is stretched" Section 38.7.3 "Master Mode": removed section "Application Block Diagram" Updated Section 38.7.3.2 "Programming Master Mode" and Section 38.7.3.3 "Master Transmitter Mode" Section 38.7.3.4 "Master Receiver Mode": removed "after the STOP condition" from end of second sentence in second paragraph; removed reference to clock stretching in the "Warning" Figure 38-9 "Master Read Wait State with Multiple Data Bytes": changed title (was "Master Read Clock Stretching with Multiple Data Bytes") and modified diagram to remove references to clock stretching Section 38.7.3.5 "Internal Address": under "10-bit Slave Addressing", removed reference to "Atmel AT24LC512 EEPROM" Section 38.7.3.6 "Using the DMA Controller": - updated step 9 under "Data Transmit with the DMA" - updated step 12 under "Data Receive with the DMA" Section 38.7.5 "Slave Mode": removed section "Application Block Diagram" Section 38.7.5.3 "Receiving Data": - under "Read Sequence", added note on clearing TXRDY flag - "Clock Synchronization Sequence": removed reference to TWI_THR - add "Clock Stretching Sequence" (references TWI_THR) Section 38.7.5.4 "Data Transfer": - "Clock Synchronization/Stretching": (previously entitled "Clock Synchronization") added details; corrected clock synchronization/stretching usage errors - "Clock Synchronization in Write Mode": at end of last sentence, changed "in Read mode" to "in Write mode" "EOSVACC" corrected to "EOSACC" in Figure 38-22 "Read Access Ordered by a Master" and Figure 38-23 "Write Access Ordered by a Master" "GCACC" corrected to "GACC" in Figure 38-24 "Master Performs a General Call" Section 38.7.5.5 "Using the DMA Controller": updated step 6 under "Data Transmit with the DMA" and under "Data Receive with the DMA" Figure 38-29 "Read Write Flowchart in Slave Mode": "SVREAD = 0" corrected to "SVREAD = 1"; "RXRDY= 0 ?" corrected to "RXRDY = 1 ?" Section 38.7.6 "Register Write Protection": changed title (was "Write Protection System"); updated content Table 38-7 "Register Mapping": removed TWI_THR reset value Removed line "Reset" from register description sections (redundant reset values in Table 38-7 "Register Mapping") Section 38.8.1 "TWI Control Register": updated MSEN and SVEN bit descriptions Section 38.8.2 "TWI Master Mode Register": updated START bit description

© 2017 Microchip Technology Inc.

Doc. Rev. 11063L	Comments (Continued)
28-Oct-15	Section 38. "Two-wire Interface (TWI)" (cont'd) Section 38.8.5 "TWI Clock Waveform Generator Register": updated CLDIV, CHDIV, and CKDIV field descriptions Section 38.8.6 "TWI Status Register": updated bit descriptions Section 38.8.12 "TWI Write Protection Mode Register": updated WPEN bit descriptions Section 38.8.13 "TWI Write Protection Status Register": updated WPVSRC field description
	Section 39. "Universal Synchronous Asynchronous Receiver Transmitter (USART)" Instances of 'MCK' changed to 'peripheral clock' in text, figures and equations Section 39.2 "Embedded Characteristics": added bullets "Register Write Protection" and "Digital Filter on Receive Line" Updated Figure 39-1 "USART Block Diagram" Removed section "Application Block Diagram" Section 39.5.1 "I/O Lines": deleted paragraph "To prevent the TXD line ..." Section 39.5.3 "Interrupt Sources": removed sentences: 'Note that it is not recommended to use the USART interrupt line in edge sensitive mode.' and 'Configuring the USART does not require the USART clock to be enabled.' Section 39.6 "Functional Description": removed list of peripheral characteristics (redundant with list in Section 39.2 "Embedded Characteristics") Updated Section 39.6.1 "Baud Rate Generator" (text and Figure 39-2 "Baud Rate Generator") Figure 39-3 "Fractional Baud Rate Generator": added label "Selected Clock" to USCLKS mux output; added CLKO = 0 and CLKO = 1 to SCK pins Section 39.6.1.1 "Baud Rate in Asynchronous Mode": updated baud rate calculation equation under "Baud Rate Calculation Example" At end of Section 39.6.1.2 "Fractional Baud Rate in Asynchronous Mode", added warning "When the value of field FP is greater than 0..." Section 39.6.1.3 "Baud Rate in Synchronous Mode or SPI Mode": updated second paragraph Table 39-7 "Possible Values for the Fi/Di Ratio": in top row, "774" corrected to "744" Section 39.6.3.3 "Asynchronous Receiver": reworded third paragraph Section 39.6.3.4 "Manchester Decoder": deleted paragraph referencing "RXIDLV bit" Section 39.6.3.8 "Parity": reworded third paragraph Updated Section 39.6.3.15 "Hardware Handshaking" Section 39.6.4.2 "Protocol T = 0": updated content under "Transmit Character Repetition" and "Disable Successive Receive NACK" Figure 39-36 "Example of RTS Drive with Timeguard": diagram modified with RTS rising edge prior to Start Bit Section 39.6.7.5 "Character Transmission": in first paragraph, "configured on INACK field" changed to "configured on the bit WRDBT"; inserted second paragraph "The chip select line is de-asserted for a period equivalent to three bits between the transmission of two data." Section 39.6.10 "Register Write Protection": updated title (was Write Protection Registers) and content Table 39-16 "Register Mapping": "Manchester Encoder Decoder Register" renamed to "Manchester Configuration Register"; US_MAN reset value '0xB0011004' corrected to '0x30011004'; added reset value '0x0' for US_MR, US_CSR, and US_NER Section 39.7.1 "USART Control Register": updated descriptions of bits STTTO, RSTIT, RETTO, RTSEN, and RTSDIS Section 39.7.3 "USART Mode Register": updated descriptions of fields USART_MODE, USCLKS, PAR, FILTER, and DSNACK Section 39.7.4 "USART Mode Register (SPI_MODE)": deleted CHMODE field description (CHMODE field is not present in bitmap); added CLKO bit Section 39.7.6 "USART Interrupt Enable Register (SPI_MODE)": added NSSE bit; updated descriptions of bits ENDRX, ENDTX, TXBUFE, and RXBUFF

