THE DATASHEET OF

MC68331CEH20

Order this document
by MC68331TS/D Rev. 2

MC68331

Technical Summary

32-Bit Modular Microcontroller

1 Introduction

The MC68331, a highly-integrated 32-bit microcontroller, combines high-performance data manipulation capabilities with powerful peripheral subsystems. The MCU is built up from standard modules that interface through a common intermodule bus (IMB). Standardization facilitates rapid development of devices tailored for specific applications.

The MCU incorporates a 32-bit CPU (CPU32), a system integration module (SIM), a general-purpose timer (GPT), and a queued serial module (QSM).

The MCU can either synthesize an internal clock signal from an external reference or use an external clock input directly. Operation with a 32.768-kHz reference frequency is standard. The maximum system clock speed is 20.97 MHz. Because MCU operation is fully static, register and memory contents are not affected by a loss of clock.

High-density complementary metal-oxide semiconductor (HCMOS) architecture makes the basic power consumption of the MCU low. Power consumption can be minimized by stopping the system clock. The CPU32 instruction set includes a low-power stop (LPSTOP) command that efficiently implements this capability.

Table 1 Ordering Information

Package Type	Temperature	Frequency (MHz)	Package Order Quantity	Order Number
132-Pin PQFP	–40 to +85 °C	16 MHz	2 pc tray	SPAKMC331CFC16
			36 pc tray	MC68331CFC16
		20 MHz	2 pc tray	SPAKMC331CFC20
			36 pc tray	MC68331CFC20
	–40 to +105 °C	16 MHz	2 pc tray	SPAKMC331VFC16
			36 pc tray	MC68331VFC16
		20 MHz	2 pc tray	SPAKMC331VFC20
			36 pc tray	MC68331VFC20
	–40 to +125 °C	16 MHz	2 pc tray	SPAKMC331MFC16
			36 pc tray	MC68331MFC16
		20 MHz	2 pc tray	SPAKMC331MFC20
			36 pc tray	MC68331MFC20
144-Pin QFP	–40 to +85 °C	16 MHz	2 pc tray	SPAKMC331CFV16
			44 pc tray	MC68331CFV16
		20 MHz	2 pc tray	SPAKMC331CFV20
			44 pc tray	MC68331CFV20
	–40 to +105 °C	16 MHz	2 pc tray	SPAKMC331VFV16
			44 pc tray	MC68331VFV16
		20 MHz	2 pc tray	SPAKMC331VFV20
			44 pc tray	MC68331VFV20
	–40 to +125 °C	16 MHz	2 pc tray	SPAKMC331MFV16
			44 pc tray	MC68331MFV16
		20 MHz	2 pc tray	SPAKMC331MFV20
			44 pc tray	MC68331MFV20

TABLE OF CONTENTS

Section			Page
1		**Introduction**	**1**
	1.1	Features	4
	1.2	Block Diagram	5
	1.3	Pin Assignments	6
	1.4	Address Map	8
	1.5	Intermodule Bus	8
2		**Signal Descriptions**	**9**
	2.1	Pin Characteristics	9
	2.2	MCU Power Connections	10
	2.3	MCU Driver Types	10
	2.4	Signal Characteristics	10
	2.5	Signal Function	11
3		**System Integration Module**	**14**
	3.1	Overview	14
	3.2	System Configuration and Protection	16
	3.3	System Clock	21
	3.4	External Bus Interface	24
	3.5	Chip Selects	28
	3.6	General-Purpose Input/Output	35
	3.7	Resets	37
	3.8	Interrupts	39
	3.9	Factory Test Block	41
4		**Central Processor Unit**	**43**
	4.1	Overview	43
	4.2	Programming Model	43
	4.3	Status Register	45
	4.4	Data Types	45
	4.5	Addressing Modes	45
	4.6	Instruction Set Summary	46
	4.7	Background Debugging Mode	50
5		**Queued Serial Module**	**51**
	5.1	Overview	51
	5.2	Pin Function	52
	5.3	QSM Registers	53
	5.4	QSPI Submodule	56
	5.5	SCI Submodule	64
6		**General-Purpose Timer Module**	**70**
	6.1	Overview	70
	6.2	Capture/Compare Unit	71
	6.3	Pulse-Width Modulator	74
	6.4	GPT Registers	75
7		**Summary of Changes**	**82**

1.1 Features

- Modular Architecture
- Central Processing Unit (CPU32)
  - — Upward Object Code Compatible
  - — New Instructions for Controller Applications
  - — 32-Bit Architecture
  - — Virtual Memory Implementation
  - — Loop Mode of Instruction Execution
  - — Table Lookup and Interpolate Instruction
  - — Improved Exception Handling for Controller Applications
  - — Trace on Change of Flow
  - — Hardware Breakpoint Signal, Background Mode
  - — Fully Static Operation
- System Integration Module (SIM)
  - — External Bus Support
  - — Programmable Chip-Select Outputs
  - — System Protection Logic
  - — Watchdog Timer, Clock Monitor, and Bus Monitor
  - — System Protection Logic
  - — System Clock Based on 32.768-kHz Crystal for Low Power Operation
  - — Test/Debug Submodule for Factory/User Test and Development
- Queued Serial Module (QSM)
  - — Enhanced Serial Communication Interface (SCI), Universal Asynchronous Receiver Transmitter (UART): Modulus Baud Rate, Parity
  - — Queued Serial Peripheral Interface (QSPI): 80-Byte RAM, Up to 16 Automatic Transfers
  - — Dual Function I/O Ports
  - — Continuous Cycling, 8 to 16 Bits per Transfer
- General-Purpose Timer (GPT)
  - — Two 16-Bit Free-Running Counters With One Nine-Stage Prescaler
  - — Three Input Capture Channels
  - — Four Output Compare Channels
  - — One Input Capture/Output Compare Channel
  - — One Pulse Accumulator/Event Counter Input
  - — Two Pulse-Width Modulation Outputs
  - — Optional External Clock Input

1.2 Block Diagram

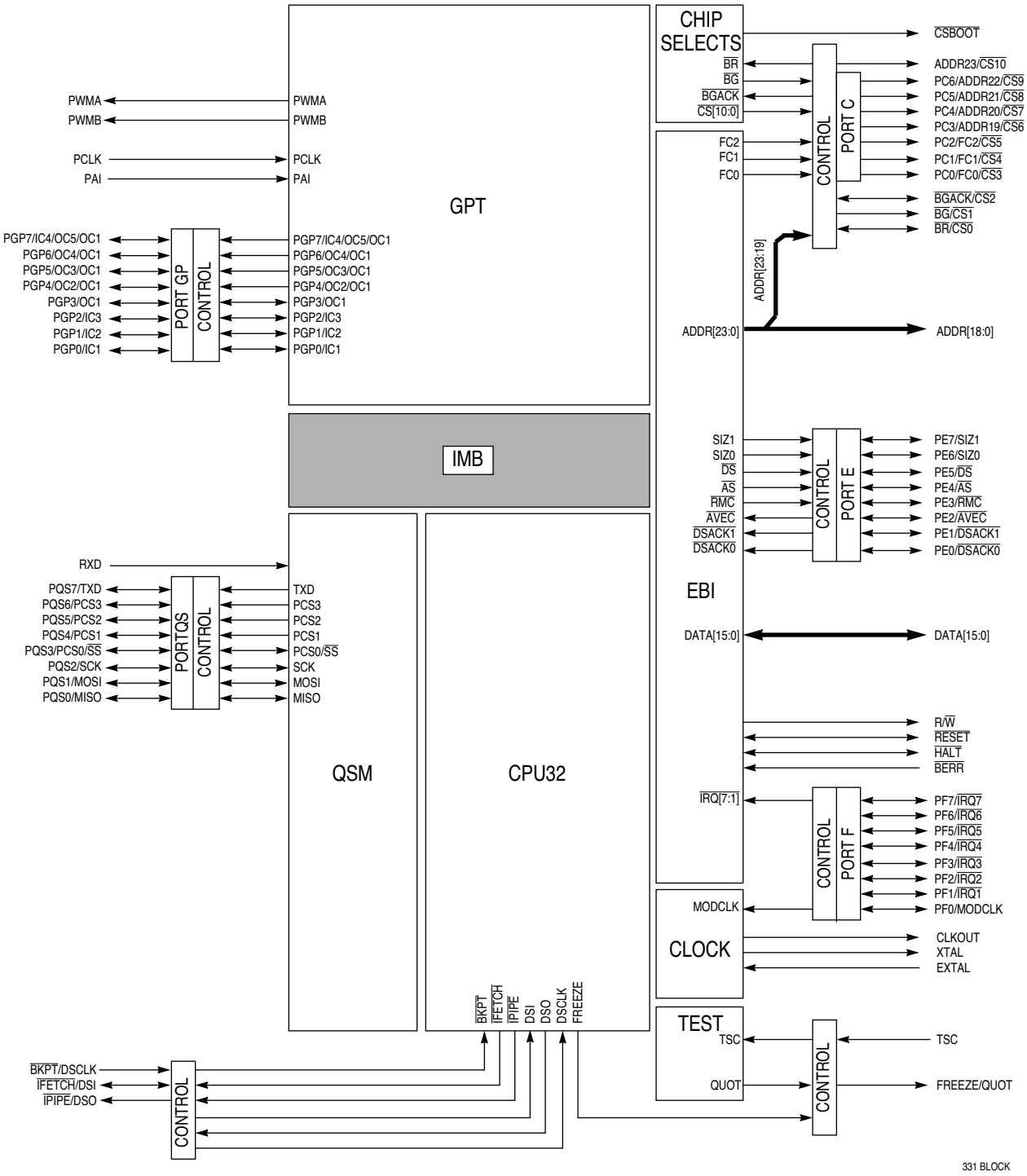


Figure 1 MCU Block Diagram

1.3 Pin Assignments

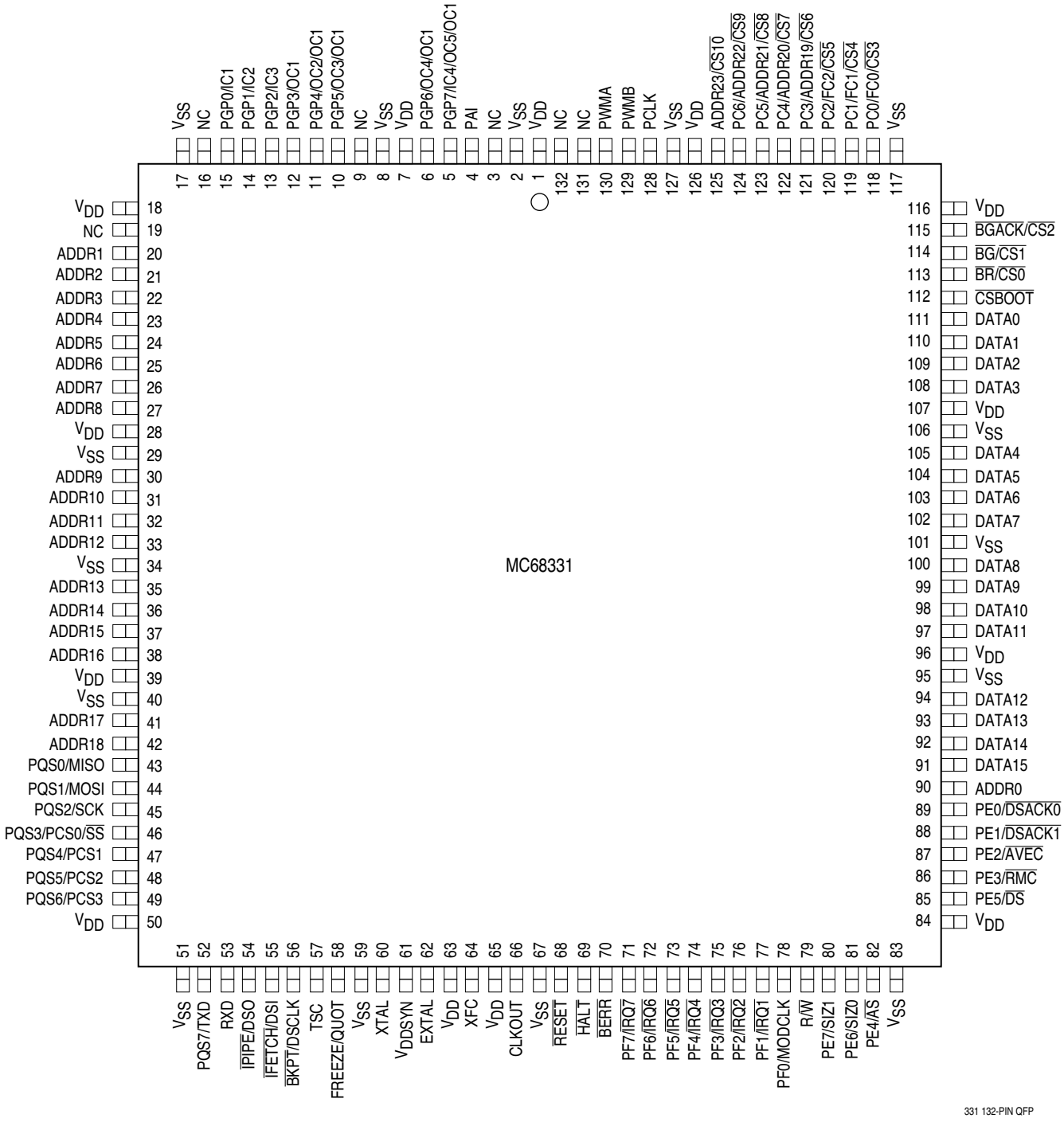


Figure 2 MC68331 132-Pin QFP Pin Assignments

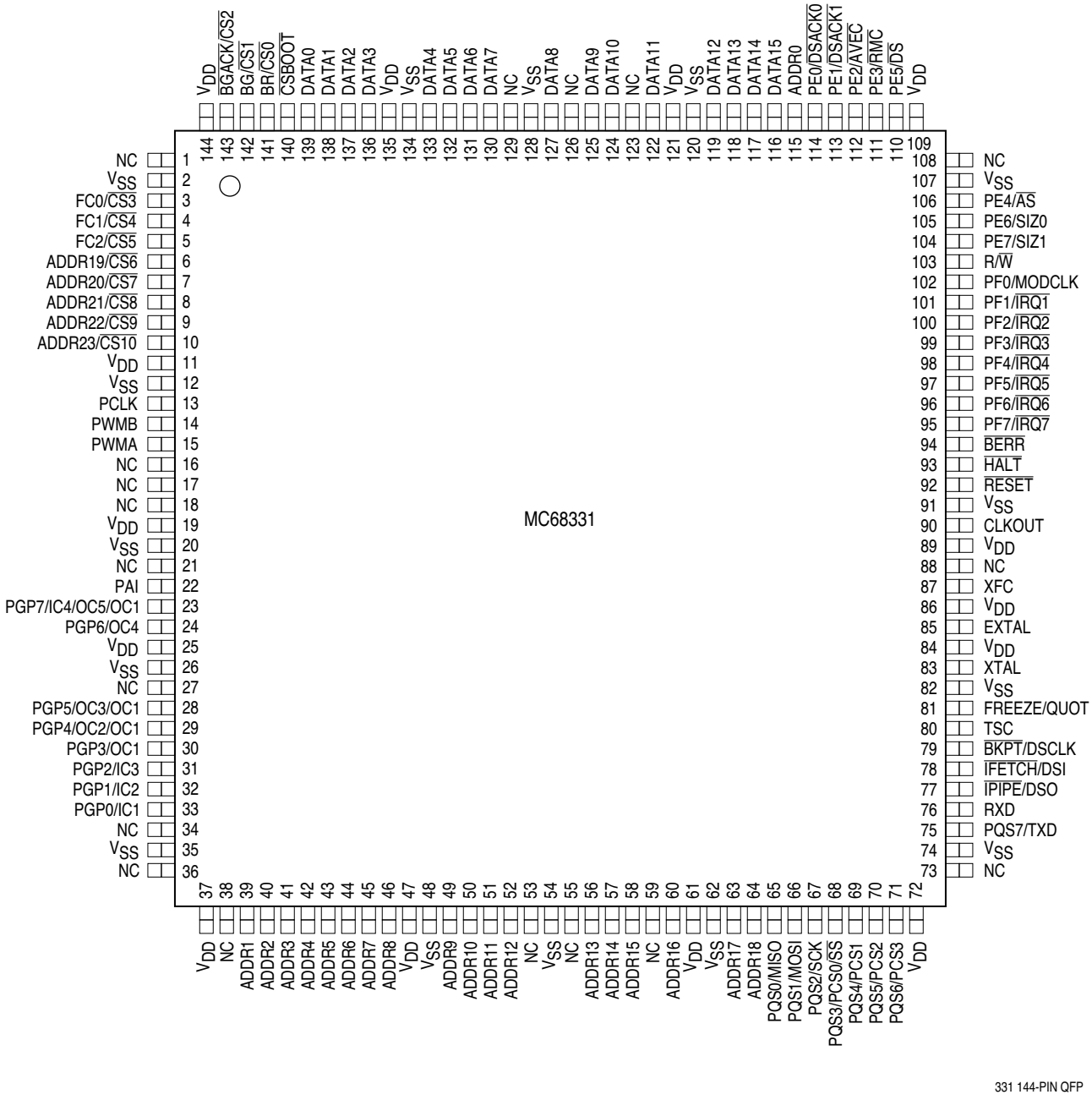


Figure 3 MC68331 144-Pin QFP Pin Assignments

1.4 Address Map

The following figure is a map of the MCU internal addresses. Unimplemented blocks are mapped externally.

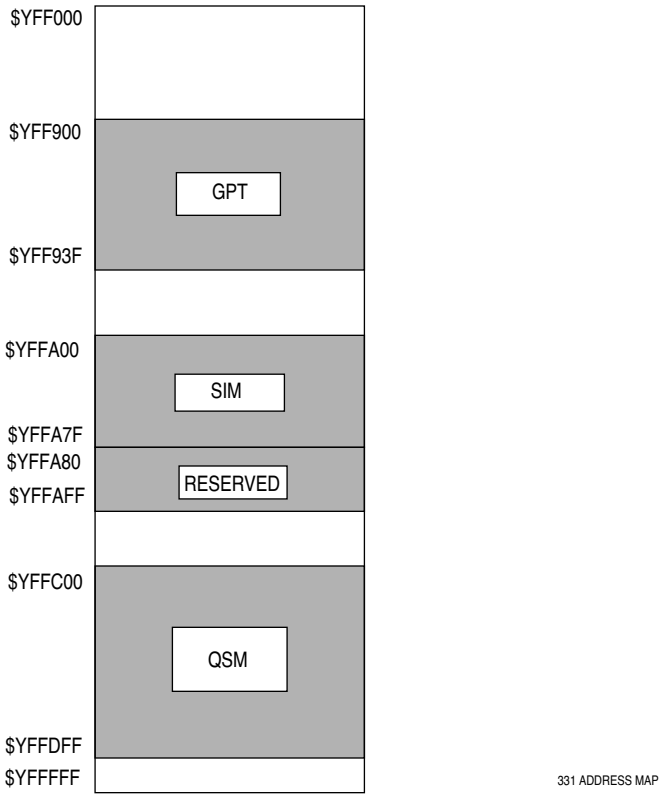


Figure 4 MCU Address Map

1.5 Intermodule Bus

The intermodule bus (IMB) is a standardized bus developed to facilitate both design and operation of modular microcontrollers. It contains circuitry to support exception processing, address space partitioning, multiple interrupt levels, and vectored interrupts. The standardized modules in the MCU communicate with one another and with external components through the IMB. The IMB in the MCU uses 24 address and 16 data lines.