Doc. Rev. 11063L	Comments (Continued)
28-Oct-15	Section 39. "Universal Synchronous Asynchronous Receiver Transmitter (USART)" (cont'd) Added NSSE bit in Section 39.7.9 "USART Interrupt Disable Register (SPI_MODE)" and Section 39.7.12 "USART Interrupt Mask Register (SPI_MODE)" Section 39.7.14 "USART Channel Status Register": updated bit descriptions Section 39.7.15 "USART Channel Status Register (SPI_MODE)": added bits NSSE and NSS; updated descriptions of bits RXRDY, TXRDY, OVRE, TXEMPTY, and UNRE Section 39.7.16 "USART Channel Status Register (LIN_MODE)": updated bit descriptions (LINBLS bit unchanged) Section 39.7.19 "USART Baud Rate Generator Register": updated descriptions of fields CD and FP Section 39.7.20 "USART Receiver Time-out Register": updated TO field description Section 39.7.21 "USART Transmitter Timeguard Register": updated TG bit description Section 39.7.22 "USART FI DI RATIO Register": deleted reset value; modified FI_DI_RATIO field size from 16 to 11 bits and updated field configuration values Section 39.7.29 "USART Write Protection Mode Register": removed "Reset" line; updated WPEN bit description Section 39.7.30 "USART Write Protection Status Register": removed "Reset" line; updated WPVSRC field description
	Section 40. "Universal Asynchronous Receiver Transmitter (UART)" Updated Figure 40-1 "UART Block Diagram" Added Table 40-3 "Peripheral IDs" Updated Section 40.5.1 "Baud Rate Generator" Updated text in Section 40.5.2.4 "Receiver Overrun" Section 40.6.9 "UART Baud Rate Generator Register": updated CD field description
	Section 41. "Analog-to-Digital Converter (ADC)" Replaced instances of 'MCK' with 'peripheral clock'; replaced instances of "ADCClock" with "ADC clock" or acronym "ADCCLK"; deleted references to ADC_SEQR2 Section 41.2 "Embedded Characteristics": '300 kHz' changed to '300 sps' in "Conversion Rate" bullet; bullet "Write Protect Registers" renamed to "Register Write Protection" Updated Figure 41-1 "Analog-to-Digital Converter Block Diagram with Touchscreen Mode" Section 41.5 "Product Dependencies": removed section "Analog Inputs" Updated text in Section 41.5.3 "I/O Lines" Revised Section 41.6.1 "Analog-to-Digital Conversion" Added Section 41.6.2 "ADC Clock" Section 41.6.3 "ADC Reference Voltage": changed title (was "Conversion Reference") Section 41.6.4 "Conversion Resolution": in 2nd paragraph, added sentence with details on reduced resolution levels Section 41.6.5 "Conversion Results": "ADC_SR" corrected to "ADC_ISR" Section 41.6.6 "Conversion Triggers": external trigger pin name "TSADTRG" corrected to "ADTRG" Figure 41-3 "EOCx and DRDY Flag Behavior": "ADC_SR" corrected to "ADC_ISR" Figure 41-4 "EOCx, OVREx and GOVREx Flag Behavior": updated title and warning text; "ADC_SR" corrected to "ADC_ISR" Section 41.6.6 "Conversion Triggers": inserted sentence "The hardware trigger can be one of the TIOA outputs of the Timer Counter channels or the external trigger input of the ADC (ADTRG)" Updated Section 41.6.7 "Sleep Mode and Conversion Sequencer" Section 41.6.10.2 "4-wire Resistive Touchscreen Principles": instance of "TSADC controller" changed to "ADC controller" Figure 41-11 "Insertion of Touchscreen Sequences (TSFREQ = 2; TSAV = 1)": added note to clarify ADC_SEL

© 2017 Microchip Technology Inc.

Doc. Rev. 11063L	Comments (Continued)
	Section 41. "Analog-to-Digital Converter (ADC)" (cont'd)
	Section 41.6.10.8 "Measured Values, Registers and Flags": deleted text fragment "for classic ADC conversions" from end of first paragraph; in fifth paragraph, "CALIB" corrected to "TSCALIB"
	Section 41.6.10.9 "Pen Detect Method": "ADC_SR" corrected to "ADC_ISR"
	Added Figure 41-13 "Buffer Structure"
	Section 41.6.12 "Register Write Protection": updated title and content
	Table 41-4 "Register Mapping": removed "Channel Sequence Register 2" (offset 0x0C now reserved)
	Section 41.7.1 "ADC Control Register": updated TSCALIB bit description
	Section 41.7.2 "ADC Mode Register": in bitmap, removed name "DIV1" from bit 22; updated PRESCAL field description
	Section 41.7.3 "ADC Channel Sequence 1 Register": updated USCHx field description
	Removed section "ADC Channel Sequence 2 Register"
	Section 41.7.4 "ADC Channel Enable Register": updated note in CHx bit description
	Section 41.7.5 "ADC Channel Disable Register": updated warning text in CHx bit description
	Section 41.7.11 "ADC Interrupt Status Register": updated bit descriptions (PENS bit description unchanged)
	Section 41.7.13 "ADC Extended Mode Register": in TAG bit description, "ADC_LDCR" corrected to "ADC_LCDR"
	Section 41.7.15 "ADC Channel Data Register": in 'DATA' field description, replaced "The Convert Data Register (CDR)" with "ADC_CDRx"
	Section 41.7.21 "ADC Trigger Register": updated TRGMOD field description
	Section 41.7.22 "ADC Write Protection Mode Register": updated WPEN bit description
	Section 41.7.23 "ADC Write Protection Status Register": updated WPVSRC field description
28-Oct-15	Section 42. "Synchronous Serial Controller (SSC)"
	Updated Figure 42-1 "Block Diagram"
	Figure 42-5 "Time Slot Application Block Diagram": removed arrowhead going into "CODEC Second Time Slot" from "Data In" line
	Section 42.8 "Functional Description": replaced instances of "Master Clock" with "peripheral clock"
	"MCK" changed to "Peripheral Clock" in Figure 42-6 "SSC Functional Block Diagram" and Figure 42-7 "Divided Clock Block Diagram"
	Figure 42-8 "Divided Clock Generation": "Master Clock" changed to "Peripheral Clock"; "MCK" changed to "$f_{peripheral\ clock}$"
	Section 42.8.5.1 "Frame Sync Data": at end of second paragraph, "has a maximum value of 16" changed to "has a maximum value of 256"
	Figure 42-15 "Receive Compare Modes": deleted instance of "Up to 16 bits (4 in this example)" under FSLEN label
	Section 42.8.6.1 "Compare Functions": at end of first sentence, "maximum value of 16 bits" changed to "maximum value of 256 bits"
	Table 42-4 "Data Frame Registers": replaced "Up to 16" with "Up to 256" as length for field FSLEN
	Section 42.8.10 "Register Write Protection": updated title and content
	Table 42-5 "Register Mapping": corrected reserved space (offset range 0x50–0xFC replaced by ranges 0x50–0xE0 and 0xEC–0xFC)
	Section 42.9.2 "SSC Clock Mode Register": updated DIV field description
	Section 42.9.6 "SSC Transmit Frame Mode Register": in FSOS field description, instances of "RF pin" corrected to "TF pin"
	Section 42.9.17 "SSC Write Protection Mode Register": removed "Reset" line (register reset values are provided in Table 42-5 "Register Mapping""); updated WPEN bit description
	Section 42.9.18 "SSC Write Protection Status Register": removed "Reset" line (register reset values are provided in Table 42-5 "Register Mapping"); updated WPVSRC field description