2 Signal Descriptions

2.1 Pin Characteristics

The following table shows MCU pins and their characteristics. All inputs detect CMOS logic levels. All inputs can be put in a high-impedance state, but the method of doing this differs depending upon pin function. Refer to **Table 4**, for a description of output drivers. An entry in the discrete I/O column of **Table 2** indicates that a pin has an alternate I/O function. The port designation is given when it applies. Refer to the MCU Block Diagram for information about port organization.

Table 2 MCU Pin Characteristics

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
ADDR23/$\overline{CS10}$/ECLK	A	Y	N	O	—
ADDR[22:19]/$\overline{CS[9:6]}$	A	Y	N	O	PC[6:3]
ADDR[18:0]	A	Y	N	—	—
$\overline{AS}$	B	Y	N	I/O	PE5
$\overline{AVEC}$	B	Y	N	I/O	PE2
$\overline{BERR}$	B	Y	N	—	—
$\overline{BG}$/$\overline{CS1}$	B	—	—	—	—
$\overline{BGACK}$/$\overline{CS2}$	B	Y	N	—	—
$\overline{BKPT}$/DSCLK	—	Y	Y	—	—
$\overline{BR}$/$\overline{CS0}$	B	Y	N	—	—
CLKOUT	A	—	—	—	—
$\overline{CSBOOT}$	B	—	—	—	—
DATA[15:0][1]	Aw	Y	N	—	—
$\overline{DS}$	B	Y	N	I/O	PE4
$\overline{DSACK1}$	B	Y	N	I/O	PE1
$\overline{DSACK0}$	B	Y	N	I/O	PE0
DSI/$\overline{IFETCH}$	A	Y	Y	—	—
DSO/$\overline{IPIPE}$	A	—	—	—	—
EXTAL[2]	—	—	Special	—	—
FC[2:0]/$\overline{CS[5:3]}$	A	Y	N	O	PC[2:0]
FREEZE/QUOT	A	—	—	—	—
IC4/OC5	A	Y	Y	I/O	GP4
IC[3:1]	A	Y	Y	I/O	GP[7:5]
$\overline{HALT}$	Bo	Y	N	—	—
$\overline{IRQ[7:1]}$	B	Y	Y	I/O	PF[7:1]
MISO	Bo	Y	Y	I/O	PQS0
MODCLK[1]	B	Y	N	I/O	PF0
MOSI	Bo	Y	Y	I/O	PQS1
OC[4:1]	A	Y	Y	I/O	GP[3:0]
PAI[3]	—	Y	Y	I	—
PCLK[3]	—	Y	Y	I	—
$\overline{PCS0}$/$\overline{SS}$	Bo	Y	Y	I/O	PQS3
$\overline{PCS[3:1]}$	Bo	Y	Y	I/O	PQS[6:4]
PWMA, PWMB	A	—	—	O	—
R/$\overline{W}$	A	Y	N	—	—
$\overline{RESET}$	Bo	Y	Y	—	—
$\overline{RMC}$	B	Y	N	I/O	PE3

Table 2 MCU Pin Characteristics (Continued)

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
RXD	—	N	N	—	—
SCK	Bo	Y	Y	I/O	PQS2
SIZ[1:0]	B	Y	N	I/O	PE[7:6]
TSC	—	Y	Y	—	—
TXD	Bo	Y	Y	I/O	PQS7
XFC[2]	—	—	—	Special	—
XTAL[2]	—	—	—	Special	—

NOTES:

1. DATA[15:0] are synchronized during reset only. MODCLK is synchronized only when used as an input port pin.
2. EXTAL, XFC, and XTAL are clock reference connections.
3. PAI and PCLK can be used for discrete input, but are not part of an I/O port.

2.2 MCU Power Connections

Table 3 MCU Power Connections

V_{DDSYN}	Clock Synthesizer Power
V_{SSE}/V_{DDE}	External Periphery Power (Source and Drain)
V_{SSI}/V_{DDI}	Internal Module Power (Source and Drain)

2.3 MCU Driver Types

Table 4 MCU Driver Types

Type	I/O	Description
A	O	Output-only signals that are always driven; no external pull-up required
Aw	O	Type A output with weak P-channel pull-up during reset
B	O	Three-state output that includes circuitry to pull up output before high impedance is established, to ensure rapid rise time. An external holding resistor is required to maintain logic level while the pin is in the high-impedance state.
Bo	O	Type B output that can be operated in an open-drain mode

2.4 Signal Characteristics

Table 5 MCU Signal Characteristics

Signal Name	MCU Module	Signal Type	Active State
ADDR[23:0]	SIM	Bus	—
$\overline{AS}$	SIM	Output	0
$\overline{AVEC}$	SIM	Input	0
$\overline{BERR}$	SIM	Input	0
$\overline{BG}$	SIM	Output	0
$\overline{BGACK}$	SIM	Input	0
$\overline{BKPT}$	CPU32	Input	0
$\overline{BR}$	SIM	Input	0
CLKOUT	SIM	Output	—
$\overline{CS}$[10:0]	SIM	Output	0
$\overline{CSBOOT}$	SIM	Output	0
DATA[15:0]	SIM	Bus	—
$\overline{DS}$	SIM	Output	0

Table 5 MCU Signal Characteristics (Continued)

Signal Name	MCU Module	Signal Type	Active State
$\overline{\text{DSACK}}$[1:0]	SIM	Input	0
DSCLK	CPU32	Input	Serial Clock
DSI	CPU32	Input	(Serial Data)
DSO	CPU32	Output	(Serial Data)
EXTAL	SIM	Input	—
FC[2:0]	SIM	Output	—
FREEZE	SIM	Output	1
$\overline{\text{HALT}}$	SIM	Input/Output	0
IC[4:1]	GPT	Input	—
$\overline{\text{IFETCH}}$	CPU32	Output	—
$\overline{\text{IPIPE}}$	CPU32	Output	—
$\overline{\text{IRQ}}$[7:1]	SIM	Input	0
MISO	QSM	Input/Output	—
MODCLK	SIM	Input	—
MOSI	QSM	Input/Output	—
OC[5:1]	GPT	Output	—
PAI	GPT	Input	—
PC[6:0]	SIM	Output	(Port)
PCS[3:0]	QSM	Input/Output	—
PE[7:0]	SIM	Input/Output	(Port)
PF[7:0]	SIM	Input/Output	(Port)
PQS[7:0]	QSM	Input/Output	(Port)
PCLK	GPT	Input	—
PWMA, PWMB	GPT	Output	—
QUOT	SIM	Output	—
$\overline{\text{RESET}}$	SIM	Input/Output	0
$\overline{\text{RMC}}$	SIM	Output	0
R/$\overline{\text{W}}$	SIM	Output	1/0
RXD	QSM	Input	—
SCK	QSM	Input/Output	—
SIZ[1:0]	SIM	Output	—
$\overline{\text{SS}}$	QSM	Input	0
TSC	SIM	Input	—
TXD	QSM	Output	—
XFC	SIM	Input	—
XTAL	SIM	Output	—

2.5 Signal Function

Table 6 MCU Signal Function

Signal Name	Mnemonic	Function
Address Bus	ADDR[23:0]	24-bit address bus
Address Strobe	$\overline{\text{AS}}$	Indicates that a valid address is on the address bus
Autovector	$\overline{\text{AVEC}}$	Requests an automatic vector during interrupt acknowledge
Bus Error	$\overline{\text{BERR}}$	Indicates that a bus error has occurred
Bus Grant	$\overline{\text{BG}}$	Indicates that the MCU has relinquished the bus
Bus Grant Acknowledge	$\overline{\text{BGACK}}$	Indicates that an external device has assumed bus mastership
Breakpoint	$\overline{\text{BKPT}}$	Signals a hardware breakpoint to the CPU
Bus Request	$\overline{\text{BR}}$	Indicates that an external device requires bus mastership
System Clockout	CLKOUT	System clock output
Chip Selects	$\overline{\text{CS}}$[10:0]	Select external devices at programmed addresses

Table 6 MCU Signal Function (Continued)

Signal Name	Mnemonic	Function
Boot Chip Select	$\overline{\text{CSBOOT}}$	Chip select for external boot startup ROM
Data Bus	DATA[15:0]	16-bit data bus
Data Strobe	$\overline{\text{DS}}$	During a read cycle, indicates when it is possible for an external device to place data on the data bus. During a write cycle, indicates that valid data is on the data bus.
Data and Size Acknowledge	$\overline{\text{DSACK}}$[1:0]	Provide asynchronous data transfers and dynamic bus sizing
Development Serial In, Out, Clock	DSI, DSO, DSCLK	Serial I/O and clock for background debugging mode
Crystal Oscillator	EXTAL, XTAL	Connections for clock synthesizer circuit reference; a crystal or an external oscillator can be used
Function Codes	FC[2:0]	Identify processor state and current address space
Freeze	FREEZE	Indicates that the CPU has entered background mode
Halt	$\overline{\text{HALT}}$	Suspend external bus activity
Input Capture	IC[3:1]	When a specified transition is detected on an input capture pin, the value in an internal GPT counter is latched
Input Capture 4/Output Compare 5	IC4/OC5	Can be configured for either an input capture or output compare
Instruction Pipeline	$\overline{\text{IPIPE}}$, $\overline{\text{IFETCH}}$	Indicate instruction pipeline activity
Interrupt Request Level	$\overline{\text{IRQ}}$[7:1]	Provides an interrupt priority level to the CPU
Master In Slave Out	MISO	Serial input to QSPI in master mode; serial output from QSPI in slave mode
Clock Mode Select	MODCLK	Selects the source and type of system clock
Master Out Slave In	MOSI	Serial output from QSPI in master mode; serial input to QSPI in slave mode
Output Compare	OC[5:1]	Change state when the value of an internal GPT counter matches a value stored in a GPT control register
Pulse Accumulator Input	PAI	Signal input to the pulse accumulator
Port C	PC[6:0]	SIM digital output port signals
Auxiliary Timer Clock Input	PCLK	External clock dedicated to the GPT
Peripheral Chip Select	PCS[3:0]	QSPI peripheral chip selects
Port E	PE[7:0]	SIM digital I/O port signals
Port F	PF[7:0]	SIM digital I/O port signals
Port QS	PQS[7:0]	QSM digital I/O port signals
Pulse-Width Modulation	PWMA, PWMB	Output for PWM
Quotient Out	QUOT	Provides the quotient bit of the polynomial divider
Reset	$\overline{\text{RESET}}$	System reset
Read-Modify-Write Cycle	$\overline{\text{RMC}}$	Indicates an indivisible read-modify-write instruction
Read/Write	R/$\overline{\text{W}}$	Indicates the direction of data transfer on the bus
SCI Receive Data	RXD	Serial input to the SCI
QSPI Serial Clock	SCK	Clock output from QSPI in master mode; clock input to QSPI in slave mode
Size	SIZ[1:0]	Indicates the number of bytes to be transferred during a bus cycle
Slave Select	$\overline{\text{SS}}$	Causes serial transmission when QSPI is in slave mode; causes mode fault in master mode
Three-State Control	TSC	Places all output drivers in a high-impedance state
SCI Transmit Data	TXD	Serial output from the SCI
External Filter Capacitor	XFC	Connection for external phase-locked loop filter capacitor

3 System Integration Module

The system integration module (SIM) consists of five functional blocks that control system start-up, initialization, configuration, and external bus.

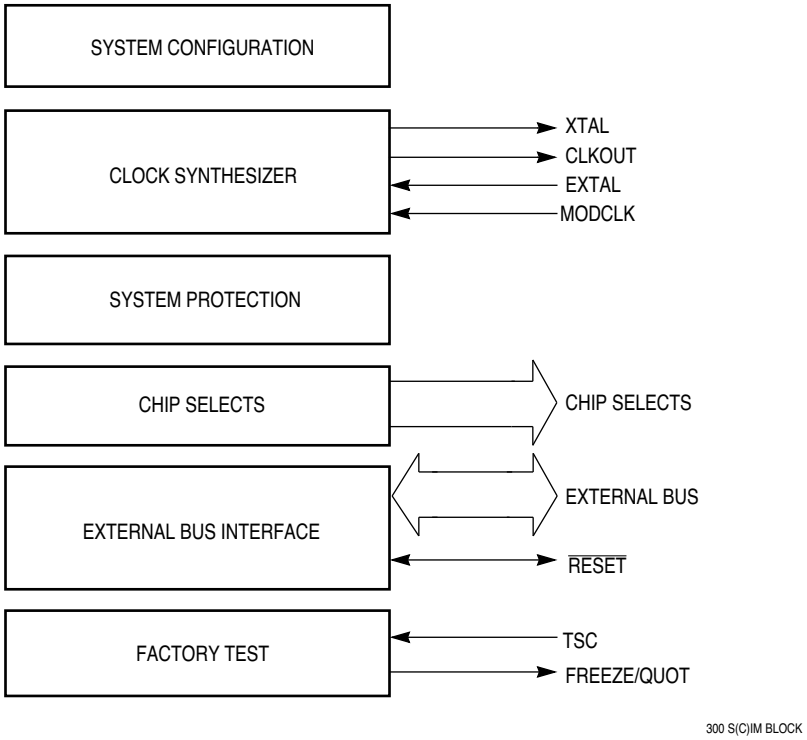


Figure 5 SIM Block Diagram

3.1 Overview

The system configuration and protection block controls MCU configuration and operating mode. The block also provides bus and software watchdog monitors.

The system clock generates clock signals used by the SIM, other IMB modules, and external devices. In addition, a periodic interrupt generator supports execution of time-critical control routines.

The external bus interface handles the transfer of information between IMB modules and external address space.

The chip-select block provides eleven general-purpose chip-select signals and a boot ROM chip-select signal. Both general-purpose and boot ROM chip-select signals have associated base address registers and option registers.

The system test block incorporates hardware necessary for testing the MCU. It is used to perform factory tests, and its use in normal applications is not supported.

The SIM control register address map occupies 128 bytes. Unused registers within the 128-byte address space return zeros when read. The “Access” column in the SIM address map below indicates which registers are accessible only at the supervisor privilege level and which can be assigned to either the supervisor or user privilege level, according to the value of the SUPV bit in the SIMCR.

Table 7 SIM Address Map

Access	Address	15 8	7 0
S	$YFFA00	SIM CONFIGURATION (SIMCR)	
S	$YFFA02	FACTORY TEST (SIMTR)	
S	$YFFA04	CLOCK SYNTHESIZER CONTROL (SYNCR)	
S	$YFFA06	NOT USED	RESET STATUS REGISTER (RSR)
S	$YFFA08	MODULE TEST E (SIMTRE)	
S	$YFFA0A	NOT USED	NOT USED
S	$YFFA0C	NOT USED	NOT USED
S	$YFFA0E	NOT USED	NOT USED
S/U	$YFFA10	NOT USED	PORT E DATA (PORTE0)
S/U	$YFFA12	NOT USED	PORT E DATA (PORTE1)
S/U	$YFFA14	NOT USED	PORT E DATA DIRECTION (DDRE)
S	$YFFA16	NOT USED	PORT E PIN ASSIGNMENT (PEPAR)
S/U	$YFFA18	NOT USED	PORT F DATA (PORTF0)
S/U	$YFFA1A	NOT USED	PORT F DATA (PORTF1)
S/U	$YFFA1C	NOT USED	PORT F DATA DIRECTION (DDRF)
S	$YFFA1E	NOT USED	PORT F PIN ASSIGNMENT (PFPAR)
S	$YFFA20	NOT USED	SYSTEM PROTECTION CONTROL (SYPCR)
S	$YFFA22	PERIODIC INTERRUPT CONTROL (PICR)	
S	$YFFA24	PERIODIC INTERRUPT TIMING (PITR)	
S	$YFFA26	NOT USED	SOFTWARE SERVICE (SWSR)
S	$YFFA28	NOT USED	NOT USED
S	$YFFA2A	NOT USED	NOT USED
S	$YFFA2C	NOT USED	NOT USED
S	$YFFA2E	NOT USED	NOT USED
S	$YFFA30	TEST MODULE MASTER SHIFT A (TSTMSRA)	
S	$YFFA32	TEST MODULE MASTER SHIFT B (TSTMSRB)	
S	$YFFA34	TEST MODULE SHIFT COUNT (TSTSC)	
S	$YFFA36	TEST MODULE REPETITION COUNTER (TSTRC)	
S	$YFFA38	TEST MODULE CONTROL (CREG)	
S/U	$YFFA3A	TEST MODULE DISTRIBUTED REGISTER (DREG)	
	$YFFA3C	NOT USED	NOT USED
	$YFFA3E	NOT USED	NOT USED
S/U	$YFFA40	NOT USED	PORT C DATA (PORTC)
	$YFFA42	NOT USED	NOT USED
S	$YFFA44	CHIP-SELECT PIN ASSIGNMENT (CSPAR0)	
S	$YFFA46	CHIP-SELECT PIN ASSIGNMENT (CSPAR1)	
S	$YFFA48	CHIP-SELECT BASE BOOT (CSBARBT)	
S	$YFFA4A	CHIP-SELECT OPTION BOOT (CSORBT)	
S	$YFFA4C	CHIP-SELECT BASE 0 (CSBAR0)	
S	$YFFA4E	CHIP-SELECT OPTION 0 (CSOR0)	
S	$YFFA50	CHIP-SELECT BASE 1 (CSBAR1)	
S	$YFFA52	CHIP-SELECT OPTION 1 (CSOR1)	
S	$YFFA54	CHIP-SELECT BASE 2 (CSBAR2)	
S	$YFFA56	CHIP-SELECT OPTION 2 (CSOR2)	
S	$YFFA58	CHIP-SELECT BASE 3 (CSBAR3)	
S	$YFFA5A	CHIP-SELECT OPTION 3 (CSOR3)	
S	$YFFA5C	CHIP-SELECT BASE 4 (CSBAR4)	

Table 7 SIM Address Map (Continued)

Access	Address	15 8	7 0
S	$YFFA5E	CHIP-SELECT OPTION 4 (CSOR4)	
S	$YFFA60	CHIP-SELECT BASE 5 (CSBAR5)	
S	$YFFA62	CHIP-SELECT OPTION 5 (CSOR5)	
S	$YFFA64	CHIP-SELECT BASE 6 (CSBAR6)	
S	$YFFA66	CHIP-SELECT OPTION 6 (CSOR6)	
S	$YFFA68	CHIP-SELECT BASE 7 (CSBAR7)	
S	$YFFA6A	CHIP-SELECT OPTION 7 (CSOR7)	
S	$YFFA6C	CHIP-SELECT BASE 8 (CSBAR8)	
S	$YFFA6E	CHIP-SELECT OPTION 8 (CSOR8)	
S	$YFFA70	CHIP-SELECT BASE 9 (CSBAR9)	
S	$YFFA72	CHIP-SELECT OPTION 9 (CSOR9)	
S	$YFFA74	CHIP-SELECT BASE 10 (CSBAR10)	
S	$YFFA76	CHIP-SELECT OPTION 10 (CSOR10)	
	$YFFA78	NOT USED	NOT USED
	$YFFA7A	NOT USED	NOT USED
	$YFFA7C	NOT USED	NOT USED
	$YFFA7E	NOT USED	NOT USED

Y = M111, where M is the logic state of the module mapping (MM) bit in the SIMCR.

3.2 System Configuration and Protection

This functional block provides configuration control for the entire MCU. It also performs interrupt arbitration, bus monitoring, and system test functions. MCU system protection includes a bus monitor, a HALT monitor, a spurious interrupt monitor, and a software watchdog timer. These functions have been made integral to the microcontroller to reduce the number of external components in a complete control system.

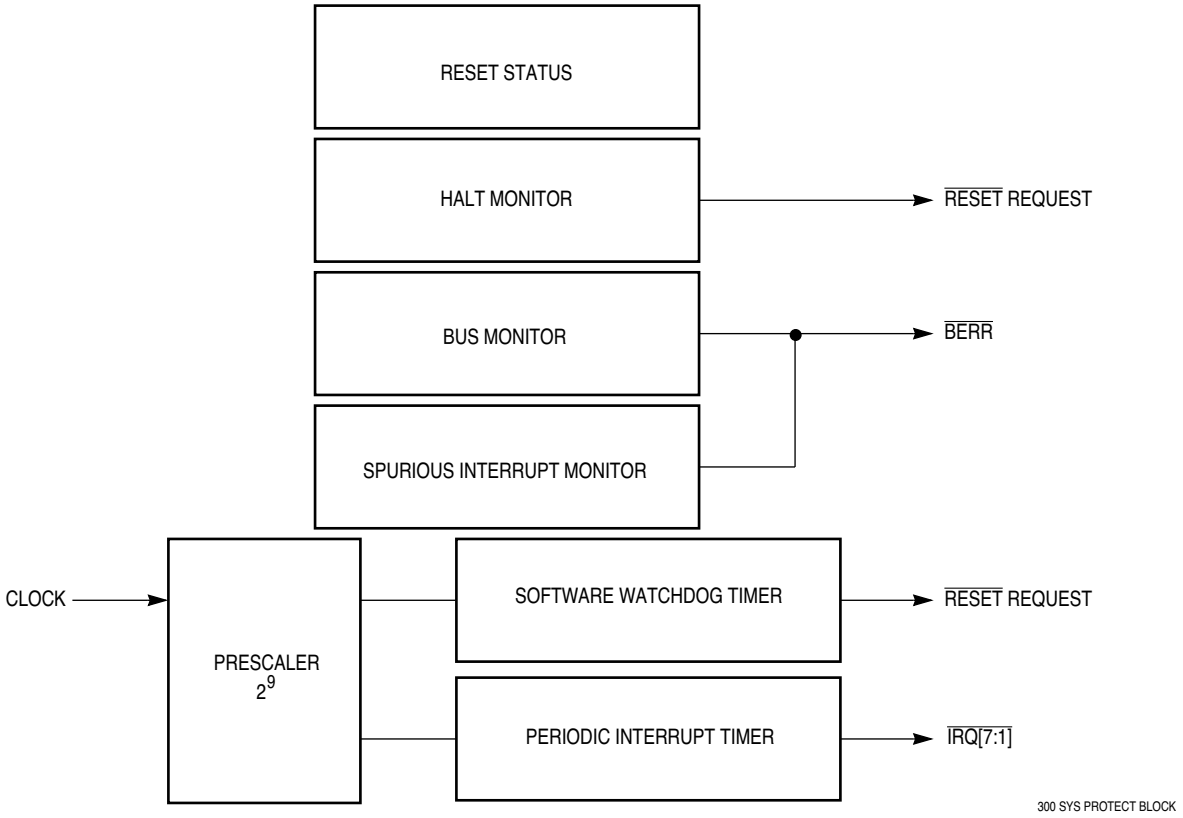


Figure 6 System Configuration and Protection Block

3.2.1 System Configuration

The SIM controls MCU configuration during normal operation and during internal testing.

SIMCR —SIM Configuration Register **$YFFA00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXOFF	FRZSW	FRZBM	0	SLVEN	0	SHEN		SUPV	MM	0	0	IARB			
RESET:															
0	0	0	0	DATA11	0	0	0	1	1	0	0	1	1	1	1

The SIM configuration register controls system configuration. It can be read or written at any time, except for the module mapping (MM) bit, which can be written only once.

EXOFF —External Clock Off

0 = The CLKOUT pin is driven from an internal clock source.
1 = The CLKOUT pin is placed in a high-impedance state.

FRZSW —Freeze Software Enable

0 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters continue to run.
1 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters are disabled, preventing interrupts during software debug.

FRZBM —Freeze Bus Monitor Enable

0 = When FREEZE is asserted, the bus monitor continues to operate.
1 = When FREEZE is asserted, the bus monitor is disabled.

SLVEN —Factory Test Mode Enabled

This bit is a read-only status bit that reflects the state of DATA11 during reset.

0 = IMB is not available to an external master.

1 = An external bus master has direct access to the IMB.

SHEN[1:0] —Show Cycle Enable

This field determines what the EBI does with the external bus during internal transfer operations. A show cycle allows internal transfers to be externally monitored. The table below shows whether show cycle data is driven externally, and whether external bus arbitration can occur. To prevent bus conflict, external peripherals must not be enabled during show cycles.

SHEN	Action
00	Show cycles disabled, external arbitration enabled
01	Show cycles enabled, external arbitration disabled
10	Show cycles enabled, external arbitration enabled
11	Show cycles enabled, external arbitration enabled, internal activity is halted by a bus grant

SUPV —Supervisor/Unrestricted Data Space

The SUPV bit places the SIM global registers in either supervisor or user data space.

0 = Registers with access controlled by the SUPV bit are accessible from either the user or supervisor privilege level.

1 = Registers with access controlled by the SUPV bit are restricted to supervisor access only.

MM —Module Mapping

0 = Internal modules are addressed from $7FF000 –$7FFFFF.

1 = Internal modules are addressed from $FFF000 –$FFFFFF.

IARB[3:0] —Interrupt Arbitration Field

Each module that can generate interrupt requests has an interrupt arbitration (IARB) field. Arbitration between interrupt requests of the same priority is performed by serial contention between IARB field bit values. Contention must take place whenever an interrupt request is acknowledged, even when there is only a single pending request. An IARB field must have a non-zero value for contention to take place. If an interrupt request from a module with an IARB field value of %0000 is recognized, the CPU processes a spurious interrupt exception. Because the SIM routes external interrupt requests to the CPU, the SIM IARB field value is used for arbitration between internal and external interrupts of the same priority. The reset value of IARB for the SIM is %1111, and the reset IARB value for all other modules is %0000, which prevents SIM interrupts from being discarded during initialization.

3.2.2 System Protection Control Register

The system protection control register controls system monitor functions, software watchdog clock prescaling, and bus monitor timing. This register can be written only once following power-on or reset, but can be read at any time.

SYPCR —System Protection Control Register **$YFFA21**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	SWE	SWP	SWT		HME	BME	BMT	
RESET:	1	$\overline{\text{MODCLK}}$	0	0	0	0	0	0

SWE —Software Watchdog Enable

0 = Software watchdog disabled

1 = Software watchdog enabled

SWP —Software Watchdog Prescale

This bit controls the value of the software watchdog prescaler.

0 = Software watchdog clock not prescaled
1 = Software watchdog clock prescaled by 512

SWT[1:0] —Software Watchdog Timing

This field selects the divide ratio used to establish software watchdog time-out period. The following table gives the ratio for each combination of SWP and SWT bits.

SWP	SWT	Ratio
0	00	2^9
0	01	2^{11}
0	10	2^{13}
0	11	2^{15}
1	00	2^{18}
1	01	2^{20}
1	10	2^{22}
1	11	2^{24}

HME —Halt Monitor Enable

0 = Disable halt monitor function
1 = Enable halt monitor function

BME —Bus Monitor External Enable

0 = Disable bus monitor function for an internal to external bus cycle.
1 = Enable bus monitor function for an internal to external bus cycle.

BMT[1:0] —Bus Monitor Timing

This field selects a bus monitor time-out period as shown in the following table.

BMT	Bus Monitor Time-out Period
00	64 System Clocks
01	32 System Clocks
10	16 System Clocks
11	8 System Clocks

3.2.3 Bus Monitor

The internal bus monitor checks for excessively long $\overline{\text{DSACK}}$ response times during normal bus cycles and for excessively long $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$ response times during interrupt acknowledge cycles. The monitor asserts $\overline{\text{BERR}}$ if response time is excessive.

$\overline{\text{DSACK}}$ and $\overline{\text{AVEC}}$ response times are measured in clock cycles. The maximum allowable response time can be selected by setting the BMT field.

The monitor does not check $\overline{\text{DSACK}}$ response on the external bus unless the CPU initiates the bus cycle. The BME bit in the SYPCR enables the internal bus monitor for internal to external bus cycles. If a system contains external bus masters, an external bus monitor must be implemented and the internal to external bus monitor option must be disabled.

3.2.4 Halt Monitor

The halt monitor responds to an assertion of $\overline{\text{HALT}}$ on the internal bus. A flag in the reset status register (RSR) indicates that the last reset was caused by the halt monitor. The halt monitor reset can be inhibited by the HME bit in the SYPCR.

3.2.5 Spurious Interrupt Monitor

The spurious interrupt monitor issues $\overline{\text{BERR}}$ if no interrupt arbitration occurs during an interrupt-acknowledge cycle.

3.2.6 Software Watchdog

The software watchdog is controlled by SWE in the SYPCR. Once enabled, the watchdog requires that a service sequence be written to SWSR on a periodic basis. If servicing does not take place, the watchdog times out and issues a reset. This register can be written at any time, but returns zeros when read.

SWSR —Software Service Register **$YFFA27**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	0	0	0	0	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

Register shown with read value

Perform a software watchdog service sequence as follows:

1. Write $55 to SWSR.
2. Write $AA to SWSR.

Both writes must occur before time-out in the order listed, but any number of instructions can be executed between the two writes.

The watchdog clock rate is affected by SWP and SWT in SYPCR. When SWT[1:0] are modified, a watchdog service sequence must be performed before the new time-out period takes effect.

The reset value of SWP is affected by the state of the MODCLK pin on the rising edge of reset, as shown in the following table.

MODCLK	SWP
0	1
1	0

3.2.7 Periodic Interrupt Timer

The periodic interrupt timer (PIT) generates interrupts of specified priorities at specified intervals. Timing for the PIT is provided by a programmable prescaler driven by the system clock.

PICR — Periodic Interrupt Control Register **$YFFA22**

15	14	13	12	11	10 … 8	7 … 0
0	0	0	0	0	PIRQL	PIV
RESET:						
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

This register contains information concerning periodic interrupt priority and vectoring. Bits [10:0] can be read or written at any time. Bits [15:11] are unimplemented and always return zero.

PIRQL[2:0] —Periodic Interrupt Request Level

The following table shows what interrupt request level is asserted when a periodic interrupt is generated. If a PIT interrupt and an external $\overline{\text{IRQ}}$ signal of the same priority occur simultaneously, the PIT interrupt is serviced first. The periodic timer continues to run when the interrupt is disabled.

PIRQL	Interrupt Request Level
000	Periodic Interrupt Disabled
001	Interrupt Request Level 1
010	Interrupt Request Level 2
011	Interrupt Request Level 3
100	Interrupt Request Level 4
101	Interrupt Request Level 5
110	Interrupt Request Level 6
111	Interrupt Request Level 7

PIV[7:0] —Periodic Interrupt Vector

The bits of this field contain the vector generated in response to an interrupt from the periodic timer. When the SIM responds, the periodic interrupt vector is placed on the bus.

PITR —Periodic Interrupt Timer Register **$YFFA24**

15	14	13	12	11	10	9	8	7							0
0	0	0	0	0	0	0	PTP	PITM							
RESET:															
0	0	0	0	0	0	0	$\overline{\text{MODCLK}}$	0	0	0	0	0	0	0	0

The PITR contains the count value for the periodic timer. A zero value turns off the periodic timer. This register can be read or written at any time.

PTP —Periodic Timer Prescaler Control

0 = Periodic timer clock not prescaled

1 = Periodic timer clock prescaled by a value of 512

The reset state of PTP is the complement of the state of the MODCLK signal during reset.

PITM[7:0] —Periodic Interrupt Timing Modulus Field

This is an 8-bit timing modulus. The period of the timer can be calculated as follows:

$$\text{PIT Period} = [(\text{PITM})(\text{Prescaler})(4)]/\text{EXTAL}$$

where

PIT Period = Periodic interrupt timer period

PITM = Periodic interrupt timer register modulus (PITR[7:0])

EXTAL Frequency = Crystal frequency

Prescale = 512 or 1 depending on the state of the PTP bit in the PITR

3.3 System Clock

The system clock in the SIM provides timing signals for the IMB modules and for an external peripheral bus. Because MCU operation is fully static, register and memory contents are not affected when the clock rate changes. System hardware and software support changes in the clock rate during operation.

The system clock signal can be generated in three ways. An internal phase-locked loop can synthesize the clock from an internal or external frequency source, or the clock signal can be input from an external source.

Following is a block diagram of the clock submodule.

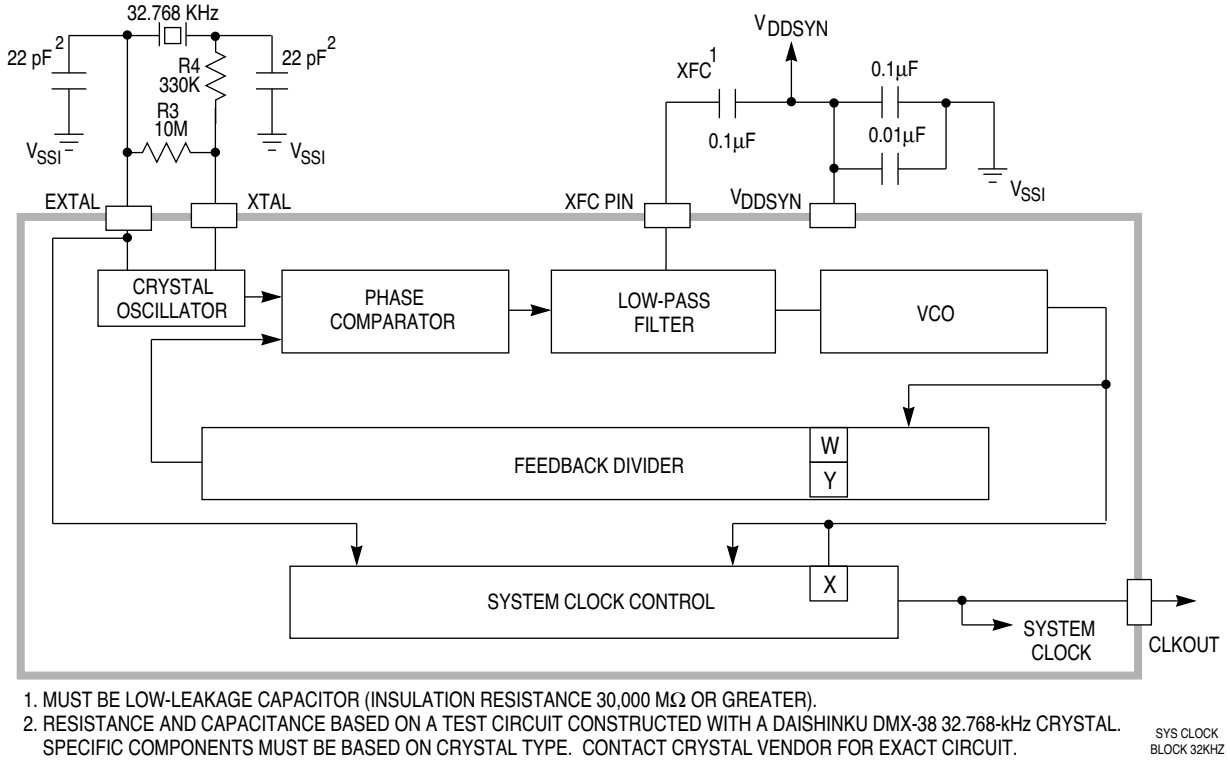


Figure 7 System Clock Block Diagram

3.3.1 Clock Sources

The state of the clock mode (MODCLK) pin during reset determines the clock source. When MODCLK is held high during reset, the clock synthesizer generates a clock signal from either a crystal oscillator or an external reference input. Clock synthesizer control register SYNCR determines operating frequency and various modes of operation. When MODCLK is held low during reset, the clock synthesizer is disabled, and an external system clock signal must be applied. When the synthesizer is disabled, SYNCR control bits have no effect.

A reference crystal must be connected between the EXTAL and XTAL pins to use the internal oscillator. Use of a 32.768-kHz crystal is recommended. These crystals are inexpensive and readily available. If an external reference signal or an external system clock signal is applied through the EXTAL pin, the XTAL pin must be left floating. External reference signal frequency must be less than or equal to maximum specified reference frequency. External system clock signal frequency must be less than or equal to maximum specified system clock frequency.