Doc. Rev. 11063L	Comments (Continued)
28-Oct-15	Section 43. "LCD Controller (LCDC)" Removed reset values from individual register description sections (reset values are provided in Table 43-31 "Register Mapping") Table 43-31 "Register Mapping": - removed reset value from write-only registers LCDC_BASECHER, LCDC_BASECHDR, LCDC_BASEIER, and LCDC_BASEIDR - added access Read/Write and reset value 0x00000000 for LCDC_BASECLUT255 - removed five registers LCDC_ADDRSIZE, LCDC_IPNAME1, LCDC_IPNAME2, LCDC_FEATURES, and LCDC_VERSION (offsets 0x1FEC–0x1FFC now reserved)
	Section 44. "Advanced Encryption Standard (AES)" Section 44.1 "Description": in second paragraph, "Peripheral DMA Controller" changed to "DMA Controller" Added Section 44.4.1 "AES Register Endianism" Updated Section 44.4.2 "Operation Modes" Section 44.4.5.1 "Manual and Auto Modes": updated content under "If AES_MR.LOD = 1" Section 44.4.5.2 "DMA Mode": updated content under "If AES_MR.LOD = 1" Section 44.5.2 "AES Mode Register": updated PROCDLY field description Section 44.5.6 "AES Interrupt Status Register": updated field descriptions
	Section 45. "Secure Hash Algorithm (SHA)" Harmonized register naming throughout Updated Section 45.4.4.3 "DMA Mode": Section 45.4.4.4 "SHA Register Endianism": rephrased first part of section; corrected values in the examples; corrected IO register numbering Table 45-2 "Register Mapping": reserved offset range 0x94-0xFC changed to 0xA0–0xFC Section 45.5.6 "SHA Interrupt Status Register": updated bit descriptions Section 45.5.8 "SHA Output Data Register x": updated ODATA field description
	Section 46. "True Random Number Generator (TRNG)" Updated names of referenced test suites in Section 46.1 "Description" and Section 46.2 "Embedded Characteristics" Section 46.5 "Functional Description": updated terminology in text and in Figure 46-2 "TRNG Data Generation Sequence" Table 46-2 "Register Mapping": defined offset ranges 0x04–0x0C, 0x20–0x4C, and 0x54–0xFC as reserved Removed the line "Reset" in Section 46.6.4 "TRNG Interrupt Mask Register", Section 46.6.5 "TRNG Interrupt Status Register", and Section 46.6.6 "TRNG Output Data Register" (reset values are provided in Table 46-2 "Register Mapping")
	Section 47. "Electrical Characteristics" Table 47-5 "Processor Clock Waveform Parameters": added footnote "With DDR2 usage. There are no limitations for LPDDR, SDRAM and mobile SDRAM." Table 47-12 "32 kHz Crystal Characteristics": added footnote "R_S is the equivalent series resistance." Table 47-13 "XIN32 Clock Electrical Characteristics": updated conditions Section 47.12 "Core Power Supply POR Characteristics": transferred subsections "Power Sequence Requirements" and "Power-Up Sequence" to Section 4. "Power Considerations" In Section 47.15.1.1 "Maximum SPI Frequency", updated description under "Master Write Mode" and "Master Read Mode" Section 47.15.2.1 "Timing Conditions": deleted sentence "These values may be product dependant and should be confirmed by the specification."

© 2017 Microchip Technology Inc.

Doc. Rev. 11063L	Comments (Continued)
28-Oct-15	Section 47. "Electrical Characteristics" (cont'd) Added Table 47-17 "PLLB Characteristics" Updated Table 47-18 "I/O Characteristics" Table 47-19 "Channel Conversion Time and ADC Clock": in footnote, "SHTIM field in TSADCC Mode Register" corrected to "TRACKTIM field in the ADC Mode Register" Table 47-22 "Transfer Characteristics": for Offset Error, unit "mV" corrected to "LSB" Table 47-26 "Zero Hold Mode Use Maximum System Clock Frequency (MCK)": specified "VDDIOM" as supply Table 47-29 "SMC Write Signals - NWE Controlled (WRITE_MODE = 1)": specified "VDDIOM" as supply Table 47-30 "SMC Write NCS Controlled (WRITE_MODE = 0)": specified "VDDIOM" as supply Figure 47-4 "SMC Timings - NCS Controlled Read and Write": corrected width of signal SMC12 Replaced single "SSC Timings" table with two updated tables Table 47-35 "SSC Timings 3.3V Domain" and Table 47-36 "SSC Timings 1.8V Domain"
	Section 48. "Mechanical Characteristics" Added Table 48-6 "BGA217 – Recommended Soldering Profile from J-STD-20" Added Table 48-12 "247-ball TFBGA – Recommended Soldering Profile from J-STD-20"
	Section 50. "Ordering Information" Table 50-1 "SAM9N12/CN11/CN12 Ordering Information": removed column "Package Type"
	Section 51. "Errata" Reorganized content into Section 51.1 "Errata: Revision A Parts" and Section 51.2 "Errata: Revision B Parts" Removed section "RSTC: Reset during SDRAM Accesses" from SAM9CN12 errata

Doc Revision 11063K	Comments	Change Request Ref.(1)
	Section 51. "Errata"	
	Section 51.4 "SAM9CN12 Errata: Revision B": Section 51.2.1.1 "Boot from SPI Data/Serial Flash Devices do not work with All Memories": Corrected data/serial Flash reference from AT25DF012 to AT25DF021	9429
	Deleted Section 51.3.2 "16 MHz Main Crystal" from Section 51.4 "SAM9CN12 Errata: Revision B".	9429

Doc. Rev 11063J	Comments	Change Request Ref.(1)
	Introduction:	
	"Description", revised content and presentation.	rfo
	Section "Features":	rfo
	- added a list on safety features	
	- moved SHA references to "Safety" list	
	Section 2. "Signal Description", removed references to VBG pins in Table 2-1, "Signal Description List" and Table 2-2, "SAM9N12/CN11/CN12 I/O Type Description".	8531

Doc. Rev 11063J	Comments (Continued)	Change Request Ref.(1)
	BSC:	
	Section 12.1 "Description", updated introduction text.	
	Section 12.4.1 "Boot Sequence Configuration Register":	9006/rfo
	- renamed the BOOTKEY bitfield to WPKEY	
	- WPKEY: replaced the bitfield description with a table	
	RSTC:	
	Section 13.2 "Embedded Characteristics", removed the "AMBA™-compliant Interface" bullet and its subbullet.	9043
	Section 13.4.4.4 "Software Reset", removed "Except for Debug purposes,..." from the "PERRST" bullet - Section 13.5.1 "Reset Controller Control Register":	9017
	- EXTRST: updated the bitfield description	8271
	- KEY: replaced the bitfield description with a table	9043
	Section 13.5.2 "Reset Controller Status Register":	
	- RSTTYP bitfield description table: updated column headings to "Value/Name/Description", replaced binary configuration values with decimal values, and updated reset type names to 'GENERAL_RST', etc.	9043
	Section 13.5.1 "Reset Controller Control Register":	
	- KEY: replaced the bitfield description with a table	9043
	RTC:	
	Updated Section 14.2 "Embedded Characteristics".	8544
	Section 14.5.3 "Alarm", added a paragraph on SECEN, MINEN, HOUREN enabling with the corresponding note.	8900/ 9027
	Added the same note in Section 14.6.5 "RTC Time Alarm Register" and Section 14.6.6 "RTC Calendar Alarm Register".	
	WDT:	
	Updated Section 16.1 "Description".	8429
	Section 16.2 "Embedded Characteristics", added the 2nd bullet and removed AMBA references.	
	Section 16.4 "Functional Description", added the 4th paragraph 'If the watchdog is restarted...' and added the corresponding notes in Section 16.5.2 "Watchdog Timer Mode Register".	8218 8128/rfo
	Section 16.5.1 "Watchdog Timer Control Register":	8844/rfo
	- WDRSTT: added details in the bitfield description	
	- KEY: replaced the bitfield description with a table	
	SHDWC:	
	Section 17.2 "Embedded Characteristics", removed AMBA references.	rfo
	Section 17.7.1 "Shutdown Control Register":	8869
	- KEY: replaced the bitfield description with a table	
	Section 17.7.2 "Shutdown Mode Register":	8432
	- WKMODE0 bitfield description table: added a column and updated other column headings to "Value/ Name/ Description", replaced binary configuration values with decimal values	
	- CPTWK0: updated the bitfield description	
	SCKC:	
	Section 19.3 "Block Diagram", updated the location of RCEN, OSC32EN, OSCSEL and OSC32BYP bits from Slow Clock Control Register (SCKCR) to Slow Clock Configuration Register (SCKC_CR).	8322

© 2017 Microchip Technology Inc.