When an external system clock signal is applied (i.e., the PLL is not used), duty cycle of the input is critical, especially at near maximum operating frequencies. The relationship between clock signal duty cycle and clock signal period is expressed:

$$\text{Minimum external clock period} = \frac{\text{minimum external clock high/low time}}{\text{50\% — percentage variation of external clock input duty cycle}}$$

3.3.2 Clock Synthesizer Operation

A voltage controlled oscillator (VCO) generates the system clock signal. A portion of the clock signal is fed back to a divider/counter. The divider controls the frequency of one input to a phase comparator. The other phase comparator input is a reference signal, either from the internal oscillator or from an external source. The comparator generates a control signal proportional to the difference in phase between its two inputs. The signal is low-pass filtered and used to correct VCO output frequency.

The synthesizer locks when VCO frequency is identical to reference frequency. Lock time is affected by the filter time constant and by the amount of difference between the two comparator inputs. Whenever comparator input changes, the synthesizer must re-lock. Lock status is shown by the SLOCK bit in SYNCR.

The MCU does not come out of reset state until the synthesizer locks. Crystal type, characteristic frequency, and layout of external oscillator circuitry affect lock time.

The low-pass filter requires an external low-leakage capacitor, typically 0.1 µF, connected between the XFC and V_{DDSYN} pins.

V_{DDSYN} is used to power the clock circuits. A separate power source increases MCU noise immunity and can be used to run the clock when the MCU is powered down. Use a quiet power supply as the V_{DDSYN} source, since PLL stability depends on the VCO, which uses this supply. Place adequate external bypass capacitors as close as possible to the V_{DDSYN} pin to ensure stable operating frequency.

When the clock synthesizer is used, control register SYNCR determines operating frequency and various modes of operation. SYNCR can be read only when the processor is operating at the supervisor privilege level.

The SYNCR X bit controls a divide by two prescaler that is not in the synthesizer feedback loop. Setting X doubles clock speed without changing VCO speed. There is no VCO relock delay. The SYNCR W bit controls a 3-bit prescaler in the feedback divider. Setting W increases VCO speed by a factor of four. The SYNCR Y field determines the count modulus for a modulo 64 down counter, causing it to divide by a value of Y + 1. When either W or Y value changes, there is a VCO relock delay.

Clock frequency is determined by SYNCR bit settings as follows:

$$F_{SYSTEM} = F_{REFERENCE}\,[4(Y + 1)(2^{2W+X})]$$

In order for the device to perform correctly, the clock frequency selected by the W, X, and Y bits must be within the limits specified for the MCU. The VCO frequency is twice the system clock frequency if X = 1 or four times the system clock frequency if X = 0.

The reset state of SYNCR ($3F00) produces a modulus-64 count.

3.3.3 Clock Control

The clock control circuits determine system clock frequency and clock operation under special circumstances, such as following loss of synthesizer reference or during low-power operation. Clock source is determined by the logic state of the MODCLK pin during reset.

SYNCR —Clock Synthesizer Control Register **$YFFA04**

15	14	13					8	7	6	5	4	3	2	1	0
W	X	Y						EDIV	0	0	SLIMP	SLOCK	RSTEN	STSIM	STEXT
RESET:															
0	0	1	1	1	1	1	1	0	0	0	U	U	0	0	0

When the on-chip clock synthesizer is used, system clock frequency is controlled by the bits in the upper byte of SYNCR. Bits in the lower byte show status of or control operation of internal and external clocks. The SYNCR can be read or written only when the CPU is operating at the supervisor privilege level.

W —Frequency Control (VCO)
This bit controls a prescaler tap in the synthesizer feedback loop. Setting the bit increases the VCO speed by a factor of four. VCO relock delay is required.

X —Frequency Control Bit (Prescale)
This bit controls a divide by two prescaler that is not in the synthesizer feedback loop. Setting the bit doubles clock speed without changing the VCO speed. There is no VCO relock delay.

Y[5:0] —Frequency Control (Counter)
The Y field controls the modulus down counter in the synthesizer feedback loop, causing it to divide by a value of Y + 1. Values range from 0 to 63. VCO relock delay is required.

EDIV —E Clock Divide Rate
0 = ECLK frequency is system clock divided by 8.
1 = ECLK frequency is system clock divided by 16.
ECLK is an external M6800 bus clock available on pin ADDR23. Refer to 3.5 Chip Selects for more information.

SLIMP —Limp Mode Flag
0 = External crystal is VCO reference.
1 = Loss of crystal reference.
When the on-chip synthesizer is used, loss of reference frequency causes SLIMP to be set. The VCO continues to run using the base control voltage. Maximum limp frequency is maximum specified system clock frequency. X-bit state affects limp frequency.

SLOCK —Synthesizer Lock Flag
0 = VCO is enabled, but has not locked.
1 = VCO has locked on the desired frequency (or system clock is external).
The MCU maintains reset state until the synthesizer locks, but SLOCK does not indicate synthesizer lock status until after the user writes to SYNCR.

RSTEN —Reset Enable
0 = Loss of crystal causes the MCU to operate in limp mode.
1 = Loss of crystal causes system reset.

STSIM —Stop Mode SIM Clock
0 = When LPSTOP is executed, the SIM clock is driven from the crystal oscillator and the VCO is turned off to conserve power.
1 = When LPSTOP is executed, the SIM clock is driven from the VCO.

STEXT —Stop Mode External Clock
0 = When LPSTOP is executed, the CLKOUT signal is held negated to conserve power.
1 = When LPSTOP is executed, the CLKOUT signal is driven from the SIM clock, as determined by the state of the STSIM bit.

3.4 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices. The external bus has 24 address lines and 16 data lines.

The EBI provides dynamic sizing between 8-bit and 16-bit data accesses. It supports byte, word, and long-word transfers. Ports are accessed through the use of asynchronous cycles controlled by the data transfer (SIZ1 and SIZ0) and data size acknowledge pins ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$). Multiple bus cycles may be required for a transfer to or from an 8-bit port.

Port width is the maximum number of bits accepted or provided during a bus transfer. External devices must follow the handshake protocol described below. Control signals indicate the beginning of the cycle, the address space, the size of the transfer, and the type of cycle. The selected device controls the length of the cycle. Strobe signals, one for the address bus and another for the data bus, indicate the validity of an address and provide timing information for data. The EBI operates in an asynchronous mode for any port width.

To add flexibility and minimize the necessity for external logic, MCU chip-select logic can be synchronized with EBI transfers. Chip-select logic can also provide internally-generated bus control signals for these accesses. Refer to **3.5 Chip Selects** for more information.

3.4.1 Bus Control Signals

The CPU initiates a bus cycle by driving the address, size, function code, and read/write outputs. At the beginning of the cycle, size signals SIZ0 and SIZ1 are driven along with the function code signals. The size signals indicate the number of bytes remaining to be transferred during an operand cycle. They are valid while the address strobe ($\overline{AS}$) is asserted. The following table shows SIZ0 and SIZ1 encoding. The read/write (R/$\overline{W}$) signal determines the direction of the transfer during a bus cycle. This signal changes state, when required, at the beginning of a bus cycle, and is valid while $\overline{AS}$ is asserted. R/$\overline{W}$ only changes state when a write cycle is preceded by a read cycle or vice versa. The signal can remain low for two consecutive write cycles.

Table 8 Size Signal Encoding

SIZ1	SIZ0	Transfer Size
0	1	Byte
1	0	Word
1	1	Three Byte
0	0	Long Word

3.4.2 Function Codes

The CPU32 automatically generates function code signals FC[2:0]. The function codes can be considered address extensions that automatically select one of eight address spaces to which an address applies. These spaces are designated as either user or supervisor, and program or data spaces. Address space 7 is designated CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Function codes are valid while $\overline{AS}$ is asserted.

Table 9 CPU32 Address Space Encoding

FC2	FC1	FC0	Address Space
0	0	0	Reserved
0	0	1	User Data Space
0	1	0	User Program Space
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Supervisor Data Space
1	1	0	Supervisor Program Space
1	1	1	CPU Space

3.4.3 Address Bus

Address bus signals ADDR[23:0] define the address of the most significant byte to be transferred during a bus cycle. The MCU places the address on the bus at the beginning of a bus cycle. The address is valid while $\overline{AS}$ is asserted.

3.4.4 Address Strobe

$\overline{AS}$ is a timing signal that indicates the validity of an address on the address bus and the validity of many control signals. It is asserted one-half clock after the beginning of a bus cycle.

3.4.5 Data Bus

Data bus signals DATA[15:0] make up a bidirectional, non-multiplexed parallel bus that transfers data to or from the MCU. A read or write operation can transfer 8 or 16 bits of data in one bus cycle. During a read cycle, the data is latched by the MCU on the last falling edge of the clock for that bus cycle. For a write cycle, all 16 bits of the data bus are driven, regardless of the port width or operand size. The MCU places the data on the data bus one-half clock cycle after $\overline{AS}$ is asserted in a write cycle.

3.4.6 Data Strobe

Data strobe ($\overline{DS}$) is a timing signal. For a read cycle, the MCU asserts $\overline{DS}$ to signal an external device to place data on the bus. $\overline{DS}$ is asserted at the same time as $\overline{AS}$ during a read cycle. For a write cycle, $\overline{DS}$ signals an external device that data on the bus is valid. The MCU asserts $\overline{DS}$ one full clock cycle after the assertion of $\overline{AS}$ during a write cycle.

3.4.7 Bus Cycle Termination Signals

During bus cycles, external devices assert the data transfer and size acknowledge signals ($\overline{DSACK1}$ and $\overline{DSACK0}$). During a read cycle, the signals tell the MCU to terminate the bus cycle and to latch data. During a write cycle, the signals indicate that an external device has successfully stored data and that the cycle can end. These signals also indicate to the MCU the size of the port for the bus cycle just completed. (Refer to 3.4.9 Dynamic Bus Sizing.)

The bus error ($\overline{BERR}$) signal is also a bus cycle termination indicator and can be used in the absence of $\overline{DSACK1}$ and $\overline{DSACK0}$ to indicate a bus error condition. It can also be asserted in conjunction with these signals, provided it meets the appropriate timing requirements. The internal bus monitor can be used to generate the $\overline{BERR}$ signal for internal and internal-to-external transfers. When $\overline{BERR}$ and $\overline{HALT}$ are asserted simultaneously, the CPU takes a bus error exception.

Autovector signal ($\overline{AVEC}$) can terminate external $\overline{IRQ}$ pin interrupt acknowledge cycles. $\overline{AVEC}$ indicates that the MCU will internally generate a vector number to locate an interrupt handler routine. If it is continuously asserted, autovectors will be generated for all external interrupt requests. $\overline{AVEC}$ is ignored during all other bus cycles.

3.4.8 Data Transfer Mechanism

The MCU architecture supports byte, word, and long-word operands, allowing access to 8- and 16-bit data ports through the use of asynchronous cycles controlled by the data transfer and size acknowledge inputs ($\overline{DSACK1}$ and $\overline{DSACK0}$).

3.4.9 Dynamic Bus Sizing

The MCU dynamically interprets the port size of the addressed device during each bus cycle, allowing operand transfers to or from 8- and 16-bit ports. During an operand transfer cycle, the slave device signals its port size and indicates completion of the bus cycle to the MCU through the use of the $\overline{DSACK0}$ and $\overline{DSACK1}$ inputs, as shown in the following table.

Table 10 Effect of $\overline{DSACK}$ Signals

$\overline{DSACK1}$	$\overline{DSACK0}$	Result
1	1	Insert Wait States in Current Bus Cycle
1	0	Complete Cycle —Data Bus Port Size is 8 Bits
0	1	Complete Cycle —Data Bus Port Size is 16 Bits
0	0	Reserved

For example, if the MCU is executing an instruction that reads a long-word operand from a 16-bit port, the MCU latches the 16 bits of valid data and then runs another bus cycle to obtain the other 16 bits. The operation for an 8-bit port is similar, but requires four read cycles. The addressed device uses the $\overline{DSACK0}$ and $\overline{DSACK1}$ signals to indicate the port width. For instance, a 16-bit device always returns $\overline{DSACK0}$ = 1 and $\overline{DSACK1}$ = 0 for a 16-bit port, regardless of whether the bus cycle is a byte or word operation.

Dynamic bus sizing requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 16-bit port must reside on data bus bits [15:0] and an 8-bit port must reside on data bus bits [15:8]. This minimizes the number of bus cycles needed to transfer data and ensures that the MCU transfers valid data.

The MCU always attempts to transfer the maximum amount of data on all bus cycles. For a word operation, it is assumed that the port is 16 bits wide when the bus cycle begins. Operand bytes are designated as shown in the following figure. OP0 is the most significant byte of a long-word operand, and OP3 is the least significant byte. The two bytes of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.

Operand	Byte Order			
	31 24	23 16	15 8	7 0
Long Word	OP0	OP1	OP2	OP3
Three Byte		OP0	OP1	OP2
Word			OP0	OP1
Byte				OP0

Figure 8 Operand Byte Order

3.4.10 Operand Alignment

The data multiplexer establishes the necessary connections for different combinations of address and data sizes. The multiplexer takes the two bytes of the 16-bit bus and routes them to their required positions. Positioning of bytes is determined by the size and address outputs. SIZ1 and SIZ0 indicate the remaining number of bytes to be transferred during the current bus cycle. The number of bytes transferred is equal to or less than the size indicated by SIZ1 and SIZ0, depending on port width.

ADDR0 also affects the operation of the data multiplexer. During an operand transfer, ADDR[23:1] indicate the word base address of the portion of the operand to be accessed, and ADDR0 indicates the byte offset from the base.

3.4.11 Misaligned Operands

CPU32 processor architecture uses a basic operand size of 16 bits. An operand is misaligned when it overlaps a word boundary. This is determined by the value of ADDR0. When ADDR0 = 0 (an even address), the address is on a word and byte boundary. When ADDR0 = 1 (an odd address), the address is on a byte boundary only. A byte operand is aligned at any address; a word or long-word operand is misaligned at an odd address. MC68300 devices do not support misaligned operand transfers.

The largest amount of data that can be transferred by a single bus cycle is an aligned word. If the MCU transfers a long-word operand via a 16-bit port, the most significant operand word is transferred on the first bus cycle and the least significant operand word on a following bus cycle.

3.4.12 Operand Transfer Cases

The following table summarizes how operands are aligned for various types of transfers. OPn entries are portions of a requested operand that are read or written during a bus cycle and are defined by SIZ1, SIZ0, and ADDR0 for that bus cycle.

Table 11 Operand Alignment

Transfer Case	SIZ1	SIZ0	ADDR0	DSACK1	DSACK0	DATA [15:8]	DATA [7:0]
Byte to 8-Bit Port (Even/Odd)	0	1	X	1	0	OP0	(OP0)
Byte to 16-Bit Port (Even)	0	1	0	0	X	OP0	(OP0)
Byte to 16-Bit Port (Odd)	0	1	1	0	X	(OP0)	OP0
Word to 8-Bit Port (Aligned)	1	0	0	1	0	OP0	(OP1)
Word to 8-Bit Port (Misaligned)[3]	1	0	1	1	0	OP0	(OP0)
Word to 16-Bit Port (Aligned)	1	0	0	0	X	OP0	OP1
Word to 16-Bit Port (Misaligned)[3]	1	0	1	0	X	(OP0)	OP0
3 Byte to 8-Bit Port (Aligned)[2]	1	1	0	1	0	OP0	(OP1)
3 Byte to 8-Bit Port (Misaligned)[2, 3]	1	1	1	1	0	OP0	(OP0)
3 Byte to 16-Bit Port (Aligned)[2]	1	1	0	0	X	OP0	OP1
3 Byte to 16-Bit Port (Misaligned)[2, 3]	1	1	1	0	X	(OP0)	OP0
Long Word to 8-Bit Port (Aligned)	0	0	0	1	0	OP0	(OP1)
Long Word to 8-Bit Port (Misaligned)[3]	1	0	1	1	0	OP0	(OP0)
Long Word to 16-Bit Port (Aligned)	0	0	0	0	X	OP0	OP1
Long Word to 16-Bit Port (Misaligned)[3]	1	0	1	0	X	(OP0)	OP0

NOTES:
1. Operands in parentheses are ignored by the CPU32 during read cycles.
2. Three-byte transfer cases occur only as a result of a long word to byte transfer.
3. The CPU32 does not support misaligned word or long-word transfers.

3.5 Chip Selects

Typical microcontrollers require additional hardware to provide external chip-select signals. Twelve independently programmable chip selects provide fast two-cycle access to external memory or peripherals. Address block sizes of 2 Kbytes to 1 Mbyte can be selected.

Chip-select assertion can be synchronized with bus control signals to provide output enable, read/write strobes, or interrupt acknowledge signals. Logic can also generate $\overline{\text{DSACK}}$ signals internally. A single $\overline{\text{DSACK}}$ generator is shared by all circuits. Multiple chip selects assigned to the same address and control must have the same number of wait states.

Chip selects can also be synchronized with the ECLK signal available on ADDR23.

When a memory access occurs, chip-select logic compares address space type, address, type of access, transfer size, and interrupt priority (in the case of interrupt acknowledge) to parameters stored in chip-select registers. If all parameters match, the appropriate chip-select signal is asserted. Select signals are active low. Refer to the following block diagram of a single chip-select circuit.

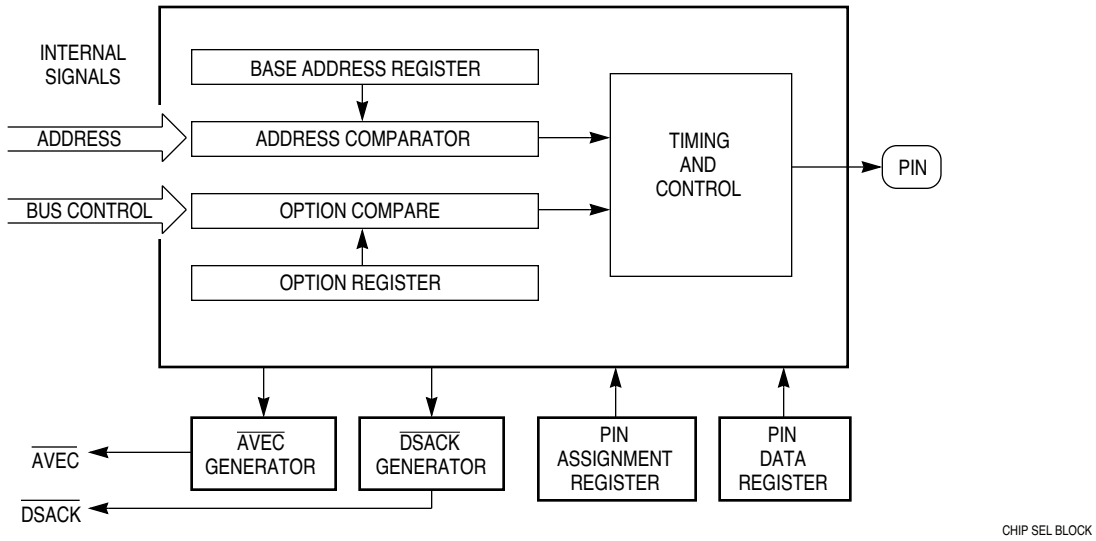


Figure 9 Chip-Select Circuit Block Diagram

The following table lists allocation of chip-selects and discrete outputs on the pins of the MCU.

Pin	Chip Select	Discrete Outputs
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—
$\overline{\text{BR}}$	$\overline{\text{CS0}}$	—
$\overline{\text{BG}}$	$\overline{\text{CS1}}$	—
$\overline{\text{BGACK}}$	$\overline{\text{CS2}}$	—
FC0	$\overline{\text{CS3}}$	PC0
FC1	$\overline{\text{CS4}}$	PC1
FC2	$\overline{\text{CS5}}$	PC2
ADDR19	$\overline{\text{CS6}}$	PC3
ADDR20	$\overline{\text{CS7}}$	PC4
ADDR21	$\overline{\text{CS8}}$	PC5
ADDR22	$\overline{\text{CS9}}$	PC6
ADDR23	$\overline{\text{CS10}}$	ECLK

3.5.1 Chip-Select Registers

Pin assignment registers CSPAR0 and CSPAR1 determine functions of chip-select pins. These registers also determine port size (8- or 16-bit) for dynamic bus allocation.

A pin data register (PORTC) latches discrete output data.

Blocks of addresses are assigned to each chip-select function. Block sizes of 2 Kbytes to 1 Mbyte can be selected by writing values to the appropriate base address register (CSBAR). Address blocks for separate chip-select functions can overlap.

Chip-select option registers (CSORBT and CSOR[10:0]) determine timing of and conditions for assertion of chip-select signals. Eight parameters, including operating mode, access size, synchronization, and wait state insertion can be specified.

Initialization code often resides in a peripheral memory device controlled by the chip-select circuits. A set of special chip-select functions and registers (CSORBT, CSBARBT) is provided to support bootstrap operation.

3.5.2 Pin Assignment Registers

The pin assignment registers (CSPAR0 and CSPAR1) contain pairs of bits that determine the function of chip-select pins. The pin assignment encodings used in these registers are shown below.

Table 12 Pin Assignment Encodings

Bit Field	Description
00	Discrete Output
01	Alternate Function
10	Chip Select (8-Bit Port)
11	Chip Select (16-Bit Port)

CSPAR0 —Chip Select Pin Assignment Register 0 **$YFFA44**

15	14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
0	0	CSPA0[6]	CSPA0[5]	CSPA0[4]	CSPA0[3]	CSPA0[2]	CSPA0[1]	CSBOOT
RESET:								
0	0	DATA2 1	DATA2 1	DATA2 1	DATA1 1	DATA1 1	DATA1 1	1 DATA0

CSPAR0 contains seven 2-bit fields that determine the functions of corresponding chip-select pins. CSPAR0[15:14] are not used. These bits always read zero; writes have no effect. CSPAR0 bit 1 always reads one; writes to CSPAR0 bit 1 have no effect.

Table 13 CSPAR0 Pin Assignments

CSPAR0 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA0[6]	$\overline{CS5}$	FC2	PC2
CSPA0[5]	$\overline{CS4}$	FC1	PC1
CSPA0[4]	$\overline{CS3}$	FC0	PC0
CSPA0[3]	$\overline{CS2}$	$\overline{BGACK}$	—
CSPA0[2]	$\overline{CS1}$	$\overline{BG}$	—
CSPA0[1]	$\overline{CS0}$	$\overline{BR}$	—
$\overline{CSBOOT}$	$\overline{CSBOOT}$	—	—

CSPAR1 —Chip Select Pin Assignment Register 1 **$YFFA46**

15	14	13	12	11	10	9 8	7 6	5 4	3 2	1 0
0	0	0	0	0	0	CSPA1[4]	CSPA1[3]	CSPA1[2]	CSPA1[1]	CSPA1[0]
RESET:										
0	0	0	0	0	0	DATA7 1	DATA [7:6] 1	DATA [7:5] 1	DATA [7:4] 1	DATA [7:3] 1

CSPAR1 contains five 2-bit fields that determine the functions of corresponding chip-select pins. CSPAR1[15:10] are not used. These bits always read zero; writes have no effect.

Table 14 CSPAR1 Pin Assignments

CSPAR0 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA1[4]	$\overline{CS10}$	ADDR23	ECLK
CSPA1[3]	$\overline{CS9}$	ADDR22	PC6
CSPA1[2]	$\overline{CS8}$	ADDR21	PC5
CSPA1[1]	$\overline{CS7}$	ADDR20	PC4
CSPA1[0]	$\overline{CS6}$	ADDR19	PC3

At reset, either the alternate function (01) or chip-select function (11) can be encoded. DATA pins are driven to logic level one by a weak internal pull-up during reset. Encoding is for chip-select function unless a data line is held low during reset. Note that bus loading can overcome the weak pull-up and hold pins low during reset. The following table shows the hierarchical selection method that determines the reset functions of pins controlled by CSPAR1.

Table 15 Reset Pin Function of $\overline{CS}$[10:6]

Data Bus Pins at Reset					Chip-Select/Address Bus Pin Function				
DATA7	DATA6	DATA5	DATA4	DATA3	$\overline{CS10}$/ ADDR23	$\overline{CS9}$/ ADDR22	$\overline{CS8}$/ ADDR21	$\overline{CS7}$/ ADDR20	$\overline{CS6}$/ ADDR19
1	1	1	1	1	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	$\overline{CS7}$	$\overline{CS6}$
1	1	1	1	0	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	$\overline{CS7}$	ADDR19
1	1	1	0	X	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	ADDR20	ADDR19
1	1	0	X	X	$\overline{CS10}$	$\overline{CS9}$	ADDR21	ADDR20	ADDR19
1	0	X	X	X	$\overline{CS10}$	ADDR22	ADDR21	ADDR20	ADDR19
0	X	X	X	X	ADDR23	ADDR22	ADDR21	ADDR20	ADDR19

A pin programmed as a discrete output drives an external signal to the value specified in the port C pin data register (PORTC), with the following exceptions:

A. No discrete output function is available on pins $\overline{BR}$, $\overline{BG}$, or $\overline{BGACK}$.
B. ADDR23 provides E-clock output rather than a discrete output signal.

When a pin is programmed for discrete output or alternate function, internal chip-select logic still functions and can be used to generate $\overline{DSACK}$ or $\overline{AVEC}$ internally on an address match.

Port size is determined when a pin is assigned as a chip select. When a pin is assigned to an 8-bit port, the chip select is asserted at all addresses within the block range. If a pin is assigned to a 16-bit port, the upper/lower byte field of the option register selects the byte with which the chip select is associated.

3.5.3 Base Address Registers

A base address is the starting address for the block enabled by a given chip select. Block size determines the extent of the block above the base address. Each chip select has an associated base register so that an efficient address map can be constructed for each application. If a chip-select base address register is programmed with the same address as a microcontroller module or memory array, an access to that address goes to the module or array and the chip-select signal is not asserted.

CSBARBT — Chip-Select Base Address Register Boot ROM **$YFFA48**

15	14	13	12	11	10	9	8	7	6	5	4	3	2		0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

CSBAR[10:0] —Chip-Select Base Address Registers **$YFFA4C–$YFFA74**

15	14	13	12	11	10	9	8	7	6	5	4	3	2		0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BLKSZ —Block Size Field

This field determines the size of the block that must be enabled by the chip select. The following table shows bit encoding for the base address registers block size field.

Block Size Field	Block Size	Address Lines Compared
000	2 K	ADDR[23:11]
001	8 K	ADDR[23:13]
010	16 K	ADDR[23:14]
011	64 K	ADDR[23:16]
100	128 K	ADDR[23:17]
101	256 K	ADDR[23:18]
110	512 K	ADDR[23:19]
111	1 M	ADDR[23:20]

ADDR[23:11] —Base Address Field

This field sets the starting address of a particular address space. The address compare logic uses only the most significant bits to match an address within a block. The value of the base address must be a multiple of block size. Base address register diagrams show how base register bits correspond to address lines.

3.5.4 Option Registers

The option registers contain eight fields that determine timing of and conditions for assertion of chip-select signals. For a chip-select signal to be asserted, all bits in the base address register must match the corresponding internal upper address lines, and all conditions specified in the option register must be satisfied. These conditions also apply to providing $\overline{\text{DSACK}}$ or autovector support.

CSORBT —Chip-Select Option Register Boot ROM **$YFFA4A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE		R/$\overline{\text{W}}$		STRB	DSACK				SPACE		IPL			$\overline{\text{AVEC}}$
RESET:															
0	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0

CSOR[10:0] —Chip-Select Option Registers **$YFFA4E–$YFFA76**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE		R/$\overline{\text{W}}$		STRB	DSACK				SPACE		IPL			$\overline{\text{AVEC}}$
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSORBT, the option register for $\overline{\text{CSBOOT}}$, contains special reset values that support bootstrap operations from peripheral memory devices.

The following bit descriptions apply to both CSORBT and CSOR[10:0] option registers.

MODE —Asynchronous/Synchronous Mode

0 = Asynchronous mode selected (chip-select assertion determined by internal or external bus control signals)

1 = Synchronous mode selected (chip-select assertion synchronized with ECLK signal)

In asynchronous mode, the chip select is asserted synchronized with $\overline{\text{AS}}$ or $\overline{\text{DS}}$.

The $\overline{\text{DSACK}}$ field is not used in synchronous mode because a bus cycle is only performed as a synchronous operation. When a match condition occurs on a chip select programmed for synchronous operation, the chip select signals the EBI that an ECLK cycle is pending.

BYTE —Upper/Lower Byte Option

This field is used only when the chip-select 16-bit port option is selected in the pin assignment register. The following table lists upper/lower byte options.

Byte	Description
00	Disable
01	Lower Byte
10	Upper Byte
11	Both Bytes

R/$\overline{W}$ —Read/Write

This field causes a chip select to be asserted only for a read, only for a write, or for both read and write. Refer to the following table for options available.

R/$\overline{W}$	Description
00	Reserved
01	Read Only
10	Write Only
11	Read/Write

STRB —Address Strobe/Data Strobe

0 = Address strobe

1 = Data strobe

This bit controls the timing for assertion of a chip select in asynchronous mode. Selecting address strobe causes chip select to be asserted synchronized with address strobe. Selecting data strobe causes chip select to be asserted synchronized with data strobe.

$\overline{DSACK}$ —Data and Size Acknowledge

This field specifies the source of $\overline{DSACK}$ in asynchronous mode. It also allows the user to adjust bus timing with internal $\overline{DSACK}$ generation by controlling the number of wait states that are inserted to optimize bus speed in a particular application. The following table shows the $\overline{DSACK}$ field encoding. The fast termination encoding (1110) is used for two-cycle access to external memory.

$\overline{DSACK}$	Description
0000	No Wait States
0001	1 Wait State
0010	2 Wait States
0011	3 Wait States
0100	4 Wait States
0101	5 Wait States
0110	6 Wait States
0111	7 Wait States
1000	8 Wait States
1001	9 Wait States
1010	10 Wait States
1011	11 Wait States
1100	12 Wait States
1101	13 Wait States
1110	Fast Termination
1111	External $\overline{DSACK}$

SPACE —Address Space

Use this option field to select an address space for the chip-select logic. The CPU32 normally operates in supervisor or user space, but interrupt acknowledge cycles must take place in CPU space.

Space Field	Address Space
00	CPU Space
01	User Space
10	Supervisor Space
11	Supervisor/User Space

IPL —Interrupt Priority Level

If the space field is set for CPU space (00), chip-select logic can be used for interrupt acknowledge. During an interrupt acknowledge cycle, the priority level on address lines ADDR[3:1] is compared to the value in the IPL field. If the values are the same, a chip select is asserted, provided that other option register conditions are met. The following table shows IPL field encoding.

IPL	Description
000	Any Level
001	IPL1
010	IPL2
011	IPL3
100	IPL4
101	IPL5
110	IPL6
111	IPL7

This field only affects the response of chip selects and does not affect interrupt recognition by the CPU. Any level means that chip select is asserted regardless of the level of the interrupt acknowledge cycle.

$\overline{\text{AVEC}}$ —Autovector Enable

0 = External interrupt vector enabled

1 = Autovector enabled

This field selects one of two methods of acquiring the interrupt vector during the interrupt acknowledge cycle. It is not usually used in conjunction with a chip-select pin.

If the chip select is configured to trigger on an interrupt acknowledge cycle (SPACE = 00) and the $\overline{\text{AVEC}}$ field is set to one, the chip select automatically generates an $\overline{\text{AVEC}}$ in response to the interrupt cycle. Otherwise, the vector must be supplied by the requesting device.

The $\overline{\text{AVEC}}$ bit must not be used in synchronous mode, as autovector response timing can vary because of ECLK synchronization.

3.5.5 Port C Data Register

Bit values in port C determine the state of chip-select pins used for discrete output. When a pin is assigned as a discrete output, the value in this register appears at the output. This is a read/write register. Bit 7 is not used. Writing to this bit has no effect, and it always returns zero when read.

PORTC — Port C Data Register **$YFFA41**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	0	PC6	PC5	PC4	PC3	PC2	PC1	PC0
RESET:	0	1	1	1	1	1	1	1

3.6 General-Purpose Input/Output

SIM pins can be configured as two general-purpose I/O ports, E and F. The following paragraphs describe registers that control the ports.

PORTE0, PORTE1 —Port E Data Register **$YFFA11, $YFFA13**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
RESET:	U	U	U	U	U	U	U	U

A write to the port E data register is stored in the internal data latch and, if any port E pin is configured as an output, the value stored for that bit is driven on the pin. A read of the port E data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register.

The port E data register is a single register that can be accessed in two locations. When accessed at $YFFA11, the register is referred to as PORTE0; when accessed at $YFFA13, the register is referred to as PORTE1. The register can be read or written at any time. It is unaffected by reset.

DDRE — Port E Data Direction Register **$YFFA15**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
RESET:	0	0	0	0	0	0	0	0

The bits in this register control the direction of the pin drivers when the pins are configured as I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input. This register can be read or written at any time.

PEPAR — Port E Pin Assignment Register **$YFFA17**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	PEPA7	PEPA6	PEPA5	PEPA4	PEPA3	PEPA2	PEPA1	PEPA0
RESET:	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8

The bits in this register control the function of each port E pin. Any bit set to one configures the corresponding pin as a bus control signal, with the function shown in the following table. Any bit cleared to zero defines the corresponding pin to be an I/O pin, controlled by PORTE and DDRE.

Data bus bit 8 controls the state of this register following reset. If DATA8 is set to one during reset, the register is set to $FF, which defines all port E pins as bus control signals. If DATA8 is cleared to zero during reset, this register is set to $00, configuring all port E pins as I/O pins.

Any bit cleared to zero defines the corresponding pin to be an I/O pin. Any bit set to one defines the corresponding pin to be a bus control signal.

Table 16 Port E Pin Assignments

PEPAR Bit	Port E Signal	Bus Control Signal
PEPA7	PE7	SIZ1
PEPA6	PE6	SIZ0
PEPA5	PE5	$\overline{AS}$
PEPA4	PE4	$\overline{DS}$
PEPA3	PE3	$\overline{RMC}$
PEPA2	PE2	$\overline{AVEC}$
PEPA1	PE1	$\overline{DSACK1}$
PEPA0	PE0	$\overline{DSACK0}$

PORTF0, PORTF1 — Port F Data Register **$YFFA19, $YFFA1B**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
RESET:	U	U	U	U	U	U	U	U

The write to the port F data register is stored in the internal data latch, and if any port F pin is configured as an output, the value stored for that bit is driven onto the pin. A read of the port F data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register.

The port F data register is a single register that can be accessed in two locations. When accessed at $YFFA19, the register is referred to as PORTF0; when accessed at $YFFA1B, the register is referred to as PORTF1. The register can be read or written at any time. It is unaffected by reset.

DDRF — Port F Data Direction Register **$YFFA1D**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
RESET:	0	0	0	0	0	0	0	0

The bits in this register control the direction of the pin drivers when the pins are configured for I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input.

PFPAR — Port F Pin Assignment Register **$YFFA1F**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	PFPA7	PFPA6	PFPA5	PFPA4	PFPA3	PFPA2	PFPA1	PFPA0
RESET:	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9

The bits in this register control the function of each port F pin. Any bit cleared to zero defines the corresponding pin to be an I/O pin. Any bit set to one defines the corresponding pin to be an interrupt request signal or MODCLK. The MODCLK signal has no function after reset.

Table 17 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA7	PF7	$\overline{\text{IRQ7}}$
PFPA6	PF6	$\overline{\text{IRQ6}}$
PFPA5	PF5	$\overline{\text{IRQ5}}$
PFPA4	PF4	$\overline{\text{IRQ4}}$
PFPA3	PF3	$\overline{\text{IRQ3}}$
PFPA2	PF2	$\overline{\text{IRQ2}}$
PFPA1	PF1	$\overline{\text{IRQ1}}$
PFPA0	PF0	MODCLK

Data bus pin 9 controls the state of this register following reset. If DATA9 is set to one during reset, the register is set to $FF, which defines all port F pins as interrupt request inputs. If DATA9 is cleared to zero during reset, this register is set to $00, defining all port F pins as I/O pins.

3.7 Resets

Reset procedures handle system initialization and recovery from catastrophic failure. The MCU performs resets with a combination of hardware and software. The system integration module determines whether a reset is valid, asserts control signals, performs basic system configuration based on hardware mode-select inputs, then passes control to the CPU.

Reset occurs when an active low logic level on the $\overline{\text{RESET}}$ pin is clocked into the SIM. Resets are gated by the CLKOUT signal. Asynchronous resets are assumed to be catastrophic. An asynchronous reset can occur on any clock edge. Synchronous resets are timed to occur at the end of bus cycles. If there is no clock when $\overline{\text{RESET}}$ is asserted, reset does not occur until the clock starts. Resets are clocked in order to allow completion of write cycles in progress at the time $\overline{\text{RESET}}$ is asserted.

Reset is the highest-priority CPU32 exception. Any processing in progress is aborted by the reset exception, and cannot be restarted. Only essential tasks are performed during reset exception processing. Other initialization tasks must be accomplished by the exception handler routine.

3.7.1 SIM Reset Mode Selection

The logic states of certain data bus pins during reset determine SIM operating configuration. In addition, the state of the MODCLK pin determines system clock source and the state of the $\overline{\text{BKPT}}$ pin determines what happens during subsequent breakpoint assertions. The following table is a summary of reset mode selection options.