Doc. Rev 11063J	Comments (Continued)	Change Request Ref.(1)
	PMC:	
	Section 20.5 "Main Clock":	
	- added Section 20.5.4 "Switching Main Clock between the Main RC Oscillator and Fast Crystal Oscillator"	9245
	- Section 20.5.5 "Software Sequence to Detect the Presence of Fast Crystal": added details on the Main RC oscillator (must be kept enabled) to the sequence introduction phrase.	
	PIO:	
	Section 22.2 "Embedded Characteristics", removed a bullet on configuration lock.	8909
	Section 22.4.4 "Interrupt Generation", revised content in the 1st paragraph.	8324
	Section 22.5 "Functional Description":	
	- added pull-down resistor and the corresponding registers in Figure 22-3, "I/O Line Control Logic"	8522
	- Section 22.5.14 "Register Write Protection":	
	- changed the section title and revised the content	rfo
	Section 22.7 "Parallel Input/Output Controller (PIO) User Interface":	
	- Table 22-2 "Register Mapping":	
	- removed references to PIO Lock Status Register (PIO_LOCKSR) and assigned the corresponding offset 0x00E0 to reserved registers	8909
	- Section 22.7.29 "PIO Slow Clock Divider Debouncing Register":	
	- DIV: updated the field name from DIVx to DIV	rfo
	- removed the "PIO Lock Status Register" section	8909
	- Section 22.7.45 "PIO Write Protection Mode Register":	
	- modified the register name	rfo
	- WPEN: replaced the list of protectable registers with a cross-reference to Section 22.5.14 "Register Write Protection"	rfo
	- WPKEY: replaced the bitfield description with a table	8522
	- Section 22.7.46 "PIO Write Protection Status Register":	
	- modified the register name	rfo
	- WPVSRC: updated the bitfield description and removed a note	rfo
	- Section 22.7.47 "PIO Schmitt Trigger Register"	
	- SCHMITTx [x=0..31]: added description of the bit name	rfo
	- Section 22.7.48 "PIO I/O Delay Register"	
	- Delayx [x=0..7]: updated the bit name from 'Delay x' to 'Delayx [x=0..7]' and added name description	rfo
	DBGU:	
	Section 23.6.10 "Debug Unit Chip ID Register":	8686
	- EPROC bitfield description table: removed the last row (value 6)	
	FUSE:	
	Section 24.5 "Fuse Controller (FUSE) User Interface":	8065/rfo
	- Table 24-1 "Register Mapping", updated FUSE_CR access from 'Read-write' to 'Write-only'	
	- Section 24.5.1 "Fuse Control Register", replaced KEY field description with a table	
	MATRIX	
	Section 25.10.6 "Write Protect Mode Register":	rfo
	- WPKEY: replaced the bitfield description with a table	

Doc. Rev 11063J	Comments (Continued)	Change Request Ref.(1)
	EBI: Added titles to figures in Section 26.7.4 “Power Supplies” and Section 26.8 “Implementation Examples”. Added Figure 26-13 "16-bit NAND Flash with NFD0_ON_D16 = 1".	rfo
	PMECC: Section 27.2 ”Embedded Characteristics”, added a line on supporting 8-bit Nand Flash data bus. Section 27.6.11 ”PMECC Interrupt Status Register”, fixed a typo in the register table: - replaced duplicate bits 31 - 24 by missing 7 - 0	8403
	SMC: Replaced ‘turned out’ with ‘switched to output mode ‘ in Section 29.9.4.1 ”Write is Controlled by NWE (WRITE_MODE = 1)” and Section 29.9.4.2 ”Write is Controlled by NCS (WRITE_MODE = 0)”.	7925
	DDRSDRC: Replaced TCSR with TCR when related to Section 30.7.7 ”DDRSDRC Low-power Register” to prevent confusion between JEDEC naming TCSR and the associated bitfield TCR. Replaced binary configuration values with decimal values throughout the document. Section 30.7 ”DDR SDR SDRAM Controller (DDRSDRC) User Interface”: - Table 30-16 ”Register Mapping”, updated offset values for reserved registers (0x54-0xE0, 0xEC-0xFC) - Section 30.7.3 ”DDRSDRC Configuration Register”: - DIC: updated the bitfield name from DIC/DIS to DIC and revised the description content - Section 30.7.11 ”DDRSDRC Write Protect Mode Register”: - WPKEY: replaced the bitfield description with a table - described with a table and/or updated data presentation in: - Section 30.7.1 ”DDRSDRC Mode Register” (MODE) - Section 30.7.3 ”DDRSDRC Configuration Register” (NC, NR, OCD, NB, DECOD) - Section 30.7.4 ”DDRSDRC Timing Parameter 0 Register” (TWTR) - Section 30.7.7 ”DDRSDRC Low-power Register”(LPCB, TIMEOUT, APDE, UPD_MR) - Section 30.7.8 ”DDRSDRC Memory Device Register” (MD, DBW)	8592 rfo 8968 8592 8883 8883
	DMAC: Added Section 31.3 ”DMA Controller Peripheral Connections” and moved Table 31-1 ”DMA Channel Definition” from Section 31.2 ”Embedded Characteristics” to this new section. Section 31.8.1 ”DMAC Global Configuration Register”: - KEY: replaced the bitfield description with a table Section 31.8.15 ”DMAC Channel x [x = 0..7] Descriptor Address Register”: - DSCR_IF bitfield description table: added DMA Master Interface references Section 31.8.17 ”DMAC Channel x [x = 0..7] Control B Register”: - SIF bitfield description table: added DMA Master Interface references - DIF bitfield description table: added DMA Master Interface references Section 31.8.21 ”DMAC Write Protect Mode Register”: - WPKEY: replaced the bitfield description with a table	8955 8835 rfo 8834

© 2017 Microchip Technology Inc.

Doc. Rev 11063J	Comments (Continued)	Change Request Ref.(1)
	UDP:	
	Section 32.2 "Embedded Characteristics", removed references on ARM7TDMI, ARM9TDMI and AMBA from the features list.	8219
	Added Section 32.7.11 "UDP Endpoint Control and Status Register (Isochronous Endpoints)" describing UDP_CSRx (ISOENDPT) alternate register.	8414
	HSMCI:	
	Section 34.8 "High Speed MultiMedia Card Operations":	9012
	- corrected HSMCI_MR to HSMCI_BLKR when referring to Block Length field that is not available in HSMCI_MR and removed Note 2 in Figure 34-8, "Read Functional Flow Diagram" and Figure 34-9, "Write Functional Flow Diagram"	
	Section 34.8.1 "Command - Response Operation":	rfo
	- reorganized table content in ALL_SEND_CID command example to eliminate empty cells in header row	
	Section 34.13 "Register Write Protection":	rfo
	- changed the section title (was "Write Protection Registers")	
	- revised content	
	Section 34.14 "High Speed MultiMedia Card Interface (HSMCI) User Interface":	
	- harmonized register access type naming throughout the section ("Read-write" change to "Read/Write", "Read" changed to "Read-only", "Write" changed to "Write-only")	rfo
	Section 34.14.7 "HSMCI Block Register":	8868
	- BLKLEN: removed the sentence on this bit accessibility in HSMCI Mode Register	
	Section 34.14.18 "HSMCI Write Protection Mode Register":	rfo
	- modified the register name (was HSMCI Write Protect Mode Register)	
	- HSMCI_WPMR register table: replaced empty bitfields 7 - 0 with dashes	
	- WPEN: replaced the list of protectable registers with a cross-reference to Section 34.13 "Register Write Protection"	
	- WPKEY: replaced the bitfield description with a table	
	Section 34.14.19 "HSMCI Write Protection Status Register":	rfo
	- modified the register name (was HSMCI Write Protect Status Register)	
	- updated description	