Table 18 Reset Mode Selection

Mode Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)
DATA0	$\overline{\text{CSBOOT}}$ 16-Bit	$\overline{\text{CSBOOT}}$ 8-Bit
DATA1	$\overline{\text{CS0}}$ $\overline{\text{CS1}}$ $\overline{\text{CS2}}$	$\overline{\text{BR}}$ $\overline{\text{BG}}$ $\overline{\text{BGACK}}$
DATA2	$\overline{\text{CS3}}$ $\overline{\text{CS4}}$ $\overline{\text{CS5}}$	FC0 FC1 FC2
DATA3 DATA4 DATA5 DATA6 DATA7	$\overline{\text{CS6}}$ $\overline{\text{CS[7:6]}}$ $\overline{\text{CS[8:6]}}$ $\overline{\text{CS[9:6]}}$ $\overline{\text{CS[10:6]}}$	ADDR19 ADDR[20:19] ADDR[21:19] ADDR[22:19] ADDR[23:19]
DATA8	$\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$, $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, $\overline{\text{SIZ[1:0]}}$	PORTE
DATA9	$\overline{\text{IRQ[7:1]}}$ MODCLK	PORTF
DATA11	Test Mode Disabled	Test Mode Enabled
MODCLK	VCO = System Clock	EXTAL = System Clock
$\overline{\text{BKPT}}$	Background Mode Disabled	Background Mode Enabled

3.7.2 Functions of Pins for Other Modules During Reset

Generally, pins associated with modules other than the SIM default to port functions, and input/output ports are set to input state. This is accomplished by disabling pin functions in the appropriate control registers, and by clearing the appropriate port data direction registers. Refer to individual module sections in this manual for more information. The following table is a summary of module pin function out of reset.

Table 19 Module Pin Functions

Module	Pin Mnemonic	Function
CPU32	DSI/IFETCH	DSI/IFETCH
	DSO/IPIPE	DSO/IPIPE
	BKPT/DSCLK	BKPT/DSCLK
GPT	PGP7/IC4/OC5	Discrete Input
	PGP[6:3]/OC[4:1]	Discrete Input
	PGP[2:0]/IC[3:1]	Discrete Input
	PAI	Discrete Input
	PCLK	Discrete Input
	PWMA, PWMB	Discrete Output
QSM	PQS7/TXD	Discrete Input
	PQS[6:4]/PCS[3:1]	Discrete Input
	PQS3/PCS0/SS	Discrete Input
	PQS2/SCK	Discrete Input
	PQS1/MOSI	Discrete Input
	PQS0/MISO	Discrete Input
	RXD	RXD

3.7.3 Reset Timing

The RESET input must be asserted for a specified minimum period in order for reset to occur. External RESET assertion can be delayed internally for a period equal to the longest bus cycle time (or the bus monitor time-out period) in order to protect write cycles from being aborted by reset. While RESET is asserted, SIM pins are either in a disabled high-impedance state or are driven to their inactive states.

When an external device asserts RESET for the proper period, reset control logic clocks the signal into an internal latch. The control logic drives the RESET pin low for an additional 512 CLKOUT cycles after it detects that the RESET signal is no longer being externally driven, to guarantee this length of reset to the entire system.

If an internal source asserts a reset signal, the reset control logic asserts RESET for a minimum of 512 cycles. If the reset signal is still asserted at the end of 512 cycles, the control logic continues to assert RESET until the internal reset signal is negated.

After 512 cycles have elapsed, the reset input pin goes to an inactive, high-impedance state for 10 cycles. At the end of this 10-cycle period, the reset input is tested. When the input is at logic level one, reset exception processing begins. If, however, the reset input is at logic level zero, the reset control logic drives the pin low for another 512 cycles. At the end of this period, the pin again goes to high-impedance state for 10 cycles, then it is tested again. The process repeats until RESET is released.

3.7.4 Power-On Reset

When the SIM clock synthesizer is used to generate the system clock, power-on reset involves special circumstances related to application of system and clock synthesizer power. Regardless of clock source, voltage must be applied to clock synthesizer power input pin V_{DDSYN} in order for the MCU to operate. The following discussion assumes that V_{DDSYN} is applied before and during reset. This minimizes crystal start-up time. When V_{DDSYN} is applied at power-on, start-up time is affected by specific crystal parameters and by oscillator circuit design. V_{DD} ramp-up time also affects pin state during reset.

During power-on reset, an internal circuit in the SIM drives the internal (IMB) and external reset lines. The circuit releases the internal reset line as V_{DD} ramps up to the minimum specified value, and SIM pins are initialized. When V_{DD} reaches the specified minimum value, the clock synthesizer VCO begins operation. Clock frequency ramps up to the specified limp mode frequency. The external RESET line remains asserted until the clock synthesizer PLL locks and 512 CLKOUT cycles elapse.

The SIM clock synthesizer provides clock signals to the other MCU modules. After the clock is running and the internal reset signal is asserted for four clock cycles, these modules reset. V_{DD} ramp time and VCO frequency ramp time determine how long these four cycles take. Worst case is approximately 15 milliseconds. During this period, module port pins may be in an indeterminate state. While input-only pins can be put in a known state by means of external pull-up resistors, external logic on input/output or output-only pins must condition the lines during this time. Active drivers require high-impedance buffers or isolation resistors to prevent conflict.

3.7.5 Use of Three State Control Pin

Asserting the three-state control (TSC) input causes the MCU to put all output drivers in an inactive, high-impedance state. The signal must remain asserted for 10 clock cycles in order for drivers to change state. There are certain constraints on use of TSC during power-on reset:

When the internal clock synthesizer is used (MODCLK held high during reset), synthesizer ramp-up time affects how long the 10 cycles take. Worst case is approximately 20 milliseconds from TSC assertion.

When an external clock signal is applied (MODCLK held low during reset), pins go to high-impedance state as soon after TSC assertion as 10 clock pulses have been applied to the EXTAL pin.

When TSC assertion takes effect, internal signals are forced to values that can cause inadvertent mode selection. Once the output drivers change state, the MCU must be powered down and restarted before normal operation can resume.

3.8 Interrupts

Interrupt recognition and servicing involve complex interaction between the central processing unit, the system integration module, and a device or module requesting interrupt service.

The CPU32 provides for eight levels of interrupt priority (0–7), seven automatic interrupt vectors and 200 assignable interrupt vector. All interrupts with priorities less than 7 can be masked by the interrupt priority (IP) field in the status register. The CPU32 handles interrupts as a type of asynchronous exception.

Interrupt recognition is based on the states of interrupt request signals $\overline{\text{IRQ}}$[7:1] and the IP mask value. Each of the signals corresponds to an interrupt priority. $\overline{\text{IRQ1}}$ has the lowest priority, and $\overline{\text{IRQ7}}$ has the highest priority.

The IP field consists of three bits. Binary values %000 to %111 provide eight priority masks. Masks prevent an interrupt request of a priority less than or equal to the mask value (except for $\overline{\text{IRQ7}}$) from being recognized and processed. When IP contains %000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

Interrupt request signals can be asserted by external devices or by microcontroller modules. Request lines are connected internally by means of a wired NOR — simultaneous requests of differing priority can be made. Internal assertion of an interrupt request signal does not affect the logic state of the corresponding MCU pin.

External interrupt requests are routed to the CPU via the external bus interface and SIM interrupt control logic. The CPU treats external interrupt requests as though they come from the SIM.

External $\overline{\text{IRQ}}$[6:1] are active-low level-sensitive inputs. External $\overline{\text{IRQ7}}$ is an active-low transition-sensitive input. $\overline{\text{IRQ7}}$ requires both an edge and a voltage level for validity.

$\overline{\text{IRQ}}$[6:1] are maskable. $\overline{\text{IRQ7}}$ is nonmaskable. The $\overline{\text{IRQ7}}$ input is transition-sensitive in order to prevent redundant servicing and stack overflow. A nonmaskable interrupt is generated each time $\overline{\text{IRQ7}}$ is asserted, and each time the priority mask changes from %111 to a lower number while $\overline{\text{IRQ7}}$ is asserted.

Interrupt requests are sampled on consecutive falling edges of the system clock. Interrupt request input circuitry has hysteresis. To be valid, a request signal must be asserted for at least two consecutive clock periods. Valid requests do not cause immediate exception processing, but are left pending. Pending requests are processed at instruction boundaries or when exception processing of higher-priority exceptions is complete.

The CPU32 does not latch the priority of a pending interrupt request. If an interrupt source of higher priority makes a service request while a lower priority request is pending, the higher priority request is serviced. If an interrupt request of equal or lower priority than the current IP mask value is made, the CPU does not recognize the occurrence of the request in any way.

3.8.1 Interrupt Acknowledge and Arbitration

Interrupt acknowledge bus cycles are generated during exception processing. When the CPU detects one or more interrupt requests of a priority higher than the interrupt priority mask value, it performs a CPU space read from address $FFFFF : [IP] : 1.

The CPU space read cycle performs two functions: it places a mask value corresponding to the highest priority interrupt request on the address bus, and it acquires an exception vector number from the interrupt source. The mask value also serves two purposes: it is latched into the CCR IP field in order to mask lower-priority interrupts during exception processing, and it is decoded by modules that have requested interrupt service to determine whether the current interrupt acknowledge cycle pertains to them.

Modules that have requested interrupt service decode the IP value placed on the address bus at the beginning of the interrupt acknowledge cycle, and if their requests are at the specified IP level, respond to the cycle. Arbitration between simultaneous requests of the same priority is performed by means of serial contention between module interrupt arbitration (IARB) field bit values.

Each module that can make an interrupt service request, including the SIM, has an IARB field in its configuration register. An IARB field can be assigned a value from %0001 (lowest priority) to %1111 (highest priority). A value of %0000 in an IARB field causes the CPU to process a spurious interrupt exception when an interrupt from that module is recognized.

Because the EBI manages external interrupt requests, the SIM IARB value is used for arbitration between internal and external interrupt requests. The reset value of IARB for the SIM is %1111, and the reset IARB value for all other modules is %0000. Initialization software must assign different IARB values in order to implement an arbitration scheme.

Each module must have a unique IARB value. When two or more IARB fields have the same nonzero value, the CPU interprets multiple vector numbers simultaneously, with unpredictable consequences.

Arbitration must always take place, even when a single source requests service. This point is important for two reasons: the CPU interrupt acknowledge cycle is not driven on the external bus unless the SIM wins contention, and failure to contend causes an interrupt acknowledge bus cycle to be terminated by a bus error, which causes a spurious interrupt exception to be taken.

When arbitration is complete, the dominant module must place an interrupt vector number on the data bus and terminate the bus cycle. In the case of an external interrupt request, because the interrupt acknowledge cycle is transferred to the external bus, an external device must decode the mask value and respond with a vector number, then generate bus cycle termination signals. If the device does not respond in time, a spurious interrupt exception is taken.

The periodic interrupt timer (PIT) in the SIM can generate internal interrupt requests of specific priority at predetermined intervals. By hardware convention, PIT interrupts are serviced before external interrupt service requests of the same priority. Refer to **3.2.7 Periodic Interrupt Timer** for more information.

3.8.2 Interrupt Processing Summary

A summary of the interrupt processing sequence follows. When the sequence begins, a valid interrupt service request has been detected and is pending.

A. The CPU finishes higher priority exception processing or reaches an instruction boundary.
B. Processor state is stacked. The contents of the status register and program counter are saved.
C. The interrupt acknowledge cycle begins:
   1. FC[2:0] are driven to %111 (CPU space) encoding.
   2. The address bus is driven as follows. ADDR[23:20] = %1111; ADDR[19:16] = %1111, which indicates that the cycle is an interrupt acknowledge CPU space cycle; ADDR[15:4] = %111111111111; ADDR[3:1] = the level of the interrupt request being acknowledged; and ADDR0 = %1.
   3. Request priority level is latched into the IP field in the status register from the address bus.
D. Modules or external peripherals that have requested interrupt service decode the request level in ADDR[3:1]. If the request level of at least one interrupting module or device is the same as the value in ADDR[3:1], interrupt arbitration contention takes place. When there is no contention, the spurious interrupt monitor asserts $\overline{\text{BERR}}$, and a spurious interrupt exception is processed.
E. After arbitration, the interrupt acknowledge cycle can be completed in one of three ways:
   1. The dominant interrupt source supplies a vector number and $\overline{\text{DSACK}}$ signals appropriate to the access. The CPU32 acquires the vector number.
   2. The $\overline{\text{AVEC}}$ signal is asserted (the signal can be asserted by the dominant interrupt source or the pin can be tied low), and the CPU32 generates an autovector number corresponding to interrupt priority.
   3. The bus monitor asserts $\overline{\text{BERR}}$ and the CPU32 generates the spurious interrupt vector number.
F. The vector number is converted to a vector address.
G. The content of the vector address is loaded into the PC, and the processor transfers control to the exception handler routine.

3.9 Factory Test Block

The test submodule supports scan-based testing of the various MCU modules. It is integrated into the SIM to support production testing.

Test submodule registers are intended for Motorola use. Register names and addresses are provided to indicate that these addresses are occupied.

SIMTR — System Integration Module Test Register **$YFFA02**

SIMTRE — System Integration Module Test Register (E Clock) **$YFFA08**

TSTMSRA — Master Shift Register A **$YFFA30**

TSTMSRB — Master Shift Register B **$YFFA32**

TSTSC — Test Module Shift Count **$YFFA34**

TSTRC — Test Module Repetition Count **$YFFA36**

CREG — Test Module Control Register **$YFFA38**

DREG — Test Module Distributed Register **$YFFA3A**

4 Central Processor Unit

Based on the powerful MC68020, the CPU32 processing module provides enhanced system performance and also uses the extensive software base for the Motorola M68000 family.

4.1 Overview

The CPU32 is fully object code compatible with the M68000 Family, which excels at processing calculation-intensive algorithms and supporting high-level languages. The CPU32 supports all of the MC68010 and most of the MC68020 enhancements, such as virtual memory support, loop mode operation, instruction pipeline, and 32-bit mathematical operations. Powerful addressing modes provide compatibility with existing software programs and increase the efficiency of high-level language compilers. Special instructions, such as table lookup and interpolate and low-power stop, support the specific requirements of controller applications. Also included is the background debugging mode, an alternate operating mode that suspends normal operation and allows the CPU to accept debugging commands from the development system.

Ease of programming is an important consideration in using a microcontroller. The CPU32 instruction set is optimized for high performance. The eight 32-bit general-purpose data registers readily support 8-bit (byte), 16-bit (word), and 32-bit (long word) operations. Ease of program checking and diagnosis is further enhanced by trace and trap capabilities at the instruction level.

Use of high-level languages is increasing as controller applications become more complex and control programs become larger. High-level languages aid rapid development of software, with less error, and are readily portable. The CPU32 instruction set supports high-level languages.

4.2 Programming Model

The CPU32 has sixteen 32-bit general registers, a 32-bit program counter, one 32-bit supervisor stack pointer, a 16-bit status register, two alternate function code registers, and a 32-bit vector base register.

The programming model of the CPU32 consists of a user model and supervisor model, corresponding to the user and supervisor privilege levels. Some instructions available at the supervisor level are not available at the user level, allowing the supervisor to protect system resources from uncontrolled access. Bit S in the status register determines the privilege level.

The user programming model remains unchanged from previous M68000 Family microprocessors. Application software written to run at the nonprivileged user level migrates without modification to the CPU32 from any M68000 platform. The move from SR instruction, however, is privileged in the CPU32. It is not privileged in the M68000.

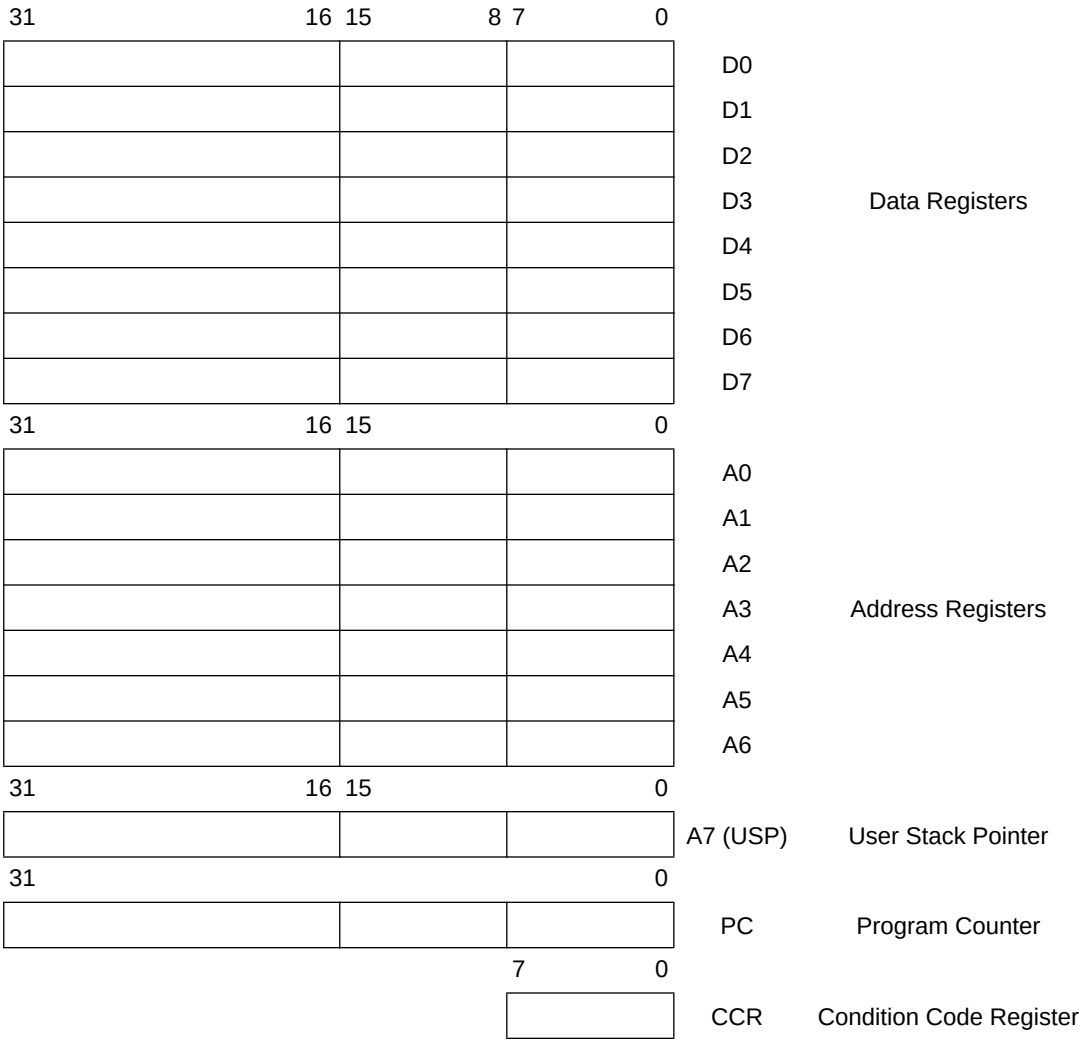


Figure 10 User Programming Model

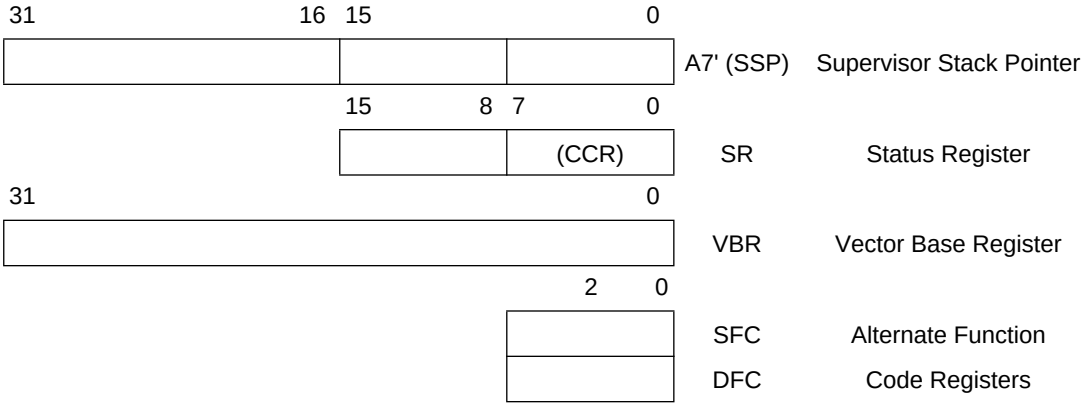


Figure 11 Supervisor Programming Model Supplement

4.3 Status Register

The status register contains the condition codes that reflect the results of a previous operation and can be used for conditional instruction execution in a program. The lower byte containing the condition codes is the only portion of the register available at the user privilege level; it is referenced as the condition code register (CCR) in user programs. At the supervisor privilege level, software can access the full status register, including the interrupt priority mask and additional control bits.

SR —Status Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T1	T0	S	0	0	IP			0	0	0	X	N	Z	V	C
RESET:															
0	0	1	0	0	1	1	1	0	0	0	U	U	U	U	U

System Byte
- T[1:0] —Trace Enable
- S —Supervisor/User State
- Bits [12:11] —Unimplemented
- IP[2:0] —Interrupt Priority Mask

User Byte (Condition Code Register)
- Bits [7:5] —Unimplemented
- X —Extend
- N —Negative
- Z —Zero
- V —Overflow
- C —Carry

4.4 Data Types

Six basic data types are supported:

- Bits
- Packed Binary Coded Decimal Digits
- Byte Integers (8 bits)
- Word Integers (16 bits)
- Long-Word Integers (32 bits)
- Quad-Word Integers (64 bits)

4.5 Addressing Modes

Addressing in the CPU32 is register-oriented. Most instructions allow the results of the specified operation to be placed either in a register or directly in memory. This flexibility eliminates the need for extra instructions to store register contents in memory. The CPU32 supports seven basic addressing modes:

- Register direct
- Register indirect
- Register indirect with index
- Program counter indirect with displacement
- Program counter indirect with index
- Absolute
- Immediate

Included in the register indirect addressing modes are the capabilities to post-increment, predecrement, and offset. The program counter relative mode also has index and offset capabilities. In addition to these addressing modes, many instructions implicitly specify the use of the status register, stack pointer, or program counter.

4.6 Instruction Set Summary

Table 20 Instruction Set Summary

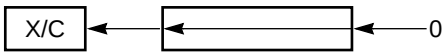
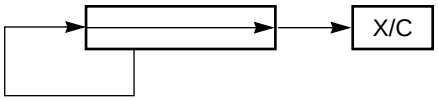
Instruction	Syntax	Operand Size	Operation
ABCD	Dn, Dn – (An), – (An)	8 8	$Source_{10} + Destination_{10} + X \Rightarrow Destination$
ADD	Dn, <ea> <ea>, Dn	8, 16, 32 8, 16, 32	Source + Destination ⇒ Destination
ADDA	<ea>, An	16, 32	Source + Destination ⇒ Destination
ADDI	#<data>, <ea>	8, 16, 32	Immediate data + Destination ⇒ Destination
ADDQ	# <data>, <ea>	8, 16, 32	Immediate data + Destination ⇒ Destination
ADDX	Dn, Dn – (An), – (An)	8, 16, 32 8, 16, 32	Source + Destination + X ⇒ Destination
AND	<ea>, Dn Dn, <ea>	8, 16, 32 8, 16, 32	Source • Destination ⇒ Destination
ANDI	# <data>, <ea>	8, 16, 32	Data • Destination ⇒ Destination
ANDI to CCR	# <data>, CCR	8	Source • CCR ⇒ CCR
ANDI to SR[1]	# <data>, SR	16	Source • SR ⇒ SR
ASL	Dn, Dn # <data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
ASR	Dn, Dn # <data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
Bcc	label	8, 16, 32	If condition true, then PC + d ⇒ PC
BCHG	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{\text{bit number}\rangle\text{of destination}} \Rightarrow Z \Rightarrow$ bit of destinatic
BCLR	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{(\langle\text{bit number}\rangle\text{of destination})}$ 0 ⇒ $\overline{\text{bit of destination}}$
BGND	none	none	If background mode enabled, then enter background mode, else format/vector ⇒ – (SSP); PC ⇒ – (SSP); SR ⇒ – (SSP); (vector) ⇒ PC
BKPT	# <data>	none	If breakpoint cycle acknowledged, then execute returned operation word, else trap as illegal instruction
BRA	label	8, 16, 32	PC + d ⇒ PC
BSET	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{(\langle\text{bit number}\rangle\text{of destination})} \Rightarrow Z$; 1 ⇒ bit of destination
BSR	label	8, 16, 32	SP – 4 ⇒ SP; PC ⇒ (SP); PC + d ⇒ PC
BTST	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{(\langle\text{bit number}\rangle\text{of destination})} \Rightarrow Z$
CHK	<ea>, Dn	16, 32	If Dn < 0 or Dn > (ea), then CHK exception
CHK2	<ea>, Rn	8, 16, 32	If Rn < lower bound or Rn > upper bound, then CHK exception
CLR	<ea>	8, 16, 32	0 ⇒ Destination
CMP	<ea>, Dn	8, 16, 32	(Destination – Source), CCR shows results
CMPA	<ea>, An	16, 32	(Destination – Source), CCR shows results
CMPI	# <data>, <ea>	8, 16, 32	(Destination – Data), CCR shows results

Table 20 Instruction Set Summary (Continued)

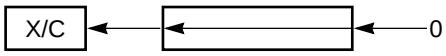
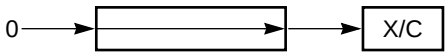
Instruction	Syntax	Operand Size	Operation
CMPM	(An) +, (An) +	8, 16, 32	(Destination – Source), CCR shows results
CMP2	<ea>, Rn	8, 16, 32	Lower bound ≤ Rn ≤ Upper bound, CCR shows result
DBcc	Dn, label	16	If condition false, then Dn – 1 ⇒ PC; if Dn ≠ (– 1), then PC + d ⇒ PC
DIVS/DIVU	<ea>, Dn	32/16 ⇒ 16 : 16	Destination / Source ⇒ Destination (signed or unsigned)
DIVSL/DIVUL	<ea>, Dr : Dq <ea>, Dq <ea>, Dr : Dq	64/32 ⇒ 32 : 32 32/32 ⇒ 32 32/32 ⇒ 32 : 32	Destination / Source ⇒ Destination (signed or unsigned)
EOR	Dn, <ea>	8, 16, 32	Source ⊕ Destination ⇒ Destination
EORI	# <data>, <ea>	8, 16, 32	Data ⊕ Destination ⇒ Destination
EORI to CCR	# <data>, CCR	8	Source ⊕ CCR ⇒ CCR
EORI to SR[1]	# <data>, SR	16	Source ⊕ SR ⇒ SR
EXG	Rn, Rn	32	Rn ⇒ Rn
EXT	Dn Dn	8 ⇒ 16 16 ⇒ 32	Sign extended Destination ⇒ Destination
EXTB	Dn	8 ⇒ 32	Sign extended Destination ⇒ Destination
ILLEGAL	none	none	SSP – 2 ⇒ SSP; vector offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SSP – 2 ⇒ SSP; SR ⇒ (SSP); Illegal instruction vector address ⇒ PC
JMP	<ea>	none	Destination ⇒ PC
JSR	<ea>	none	SP – 4 ⇒ SP; PC ⇒ (SP); destination ⇒ PC
LEA	<ea>, An	32	<ea> ⇒ An
LINK	An, # d	16, 32	SP – 4 ⇒ SP, An ⇒ (SP); SP ⇒ An, SP + d ⇒ SP
LPSTOP[1]	# <data>	16	Data ⇒ SR; interrupt mask ⇒ EBI; STOP
LSL	Dn, Dn # <data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
LSR	Dn, Dn #<data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
MOVE	<ea>, <ea>	8, 16, 32	Source ⇒ Destination
MOVEA	<ea>, An	16, 32 ⇒ 32	Source ⇒ Destination
MOVEA[1]	USP, An An, USP	32 32	USP ⇒ An An ⇒ USP
MOVE from CCR	CCR, <ea>	16	CCR ⇒ Destination
MOVE to CCR	<ea>, CCR	16	Source ⇒ CCR
MOVE from SR[1]	SR, <ea>	16	SR ⇒ Destination
MOVE to SR[1]	<ea>, SR	16	Source ⇒ SR
MOVE USP[1]	USP, An An, USP	32 32	USP ⇒ An An ⇒ USP
MOVEC[1]	Rc, Rn Rn, Rc	32 32	Rc ⇒ Rn Rn ⇒ Rc
MOVEM	list, <ea> <ea>, list	16, 32 16, 32 ⇒ 32	Listed registers ⇒ Destination Source ⇒ Listed registers

Table 20 Instruction Set Summary (Continued)

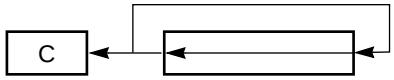
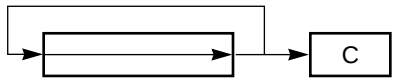
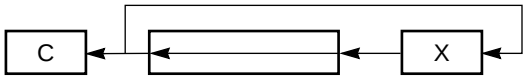
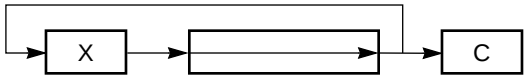
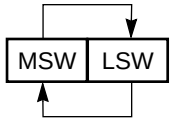
Instruction	Syntax	Operand Size	Operation
MOVEP	Dn, (d16, An) (d16, An), Dn	16, 32	Dn [31 : 24] ⇒ (An + d); Dn [23 : 16] ⇒ (An + d + 2); Dn [15 : 8] ⇒ (An + d + 4); Dn [7 : 0] ⇒ (An + d + 6) (An + d) ⇒ Dn [31 : 24]; (An + d + 2) ⇒ Dn [23 : 16]; (An + d + 4) ⇒ Dn [15 : 8]; (An + d + 6) ⇒ Dn [7 : 0]
MOVEQ	#<data>, Dn	8 ⇒ 32	Immediate data ⇒ Destination
MOVES[1]	Rn, <ea> <ea>, Rn	8, 16, 32	Rn ⇒ Destination using DFC Source using SFC ⇒ Rn
MULS/MULU	<ea>, Dn <ea>, Dl <ea>, Dh : Dl	16 ∗ 16 ⇒ 32 32 ∗ 32 ⇒ 32 32 ∗ 32 ⇒ 64	Source ∗ Destination ⇒ Destination (signed or unsigned)
NBCD	<ea>	8 8	0 – Destination$_{10}$ – X ⇒ Destination
NEG	<ea>	8, 16, 32	0 – Destination ⇒ Destination
NEGX	<ea>	8, 16, 32	0 – Destination – X ⇒ Destination
NOP	none	none	PC + 2 ⇒ PC
NOT	<ea>	8, 16, 32	$\overline{\text{Destination}}$ ⇒ Destination
OR	<ea>, Dn Dn, <ea>	8, 16, 32 8, 16, 32	Source + Destination ⇒ Destination
ORI	#<data>, <ea>	8, 16, 32	Data + Destination ⇒ Destination
ORI to CCR	#<data>, CCR	16	Source + CCR ⇒ SR
ORI to SR[1]	#<data>, SR	16	Source ; SR ⇒ SR
PEA	<ea>	32	SP – 4 ⇒ SP; <ea> ⇒ SP
RESET[1]	none	none	Assert $\overline{\text{RESET}}$ line
ROL	Dn, Dn #<data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
ROR	Dn, Dn #<data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
ROXL	Dn, Dn #<data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
ROXR	Dn, Dn #<data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
RTD	#d	16	(SP) ⇒ PC; SP + 4 + d ⇒ SP
RTE[1]	none	none	(SP) ⇒ SR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP; Restore stack according to format
RTR	none	none	(SP) ⇒ CCR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP
RTS	none	none	(SP) ⇒ PC; SP + 4 ⇒ SP
SBCD	Dn, Dn – (An), – (An)	8 8	Destination10 – Source10 – X ⇒ Destination
Scc	<ea>	8	If condition true, then destination bits are set to 1; else, destination bits are cleared to 0
STOP[1]	#<data>	16	Data ⇒ SR; STOP

Table 20 Instruction Set Summary (Continued)

Instruction	Syntax	Operand Size	Operation
SUB	<ea>, Dn Dn, <ea>	8, 16, 32	Destination – Source ⇒ Destination
SUBA	<ea>, An	16, 32	Destination – Source ⇒ Destination
SUBI	#<data>, <ea>	8, 16, 32	Destination – Data ⇒ Destination
SUBQ	#<data>, <ea>	8, 16, 32	Destination – Data ⇒ Destination
SUBX	Dn, Dn – (An), – (An)	8, 16, 32 8, 16, 32	Destination – Source – X ⇒ Destination
SWAP	Dn	16	
TAS	<ea>	8	Destination Tested Condition Codes bit 7 of Destination
TBLS/TBLU	<ea>, Dn Dym : Dyn, Dn	8, 16, 32	Dyn – Dym ⇒ Temp (Temp ∗ Dn [7 : 0]) ⇒ Temp (Dym ∗ 256) + Temp ⇒ Dn
TBLSN/TBLUN	<ea>, Dn Dym : Dyn, Dn	8, 16, 32	Dyn – Dym ⇒ Temp (Temp ∗ Dn [7 : 0]) / 256 ⇒ Temp Dym + Temp ⇒ Dn
TRAP	#<data>	none	SSP – 2 ⇒ SSP; format/vector offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SR ⇒ (SSP); vector address ⇒ PC
TRAPcc	none #<data>	none 16, 32	If cc true, then TRAP exception
TRAPV	none	none	If V set, then overflow TRAP exception
TST	<ea>	8, 16, 32	Source – 0, to set condition codes
UNLK	An	32	An ⇒ SP; (SP) ⇒ An, SP + 4 ⇒ SP

1. Privileged instruction.

4.7 Background Debugging Mode

The background debugger on the CPU32 is implemented in CPU microcode. The background debugging commands are summarized below.

Table 21 Background Debugging Mode

Command	Mnemonic	Description
Read D/A Register	RDREG/RAREG	Read the selected address or data register and return the results through the serial interface.
Write D/A Register	WDREG/WAREG	The data operand is written to the specified address or data register.
Read System Register	RSREG	The specified system control register is read. All registers that can be read in supervisor mode can be read in background mode.
Write System Register	WSREG	The operand data is written into the specified system control register.
Read Memory Location	READ	Read the sized data at the memory location specified by the long-word address. The source function code register (SFC) determines the address space accessed.
Write Memory Location	WRITE	Write the operand data to the memory location specified by the long-word address. The destination function code (DFC) register determines the address space accessed.
Dump Memory Block	DUMP	Used in conjunction with the READ command to dump large blocks of memory. An initial READ is executed to set up the starting address of the block and retrieve the first result. Subsequent operands are retrieved with the DUMP command.
Fill Memory Block	FILL	Used in conjunction with the WRITE command to fill large blocks of memory. Initially, a WRITE is executed to set up the starting address of the block and supply the first operand. The FILL command writes subsequent operands.
Resume Execution	GO	The pipe is flushed and refilled before resuming instruction execution at the current PC.
Patch User Code	CALL	Current program counter is stacked at the location of the current stack pointer. Instruction execution begins at user patch code.
Reset Peripherals	RST	Asserts RESET for 512 clock cycles. The CPU is not reset by this command. Synonymous with the CPU RESET instruction.
No Operation	NOP	NOP performs no operation and can be used as a null command.

5 Queued Serial Module

The QSM contains two serial interfaces, the queued serial peripheral interface (QSPI) and the serial communication interface (SCI).