Doc. Rev 11063J	Comments (Continued)	Change Request Ref.(1)
	SPI:	
	Section 35.2 ”Embedded Characteristics”:	8210/
	- removed redundant text line (1st bullet)	8541
	- added SPCK references	
	- updated the line on Programmable Transfer Delay	
	Section 35.7.4 ”SPI Slave Mode”, updated the last sentence.	8792
	Section 35.8 ”Serial Peripheral Interface (SPI) User Interface”, updated offset values for reserved registers in Table 35-5 ”Register Mapping”.	8840
	Section 35.8.10 ”SPI Write Protection Mode Register”:	8840
	- removed ‘SPI’ prefix in bitfield names (SPIWPKEY --> WPKEY, SPIWPEN --> WPEN)	
	- WPEN: added details on Disable/Enable conditions and the list of protected registers	
	- WPKEY: replaced the bitfield description with a table	
	Section 35.8.11 ”SPI Write Protection Status Register”:	8136
	- removed ‘SPI’ prefix in bitfield names (SPIWPVSRC --> WPVSRC, SPIWPVS --> WPVS)	
	- WPVS: replaced the bitfield description table with the corresponding text (simplified)	
	TC:	
	Section 36.7 ”Timer Counter (TC) User Interface”:	rfo
	- changed the order of register description sections to match Table 36-5 ”Register Mapping”	
	Section 36.7.2 ”TC Channel Mode Register: Capture Mode”:	9107
	- TCCLKS: added details for values 0 - 4 in the bitfield description table	
	Section 36.7.3 ”TC Channel Mode Register: Waveform Mode”:	8885/
	- TCCLKS: added details for values 0 - 4 in the bitfield description table	9107
	- ENETRG: added a note on TIOA and TIOB controled by a selected external event	
	PWM:	8105
	Section 37.5.2 ”Power Management”, replaced the 2nd paragraph with a new content	

© 2017 Microchip Technology Inc.

Doc. Rev 11063J	Comments (Continued)	Change Request Ref.(1)
	TWI: Section 38.6.2 "Power Management", removed erroneous bullet "Enable the peripheral clock". Section 38.8 "Master Mode": - Section 38.8.6.1 "7-bit Slave Addressing", replaced 'N' acronym with 'NA' in Table 38-6 "Abbreviations". - Section 38.8.7.1 "Data Transmit with the DMA": - added Steps 6 - 9 - Section 38.8.7.2 "Data Receive with the DMA": - replaced the acronym 'PDC' with 'DMA' in the 1st paragraph - added a paragraph on slave mode - updated Step 2 - added Step 4 and Step 12 - Section 38.8.9 "Read-write Flowcharts", added missing "yes" and "no" in: - Figure 38-18, "TWI Write Operation with Multiple Data Bytes with or without Internal Address" - Figure 38-21, "TWI Read Operation with Multiple Data Bytes with or without Internal Address" Section 38.10 "Slave Mode" - added Section 38.10.6 "Using the DMA Controller" including subsections on data transmit and data receive	rfo rfo 8555/ 8552/ rfo 8944/rfo 9055
	Section 38.11 "Write Protection System": - replaced the acronym of the TWI Write Protection Status register with a cross-reference to the corresponding section (the acronym changed: TWI_WPROT_STATUS --> TWI_WPSR) - renamed bitfields of the write protection registers (WPROTERR --> WPVSRC, WPROTADDR --> WPVSRC, SECURITY_CODE --> WPKEY) Section 38.12 "Two-wire Interface (TWI) User Interface": - Table 38-7 "Register Mapping", added an offset for reserved registers (0x38-0xE0). - Section 38.12.5 "TWI Clock Waveform Generator Register", fixed typos. - Section 38.12.6 "TWI Status Register", replaced the description of "NACK: Not Acknowledged (clear on read)", used in master mode, with a new text (value "1", address byte is now referenced too) - Section 38.12.11 "TWI Transmit Holding Register", changed the register access to Write-only - Section 38.12.12 "TWI Write Protection Mode Register" - changed the register acronym: TWI_WPROT_MODE --> TWI_WPMR - renamed bitfields: WPROT --> WPEN, SECURITY_CODE --> WPKEY - WPROT/WPEN: added details on Disable/Enable conditions and the list of protected registers - SECURITY_CODE/WPKEY: replaced the bit description with a table - Section 38.12.13 "TWI Write Protection Status Register" - change the register acronym: TWI_WPROT_STATUS --> TWI_WPSR - renamed bitfields: WPROTERR --> WPVSRC, WPROTADDR --> WPVSRC - updated entirely the description of bitfields	8845 8814 9145 9050 9050 9050

Doc. Rev 11063J	Comments (Continued)	Change Request Ref.(1)
	USART:	
	Section 39.3 "Block Diagram" and Section 39.7.7 "SPI Mode":	9356
	- moved Table 40-1. SPI Operating Modes to Section 39.7.7.1 "Modes of Operation" (now Table 39-13 "SPI Operating Mode")	
	Section 39.7.1 "Baud Rate Generator", replaced "or 6" with "or 6 times lower" in the last phrase of the introduction text.	rfo
	Section 39.7.3.8 "Parity", corrected Figure 39-22, "Parity Error" for stop bit value.	8943
	Section 39.7.3.10 "Transmitter Timeguard", updated the Baud Rate value from "33,400" to "38,400" in Table 39-9 "Maximum Timeguard Length Depending on Baud Rate".	
	Section 39.7.3.11 "Receiver Time-out", updated the Baud Rate value from "33,400" to "38,400" in Table 39-10 "Maximum Time-out Period".	
	Section 39.7.5.3 "IrDA Demodulator", added a paragraph on IRDA_FILTER programming criteria.	8508
	Section 39.7.8.16 "LIN Frame Handling With the DMAC", removed abundant "DMAC" acronym in the 1st paragraph.	
	Section 39.8 "Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface":	
	- Table 39-17 "Register Mapping":	
	- added a row for LIN Baud Rate Register (offset 0x005C)	8445/
	- updated offset values and added a new row for reserved registers (0x0060-0x00E0, 0x00EC-0x00FC)	8943
	- added Section 39.8.28 "USART LIN Baud Rate Register"	8445
	- Section 39.8.22 "USART FI DI RATIO Register":	8643
	- US_FIDI register table: expanded FI_DI_RATIO bitfield to 16 bits	
	- Section 39.8.24 "USART IrDA FILTER Register":	8508
	- IRDA_FILTER: replaced the bitfield description with a new content including IRDA_FILTER programming criteria	
	- Section 39.8.29 "USART Write Protect Mode Register":	8791
	- WPKEY: replaced the bitfield description with a table	
	UART:	
	Section 40.2 "Embedded Characteristics", removed a redundant bullet on UART compatible features	8326
	Section 40.4.3 "Interrupt Source", replaced "NVIC" with a generic term "Interrupt Controller".	
	Section 40.6 "Universal Asynchronous Receiver Transmitter (UART) User Interface":	7967
	- Table 40-3 "Register Mapping": updated offset values and added a new row for reserved registers (0x0040-0x00E8, 0x00EC-0x00FC)	

© 2017 Microchip Technology Inc.

Doc. Rev 11063J	Comments (Continued)	Change Request Ref.(1)
	ADC:	
	Section 41.1 "Description", updated text in the entire section.	8509/rfo
	Section 41.6.2 "Conversion Reference", added details on reference voltage in the 1st sentence.	8385/rfo
	Section 41.6.4 "Conversion Results", removed "...and EOC bit corresponding to the last converted channel" from the last phrase of the third paragraph.	8357
	Section 41.6.5 "Conversion Triggers", added a title to Figure 41-5, "Hardware Trigger Delay".	8997
	Section 41.5.6 "Conversion Performance", updated references to electrical characteristics of the product.	
	Section 41.7.11 "Write Protected Registers", added ADC Analog Control Register in the list of write-protected registers.	8583
	Section 41.8 "Analog-to-Digital Converter (ADC) User Interface":	
	- removed duplicate information on registers not listed in Table 41-4 "Register Mapping"	rfo
	- Section 41.8.2 "ADC Mode Register", removed FWUP bitfield	8461/rfo
	- Section 41.8.15 "ADC Compare Window Register":	8045
	- LOWTHRES: added details in the bitfield descriptions on programming conditions	
	- HIGHTHRES: added details in the bitfield descriptions on programming conditions	
	- Section 41.8.19 "ADC Touchscreen X Position Register":	8229
	- ADC_XPOSR register table: expanded XPOS and XSCALE bitfields (bits 0-11 and 16-27 respectively)	
	- Section 41.8.20 "ADC Touchscreen Y Position Register":	
	- ADC_YPOSR register table: expanded YPOS and YSCALE bitfields (bits 0-11 and 16-27 respectively)	
	- Section 41.8.21 "ADC Touchscreen Pressure Register":	
	- ADC_PRESSR register table: expanded Z1 and Z2 bitfields (bits 0-11 and 16-27 respectively)	
	- Section 41.8.23 "ADC Write Protect Mode Register":	8583/
	- WPEN: added ADC Analog Control Register in the list of write-protected registers	8856
	- WPKEY: replaced the bitfield description with a table	
	SSC:	
	Section 42.9.14 "SSC Interrupt Enable Register":	8993
	- TXRDY: fixed a typo ('0 = 0 =' --> '0 = ')	
	Section 42.9.17 "SSC Write Protect Mode Register":	8841
	- WPKEY: replaced the bitfield description with a table	
	AES:	
	Section 44.6.2 "AES Mode Register":	8859
	- CKEY: replaced the bitfield description with a table	
	Section 44.6.10 "AES Initialization Vector Register x":	8892
	- IV: added details on CBC, OFB, CFB, and CTR modes in the bitfield description	
	SHA:	
	Section 45.5 "Secure Hash Algorithm (SHA) User Interface":	8216
	- Table 45-2 "Register Mapping": changed the reset value for SHA_MR from 0x1 to 0x0000100	

Doc. Rev 11063J	Comments (Continued)	Change Request Ref.(1)
	TRNG: Added new sections: - Section 46.3 ”Block Diagram” - Section 46.4 ”Product Dependencies” including subsections - Section 46.5 ”Functional Description” Section 46.1 ”Description”, moved 2nd and 3d paragraphs to Section 46.5 ”Functional Description” and added a paragraph on TRNG as an entropy source. Section 46.2 ”Embedded Characteristics”: - added a bullet on TRNG as an entropy source - moved Figure 46-2, "TRNG Data Generation Sequence" to Section 46.5 ”Functional Description”	8567
	Electrical Characteristics: Section 47.1 “Absolute Maximum Ratings”: - removed operating temperature references from Table 47-1 ”Absolute Maximum Ratings*” Section 47.2 “DC Characteristics”: - added operating temperature references to Table 47-2 ”DC Characteristics” - removed references to VDDCORE ripple, VDDBU ripple, VDDPLL ripple, and VDDOSC ripple parameters Added titles to Figure 47-2 "Main Oscillator Schematic" and Figure 47-3 "32 kHz Oscillator Schematic". Section 47.12 “Core Power Supply POR Characteristics”: - added Section 47.13 “Power Sequence Requirements” - added Section 47.13.1 “Power-Up Sequence”	rfo rfo rfo rfo 8807/rfo
	Ordering Information: Table 50-1, “SAM9N12/CN11/CN12 Ordering Information”: - added MRL and Carrier Type columns - added ordering codes for MRLB	8804
	ERRATA: Added Section 51.2.2 “12 MHz RC Oscillator”. Updated Section 51.1.1.1 “Boot from SPI Serial Flash Devices (xx25xxx) Is not Working”. Updated the section structure and added Section 51.4 ”SAM9CN12 Errata: Revision B”.	9164 9051 rfo

Doc. Rev 11063I	Comments	Change Request Ref.
	PMC: Note added in MAINFRDY field in Section 21.12.8 “PMC Clock Generator Main Clock Frequency Register”.	8870

© 2017 Microchip Technology Inc.

Doc. Rev 11063I	Comments	Change Request Ref.
	DMAC: Text updated in Section 31.1 “Description”, Section 31.2 “Embedded Characteristics”, Section 31.5.4.3 “Ending Multi-buffer Transfers”, Section 31.6 “DMAC Software Requirements”.	8441
	Ordering Information: In Table 50-1, “SAM9N12/CN11/CN12 Ordering Information”, BGA247 package ordering codes added.	8804
	Errata: Section 51.2.1 “LCD Controller (LCDC)” deleted. Section 51.1.2 “16 MHz Main Crystal - SAM9CN12 - Rev. A” added.	8804

Doc. Rev 11063H	Comments	Change Request Ref.(1)
	PMC: Reset value of CKGR_MOR register updated to 0x0000_0008 in Table 21-3, “Register Mapping”.	8447
	PMERRLOC: SIGMAN replaced with SIGMAx in Section 28.5.10 “Error Location SIGMAx Register”.	8339
	HSMCI: In Section 34.14.7 “HSMCI Block Register”, replaced BNCT bitfield table with the corresponding description and updated warning note. Table updated in Section 34.14.16 “HSMCI DMA Configuration Register”. Replaced references to advanced interrupt controller/AIC with “interrupt controller” in Section 34.6.3 “Interrupt”.	8431 rfo
	TWI: In Section 38.1 “Description”, removed “20” at the end of the 1st paragraph. Add-on for PDC/DMA transfer in Section 38.8.7 “Using the DMA Controller”.	7921 8426
	AES: Information on processing files greater than 1 megabyte added in Section 44.4.1 “Operation Modes”. Typo fixed in Section 44.4.3.1 “Manual Mode”. Removed units in the Chunk Size column in Table 44-3, “DMA Data Transfer Type for the Different Operation Modes”.	7966 8389 rfo
	Electrical Characteristics: Section 47.11 “USB Transceiver Characteristics” and Section 47.15.5 “UDP” added.	8504
	Errata: Section 51.2.1 “LCD Controller (LCDC)” added. Section 51.2.1 “LCD Controller (LCDC)” added.	7996 8321

Doc. Rev 11063G	Comments	Change Request Ref.[1]
	Overview: Added “Write Protected Registers in Section “Features”. Product name updated to SAM9N12/SAM9CN11/SAM9CN12. “Description” updated with the various devices configurations. Bullets for SAM9CN11 and SAM9N12 added in Section 6.3 “Chip Identification”.	 8213 8244
	Boot Strategies: Boot Strategy from SAM9CN12 removed to create the separate Secure Boot document, and replaced by the previous Boot Strategies from SAM9N12. Table 11-1, “External Clock and Crystal frequencies allowed for Boot Sequence (in MHz)” added in Section 11.2.3 “Chip Setup”.	 8202 8270
	RSTC: RSTC conditions improved.	 8083
	HSMCI: Sentence "This flag must be used only for Write Operations” removed in “NOTBUSY: HSMCI Not Busy” on page 616.	 8394
	USART: Whole chapter updated.	 rfo
	SSC: Reworked tables and bitfield descriptions in Section 42.9.3 “SSC Receive Clock Mode Register”, Section 42.9.4 “SSC Receive Frame Mode Register”, Section 42.9.5 “SSC Transmit Clock Mode Register”, Section 42.9.6 “SSC Transmit Frame Mode Register”. Replaced AIC/NVIC wording with “interrupt controller”.	 8466
	AES: Hardware Counter Measures updated in Section 44.2 ”Embedded Characteristics” and in Section 44.5.1 ”Unspecified Register Access Detection”.	 rfo
	SHA: Mode Register reset value updated to 0x1 in Table 45-2, “Register Mapping”.	 rfo
	Ordering Information: Ordering codes added for SAM9N12 and SAM9CN11.	 8244
	Errata: Errata created. Section 49.3 “Marking” moved to Section 51.2 “SAM9N12/CN11/CN12 Errata” on page 1112.	 rfo
	Back page: Date updated.	 rfo

© 2017 Microchip Technology Inc.

Doc. Rev 11063F	Comments	Change Request Ref.[1]
	Description: Section 1. “Description”, 125 MHz --> 133 MHz “FIPS PUB 46-3 compliant TDES” removed from 3rd paragraph	 7928 rfo
	Signal Description: Table 2-1, “Signal Description List”, NFD0-NFD16 --> NFD0-NFD15	rfo
	Power Considerations: Section 5.2 Programmable I/O Lines Power Supplies and Current Drive removed from Section 4. “Power Considerations”, as the same contents already exists in Section 26.7.4 “Power Supplies”	rfo
	System Controller: Section 6.3 “Chip Identification”, Chip ID: 0x819A_07A0 --> 0x819A_07A1	rfo
	Peripherals: Table 7-1, “SAM9N12/CN11/CN12 Peripheral Identifiers”:Replaced keyword ‘Reserved’ on 4th row with ‘FUSE’	8039
	EBI: Section 26.7.4 “Power Supplies”, following sentences added before the 2nd figure: “This can be used if the SMC connects to the NAND Flash only. Using this function with another device on the SMC will lead to an unpredictable behavior of that device. In that case, a default value must be selected.”	8008
	FUSE: Section 24. “Fuse Controller (FUSE)” added.	7928
	MATRIX: Section 25.10.5.1 “EBI Chip Select Assignment Register”, NFD0_ON_D16 bitfield description updated	8008
	PMC: Section 20.2 “Embedded Characteristics”, 266 MHz DDR --> 133 MHz DDR Section 21.8 “Peripheral Clock Controller”, PMC_PCR, 0x10030102 --> PMC_PCR,0x10031002	 7975 7920
	Electrical Characteristics: Table 47-5, “Processor Clock Waveform Parameters” and Table 47-6, “System Clock Waveform Parameters”, ‘Corner MAX’ changed to ‘VVDDCORE min’ and second row removed. In the note below, ‘LDDDR’ changed to ‘LPDDR’ Table 47-9, “XIN Clock Electrical Characteristics”, VIN row split into 2 rows: V_{XINLOW} and $V_{XINHIGH}$ Section 47.13 “SMC Timings”, “SMC Timings are given for MAX corners” removed Table 47-19, “Channel Conversion Time and ADC Clock” : ‘ADC Clock = 5 MHz’ row added to Conversion Time (TCT) and to ‘Throughput Rate’ Table 47-22, “Transfer Characteristics”, 2 rows added: ‘ADC Clock = 13.2 MHz’ and ‘ADC Clock = 5 MHz’	rfo 8009 rfo 7947

Doc. Rev 11063E	Comments	Change Request Ref.
	Overview: "Description", updated..."Processor running up to 400 MHz..." updated..."System running up to 133 MHz..."	7847
	DDRSDRC: Former Section 29.7 "Programmable IO Delays" removed from datasheet.	7891
	PIO: Section 22.5.11 "Programmable I/O Delays", "Only PADs PA[15:11] and PA[20:18] can be configured." Section 22.5.12 "Programmable I/O Drive", "It is possible to configure the I/O drive for pads PA[31:0], PB[18:0] and PC[31:0]."	7886
	PMC: Section 20.2 "Embedded Characteristics", updated, "266 MHz DDR system clock". Section 21.12.8 "PMC Clock Generator Main Clock Frequency Register", added RCMEAS bit to register.	7874 7726
	Electrical Characteristics: Table 47-3, "Power Consumption for Different Modes", • Updated, Active mode power consumption, 103 mA • Updated, Idle mode power consumption, 33 mA Table 47-5, "Processor Clock Waveform Parameters" • Updated, MAX = 400 MHz Table 47-6, "System Clock Waveform Parameters" Updated, MAX = 133 MHz	7847
	Section 46.14.5 Two-wire Serial Interface Characteristics removed. Footnotes updated in Table 47-35, "SSC Timings 3.3V Domain"	7863
	Back page: Updated point of contact information.	Marcom

Doc. Rev 11063D	Comments	Change Request Ref.(1)
	Overview: Section 5. "Memories", ...Internal ROM... bootstrap routine, revised. Section 7.5 "Fuse Box Features", removed table 8.3	rfo
	Debug and Test: Section 9.6.3 "Debug Unit", removed unnecessary line on Chip ID Section 9.6.5 "JTAG ID Code Register", fixed typo in title, revised part number and JTAG ID Code value.	rfo
	DMAC: Section 31.2 "Embedded Characteristics", missing elements recovered.	7271
	TRNG: Section 46. "True Random Number Generator (TRNG)", faulty section number corrected. Subsequent section numbering and TOC affected.	rfo:

© 2017 Microchip Technology Inc.

Doc. Rev 11063C	Comments	Change Request Ref.(1)
	Overview:	
	"Description" SLC NAND Flash is supported.	rfo
	Section 1. "Description", 1st paragraph, the 2nd sentence was removed.	
	Table 3-1, "BGA217 Pin Description", table updated with values in Ball column.	
	Table 4-1, "SAM9N12/CN11/CN12 Power Supplies", VDDFUSE Voltage Range updated, 3.0V-3.6V.	7395
	Section 5.3.3 "DDR-SDRAM Controller", revised.	rfo
	Section 6.3 "Chip Identification", removed "two" lines.	7269
	Section 7.4 "Peripheral Signal Multiplexing on I/O Lines", removed irrelevant text.	rfo
	Elsewhere, minor grammar revisions. Advance Information status moved to Preliminary.	
	ARM Processor:	rfo:
	Section 8. "ARM926EJ-S Processor Overview", removed Tightly-Coupled Memory Interface chapter.	
	Debug and Test:	
	Figure 9-1, Debug and Test Block Diagram, removed PDC.	rfo
	Boot Program:	
	Figure 11-1, ROM Code Algorithm Flow Diagram, updated.	7304
	Section 11.2 "SAM9CN11 and SAM9N12 Only", and forward, grammar and format edits.	rfo
	ADC:	
	Section 41.8.12 "ADC Interrupt Status Register", fixed ADC_SR typos to ADC_ISR.	7249-7250
	Section 41.8.14 "ADC Extended Mode Register", values 2 and 3 swapped in CMPMODE bitfield table.	7313
	Section 41.8.16 "ADC Channel Data Register", DATA bitfield extended to fields 11 and 10.	rfo
	Section 41.6.5 "Conversion Triggers", TRGMOD bitfield refers to Section 41.8.22 "ADC Trigger Register".	
	AES:	
	Section 44.5.1 "Unspecified Register Access Detection", updated.	7357
	AIC:	
	Section 10.9 "Write Protection Registers" added to datasheet.	7045
	"SRCTYPE: Interrupt Source Type" on page 71 bitfield description table updated.	7144
	"PRIOR: Priority Level" on page 71, bitfield described in a table.	7191
	DDRSDRC:	
	Section 30.2 "Embedded Characteristics",removed... "eight internal banks not supported."	7396
	Section 30.5.4.1 "Self-refresh Mode" UDP_EN replaced by UPD_MR. In Section 30.7.7 "DDRSDRC Low-power Register" UDP_MR typo corrected.	7210
	"TWTR: Internal Write to Read Delay", bitfield table updated.	rfo
	DMAC:	
	"FC: Flow Control", removed last four lines from bitfield table.	7353
	Section 31.5.1 "Basic Definitions", added Programmable Arbitration Policy.	7366
	External Memories:	
	Section 26.8.7 "8-bit NAND Flash with NFD0_ON_D16 = 1"	rfo
	Section 26.8.7.2 "Software Configuration", added the line: "Configure the PIOD controller to assign..."	

Doc. Rev 11063C	Comments (Continued)	Change Request Ref.(1)
	HSMCI:	
	Table 35-8, “Register Mapping” and Section 35.14.20 “HSMCI FIFOx Memory Aperture”, HSMCI_FIFOx offset updated.	7253
	MATRIX:	
	Section 25-5 “Chip Configuration User Interface”, CCFG_EBICSA offset values revised.	rfo
	PIO:	
	Figure 23-3, I/O Line Control Logic, Table 23-2, “Register Mapping”, “PIO Input Filter Slow Clock Disable Register”, “PIO Input Filter Slow Clock Enable Register”, “PIO Input Filter Slow Clock Status Register”,updated IFSxx register acronyms.	6787
	Table 23-2, “Register Mapping”, “PIO I/O Drive Register 1” and “PIO I/O Drive Register 2” added to datasheet.	6876, 7255
	PMC:	
	Section 21.12 “Power Management Controller (PMC) User Interface”, PLLB is usable as input clock.	7304
	Section 21-3 “Register Mapping”, offset 0x0038 updated with USB Clock Register (PMC_USB).	rfo
	Section 21.5 “Processor Clock Controller”, revised.	7369
	Section 21.12.10 “PMC Clock Generator PLLB Register”, removed USBDIV bitfields.	rfo
	Section 21.12.12 “USB Clock Register”, added to datasheet.	
	PMEEC:	
	ERRIE, ERRID, ERRIM bitfields are 1 bit wide. See:Section 28.6.8 “PMECC Interrupt Enable Register”, Section 28.6.9 “PMECC Interrupt Disable Register” and Section 28.6.10 “PMECC Interrupt Mask Register”.	7202
	PMERRLOC:	
	Table 29-3, “Register Mapping” PMECC SIGMA 24 is located at 0x088.	7203
	Section 29.5.10 “Error Location SIGMAx Register”, updated.	
	SCKC:	
	Section 20. “Slow Clock Controller (SCKC)”, added to datasheet.	rfo
	SMC:	
	Table 30-1, “I/O Line Description”, replaced NCS[7:0] by NCS[5:0]	rfo
	SPI:	
	Section 36.8.9 “SPI Chip Select Register”, “SCBR: Serial Clock Baud Rate”, data transfer note added.	7247
	Section 36.8.3 “SPI Receive Data Register”added requirements to bitfield “PCS: Peripheral Chip Select”.	
	Section 36.8.9 “SPI Chip Select Register”, “BITS: Bits Per Transfer”, bitfield table; Description column revised.	7319
	Section 36.7.3.5 “Peripheral Selection”, added paragraph at end of the section.	7267
	TC:	
	Section 37.7.5 “TC Channel Mode Register: Waveform Mode”, updated WAVSEL bitfield table.	7190
	“TC Counter Value Register”, “TC Register A”, “TC Register B”, “TC Register C” all bitfields are filled.	
	Figure 37-5, Capture Mode and Figure 37-6, Waveform Mode, revise the counter component.	7318
	TRNG:	
	Section 47.2 “Embedded Characteristics”, removed 133 MHz Clock Frequency.	rfo
	Section 47.3.1 “TRNG Control Register”, added KEY bitfield.	5914

© 2017 Microchip Technology Inc.

Doc. Rev 11063C	Comments (Continued)	Change Request Ref.(1)
	TWI: Section 39.8.7 “Using the DMA Controller”, added to the datasheet.	7306
	UDP: Section 33.4 “Product Dependencies”, second paragraph removed. Section 33.5 “Typical Connection”, revised schematic and VBUS Monitoring. Section 33.6.3.2 “Entering Attached State”, revised, replaced paragraphs before **Warning**. Section 33.7.12 “UDP Transceiver Control Register”, bit field 9 is dedicated to PUON.	7322
	UHP: Section 33-1 “Block Diagram”, removed **Warning**. Section 33.6 “Typical Connection”, revised schematic and text.	7322
	USART: Melange of references to PDC/DMA removed in favor of DMA implementation.	7284
	UART: Section 41-1 “UART Functional Block Diagram”, revised.	7285
	Electrical Characteristics: Table 47-7, “Main Oscillator Characteristics”, revised schematic in note below table. Table 47-11, “32 kHz Oscillator Characteristics”, revised schematic in note below table.	7304
	Table 47-5, “Processor Clock Waveform Parameters”, updated. Table 47-6, “System Clock Waveform Parameters”, replaces “Master Clock Waveform Parameters”. Section 47.13 “SMC Timings”, added to datasheet. Section 47.14 “DDRSDRC Timings”, added to datasheet. Section 47.15 “Peripheral Timings”, added to datasheet.	7334
	Table 47-2, “DC Characteristics” Table 47-3, “Power Consumption for Different Modes” Table 47-4, “Power Consumption by Peripheral in Active Mode” Table 47-18, “I/O Characteristics” former TBDs assigned values	rfo

Doc. Rev 11063B	Comments	Change Request Ref.
	Table 3-2, “BGA247 Pin Description” updated.	7271

Note 1: “rfo” indicates changes requested during document review and approval loop.

Doc. Rev 11063A	Comments	Change Request Ref.
	First issue	

The Microchip Web Site

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com. Under "Design Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

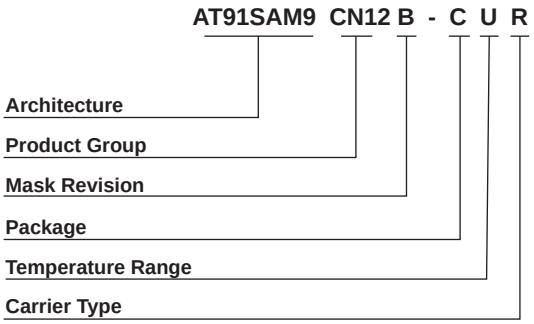
Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://microchip.com/support

© 2017 Microchip Technology Inc.

Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



Architecture:	ATSAM9 = Arm926 processor		
Product Group:	N12 CN11 CN12	= = =	General-purpose microprocessor N12 + crypto (for evaluation only) CN11 + crypto + Secure Boot
Mask Revision:	B		
Package:	C CF	= =	BGA217 TFBGA247
Temperature Range:	U	=	-40°C to +85°C (industrial)
Carrier Type:	Blank R	= =	Standard Packaging (tray) Tape and Reel(1)

Example:

a) AT91SAM9CN12B-CUR = Arm926 general-purpose microprocessor, crypto, Secure Boot, 217-ball, industrial temperature, BGA package.

Note 1: Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.

2: Small form-factor packaging options may be available. Please check www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, KeeLoq logo, Kleer, LANCheck, LINK MD, MediaLB, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT logo, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademarks of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2017, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-2170-2

© 2017 Microchip Technology Inc.

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

France - Saint Cloud
Tel: 33-1-30-60-70-00

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

© 2017 Microchip Technology Inc.

Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View AT91SAM9CN12-CFU on WIN SOURCE
- Microchip Technology Information

Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management