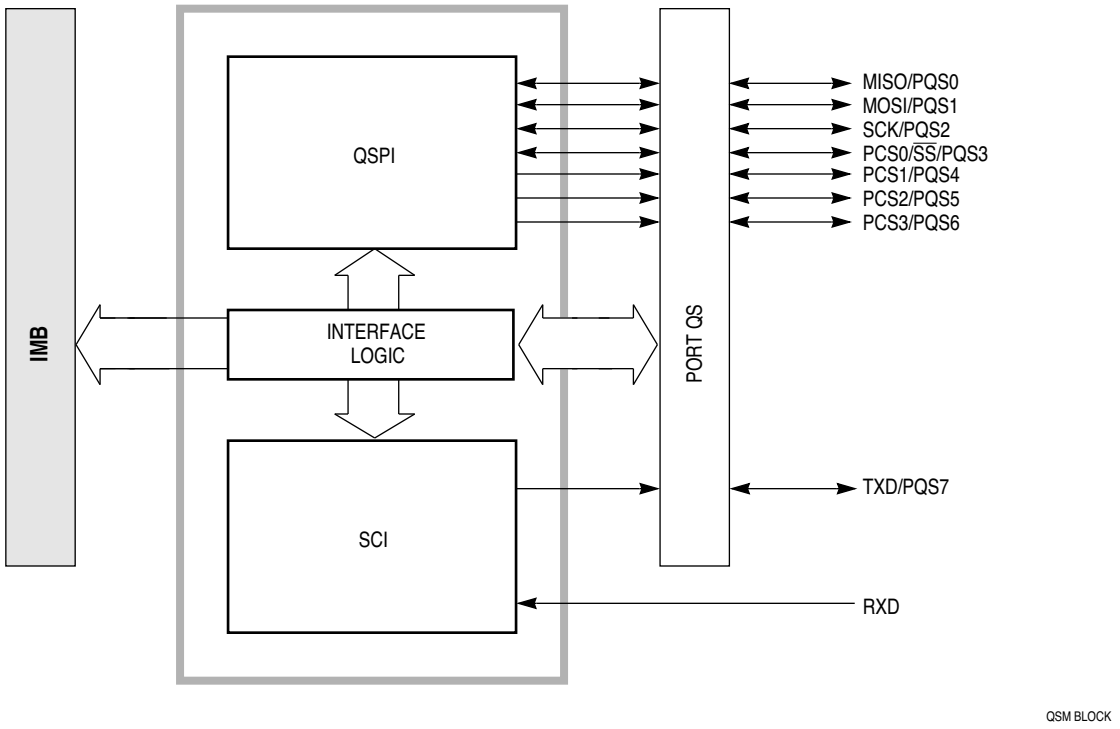


Figure 12 QSM Block Diagram

5.1 Overview

The QSPI provides easy peripheral expansion or interprocessor communication through a full-duplex, synchronous, three-line bus: data in, data out, and a serial clock. Four programmable peripheral chip-select pins provide addressability for up to 16 peripheral devices. A self-contained RAM queue allows up to 16 serial transfers of 8 to 16 bits each, or transmission of a 256-bit data stream without CPU intervention. A special wraparound mode supports continuous sampling of a serial peripheral, with automatic QSPI RAM updating, which makes the interface to A/D converters more efficient.

The SCI provides a standard nonreturn to zero (NRZ) mark/space format. It operates in either full- or half-duplex mode. There are separate transmitter and receiver enable bits and dual data buffers. A modulus-type baud rate generator provides rates from 64 to 524 kbaud with a 16.78-MHz system clock, or 110 to 655 kbaud with a 20.97 MHz system clock. Word length of either 8 or 9 bits is software selectable. Optional parity generation and detection provide either even or odd parity check capability. Advanced error detection circuitry catches glitches of up to 1/16 of a bit time in duration. Wakeup functions allow the CPU to run uninterrupted until meaningful data is available.

An address map of the QSM is shown below. The “Access” column indicates which registers are accessible only at the supervisor privilege level and which can be assigned to either the supervisor or user privilege level, according to the value of the SUPV bit in the QSMCR.

Table 22 QSM Address Map

Access	Address	15 8	7 0
S	$YFFC00	QSM MODULE CONFIGURATION (QSMCR)	
S	$YFFC02	QSM TEST (QTEST)	
S	$YFFC04	QSM INTERRUPT LEVEL (QILR)	QSM INTERRUPT VECTOR (QIVR)
S/U	$YFFC06	NOT USED	
S/U	$YFFC08	SCI CONTROL 0 (SCCR0)	
S/U	$YFFC0A	SCI CONTROL 1 (SCCR1)	
S/U	$YFFC0C	SCI STATUS (SCSR)	
S/U	$YFFC0E	SCI DATA (SCDR)	
S/U	$YFFC10	NOT USED	
S/U	$YFFC12	NOT USED	
S/U	$YFFC14	NOT USED	PQS DATA (PORTQS)
S/U	$YFFC16	PQS PIN ASSIGNMENT (PQSPAR)	PQS DATA DIRECTION (DDRQS)
S/U	$YFFC18	SPI CONTROL 0 (SPCR0)	
S/U	$YFFC1A	SPI CONTROL 1 (SPCR1)	
S/U	$YFFC1C	SPI CONTROL 2 (SPCR2)	
S/U	$YFFC1E	SPI CONTROL 3 (SPCR3)	SPI STATUS (SPSR)
S/U	$YFFC20–$YFFCFF	NOT USED	
S/U	$YFFD00–$YFFD1F	RECEIVE RAM (RR[0:F])	
S/U	$YFFD20–$YFFD3F	TRANSMIT RAM (TR[0:F])	
S/U	$YFFD40–$YFFD4F	COMMAND RAM (CR[0:F])	

Y = M111, where M is the logic state of the module mapping (MM) bit in the SIMCR

5.2 Pin Function

The following table is a summary of the functions of the QSM pins when they are not configured for general-purpose I/O. The QSM data direction register (DDRQS) designates each pin except RXD as an input or output.

	Pin	Mode	Pin Function
QSPI Pins	MISO	Master	Serial Data Input to QSPI
		Slave	Serial Data Output from QSPI
	MOSI	Master	Serial Data Output from QSPI
		Slave	Serial Data Input to QSPI
	SCK	Master	Clock Output from QSPI
		Slave	Clock Input to QSPI
	PCS0/$\overline{SS}$	Master	Input: Assertion Causes Mode Fault Output: Selects Peripherals
		Slave	Input: Selects the QSPI
	PCS[3:1]	Master	Output: Selects Peripherals
		Slave	None
SCI Pins	TXD	Transmit	Serial Data Output from SCI
	RXD	Receive	Serial Data Input to SCI

5.3 QSM Registers

QSM registers are divided into four categories: QSM global registers, QSM pin control registers, QSPI submodule registers, and SCI submodule registers. The QSPI and SCI registers are defined in separate sections below. Writes to unimplemented register bits have no meaning or effect, and reads from unimplemented bits always return a logic zero value.

The module mapping bit of the SIM configuration register (SIMCR) defines the most significant bit (ADDR23) of the address, shown in each register figure as Y (Y = $7 or $F). This bit, concatenated with the rest of the address given, forms the absolute address of each register. Refer to the SIM section of this technical summary for more information about how the state of MM affects the system.

5.3.1 Global Registers

The QSM global registers contain system parameters used by both the QSPI and the SCI submodules. These registers contain the bits and fields used to configure the QSM.

QSMCR — QSM Configuration Register **$YFFC00**

15	14	13	12	11	10	9	8	7	6	5	4	3			0
STOP	FRZ1	FRZ0	0	0	0	0	0	SUPV	0	0	0	IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

The QSMCR contains parameters for the QSM/CPU/intermodule bus (IMB) interface.

STOP — Stop Enable

0 = Normal QSM clock operation
1 = QSM clock operation stopped

STOP places the QSM in a low-power state by disabling the system clock in most parts of the module. The QSMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable. However, writes to RAM or any register are guaranteed to be valid while STOP is asserted. STOP can be negated by the CPU and by reset.

The system software must stop each submodule before asserting STOP to avoid complications at restart and to avoid data corruption. The SCI submodule receiver and transmitter should be disabled, and the operation should be verified for completion before asserting STOP. The QSPI submodule should be stopped by asserting the HALT bit in SPCR3 and by asserting STOP after the HALTA flag is set.

FRZ1 — Freeze 1

0 = Ignore the FREEZE signal on the IMB
1 = Halt the QSPI (on a transfer boundary)

FRZ1 determines what action is taken by the QSPI when the FREEZE signal of the IMB is asserted. FREEZE is asserted whenever the CPU enters the background mode.

FRZ0 — Freeze 0

Reserved

Bits [12:8] — Not Implemented

SUPV — Supervisor/Unrestricted

0 = User access
1 = Supervisor access

SUPV defines the assignable QSM registers as either supervisor-only data space or unrestricted data space.

IARB — Interrupt Arbitration Identification Number

The IARB field is used to arbitrate between simultaneous interrupt requests of the same priority. Each module that can generate interrupt requests must be assigned a unique, non-zero IARB field value. Refer to **3.8 Interrupts** for more information.

QTEST — QSM Test Register **$YFFC02**

QTEST is used during factory testing of the QSM. Accesses to QTEST must be made while the MCU is in test mode.

QILR — QSM Interrupt Levels Register **$YFFC04**

15	14	13 11	10 8	7 0
0	0	ILQSPI	ILSCI	QIVR
RESET:				
0	0	0 0 0	0 0 0	

QILR determines the priority level of interrupts requested by the QSM and the vector used when an interrupt is acknowledged.

ILQSPI — Interrupt Level for QSPI

ILQSPI determines the priority of QSPI interrupts. This field must be given a value between $0 (interrupts disabled) to $7 (highest priority).

ILSCI — Interrupt Level of SCI

ILSCI determines the priority of SCI interrupts. This field must be given a value between $0 (interrupts disabled) to $7 (highest priority).

If ILQSPI and ILSCI are the same nonzero value, and both submodules simultaneously request interrupt service, QSPI has priority.

QIVR — QSM Interrupt Vector Register **$YFFC05**

15 8	7 0
QILR	INTV
RESET:	
	0 0 0 0 1 1 1 1

QIVR determines which two vector numbers in the exception vector table are to be used for QSM interrupts. The seven MSB of a user-defined vector number ($40–$FF) must be written into the INTV field during initialization. The value of INTV0 is supplied by the QSM when an interrupt service request is acknowledged.

During an interrupt-acknowledge cycle, INTV[7:1] are driven on DATA[7:1] IMB lines. DATA0 is negated for an SCI interrupt and asserted for a QSPI interrupt. Writes to INTV0 have no meaning or effect. Reads of INTV0 return a value of one.

At reset, QIVR is initialized to $0F, which corresponds to the uninitialized interrupt vector in the exception table.

5.3.2 Pin Control Registers

The QSM uses nine pins, eight of which form a parallel port (PORTQS) on the MCU. Although these pins are used by the serial subsystems, any pin can alternately be assigned as general-purpose I/O on a pin-by-pin basis.

Pins used for general-purpose I/O must not be assigned to the QSPI by register PQSPAR. To avoid driving incorrect data, the first byte to be output must be written before DDRQS is configured. DDRQS must then be written to determine the direction of data flow and to output the value contained in register PORTQS. Subsequent data for output is written to PORTQS.

PORTQS — Port QS Data Register **$YFFC14**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	PQS7	PQS6	PQS5	PQS4	PQS3	PQS2	PQS1	PQS0
	RESET:							
	0	0	0	0	0	0	0	0

PORTQS latches I/O data. Writes drive pins defined as outputs. Reads return data present on the pins. To avoid driving undefined data, first write a byte to PORTQS, then configure DDRQS.

PQSPAR — PORT QS Pin Assignment Register **$YFFC16**
DDRQS — PORT QS Data Direction Register **$YFFC17**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PQSPA6	PQSPA5	PQSPA4	PQSPA3	0	PQSPA1	PQSPA0	DDQS7	DDQS6	DDQS5	DDQS4	DDQS3	DDQS2	DDQS1	DDQS0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Clearing a bit in the PQSPAR assigns the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI. The PQSPAR does not affect operation of the SCI.

Table 23 QSPAR Pin Assignments

PQSPAR Field	PQSPAR Bit	Pin Function
PQSPA0	0 1	PQS0 MISO
PQSPA1	0 1	PQS1 MOSI
PQSPA2	0 1	PQS2[1] SCK
PQSPA3	0 1	PQS3 PCS0/$\overline{SS}$
PQSPA4	0 1	PQS4 PCS1
PQSPA5	0 1	PQS5 PCS2
PQSPA6	0 1	PQS6 PCS3
PQSPA7	0 1	PQS7[2] TXD

NOTES:
1. PQS2 is a digital I/O pin unless the SPI is enabled (SPE in SPCR1 set), in which case it becomes SPI serial clock SCK.
2. PQS7 is a digital I/O pin unless the SCI transmitter is enabled (TE in SCCR1 = 1), in which case it becomes SCI serial output TXD.

DDRQS determines whether pins are inputs or outputs. Clearing a bit makes the corresponding pin an input; setting a bit makes the pin an output. DDRQS affects both QSPI function and I/O function.

Table 24 Effect of DDRQS on QSM Pin Function

QSM Pin	Mode	DDRQS Bit	Bit State	Pin Function
MISO	Master	DDQ0	0	Serial Data Input to QSPI
			1	Disables Data Input
	Slave		0	Disables Data Output
			1	Serial Data Output from QSPI
MOSI	Master	DDQ1	0	Disables Data Output
			1	Serial Data Output from QSPI
	Slave		0	Serial Data Input to QSPI
			1	Disables Data Input
SCK[1]	Master	DDQ2	0	Disables Clock Output
			1	Clock Output from QSPI
	Slave		0	Clock Input to QSPI
			1	Disables Clock Input
PCS0/$\overline{SS}$	Master	DDQ3	0	Assertion Causes Mode Fault
			1	Chip-Select Output
	Slave		0	QSPI Slave Select Input
			1	Disables Select Input
PCS[3:1]	Master	DDQ[4:6]	0	Disables Chip-Select Output
			1	Chip-Select Output
	Slave		0	Inactive
			1	Inactive
TXD[2]	Transmit	DDQ7	X	Serial Data Output from SCI
RXD	Receive	None	NA	Serial Data Input to SCI

NOTES:

1. PQS2 is a digital I/O pin unless the SPI is enabled (SPE in SPCR1 set), in which case it becomes SPI serial clock SCK.
2. PQS7 is a digital I/O pin unless the SCI transmitter is enabled (TE in SCCR1 = 1), in which case it becomes SCI serial output TXD.

DDRQS determines the direction of the TXD pin only when the SCI transmitter is disabled. When the SCI transmitter is enabled, the TXD pin is an output.

5.4 QSPI Submodule

The QSPI submodule communicates with external devices through a synchronous serial bus. The QSPI is fully compatible with the serial peripheral interface (SPI) systems found on other Motorola products. A block diagram of the QSPI is shown below.

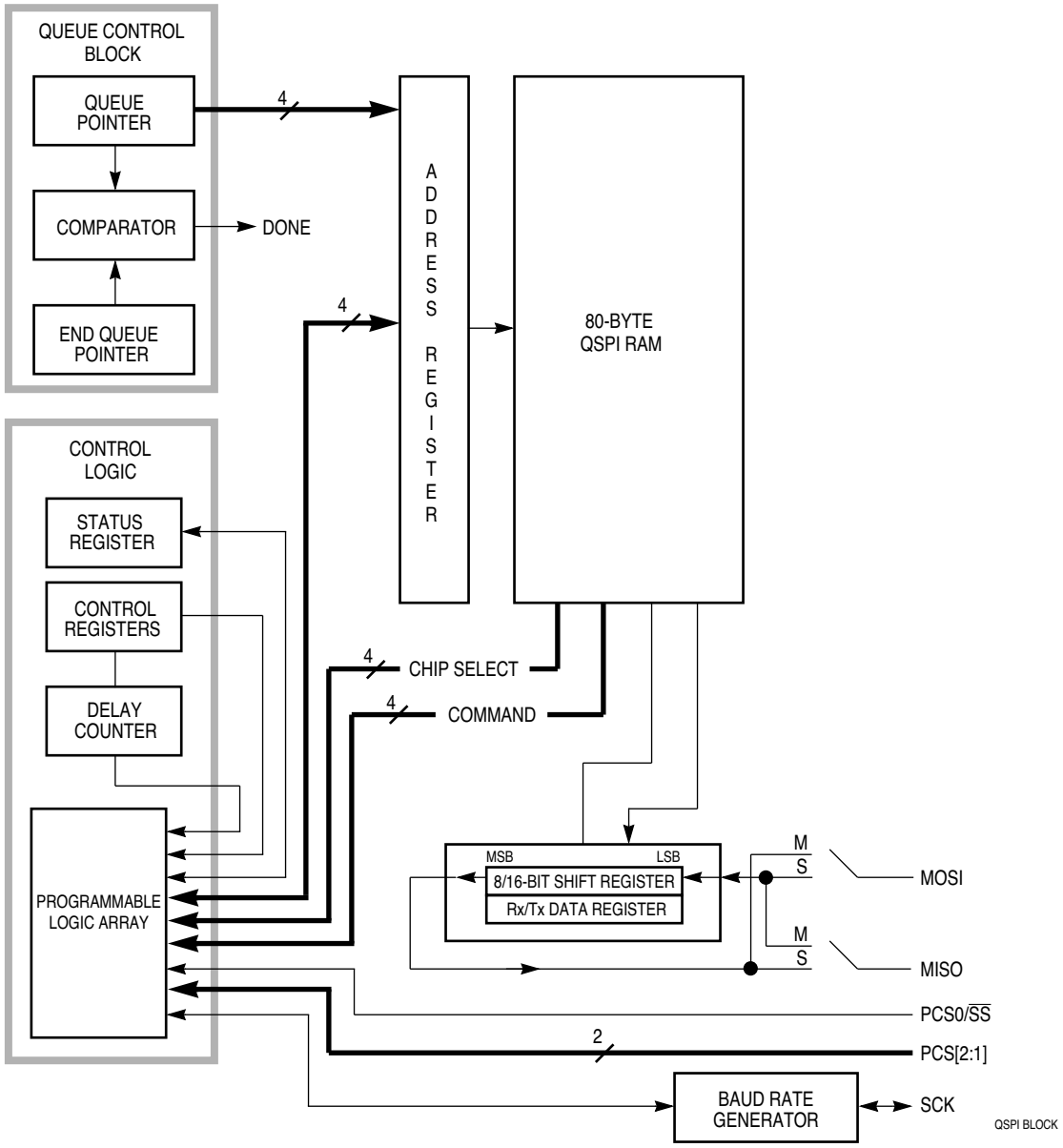


Figure 13 QSPI Block Diagram

5.4.1 QSPI Pins

Seven pins are associated with the QSPI. When not needed for a QSPI application, they can be configured as general-purpose I/O pins. The PCS0/$\overline{SS}$ pin can function as a peripheral chip select output, slave select input, or general-purpose I/O. Refer to the following table for QSPI input and output pins and their functions.

Pin Names	Mnemonics	Mode	Function
Master In Slave Out	MISO	Master Slave	Serial Data Input to QSPI Serial Data Output from QSPI
Master Out Slave In	MOSI	Master Slave	Serial Data Output from QSPI Serial Data Input to QSPI
Serial Clock	SCK	Master Slave	Clock Output from QSPI Clock Input to QSPI
Peripheral Chip Selects	PCS[3:1]	Master	Select Peripherals
Peripheral Chip Select Slave Select	PCS0 $\overline{SS}$	Master Master Slave	Selects Peripheral Causes Mode Fault Initiates Serial Transfer

5.4.2 QSPI Registers

The programmer's model for the QSPI submodule consists of the QSM global and pin control registers, four QSPI control registers, one status register, and the 80-byte QSPI RAM.

The CPU can read and write to registers and RAM. The four control registers must be initialized before the QSPI is enabled to ensure defined operation. SPCR1 should be written last because it contains QSPI enable bit SPE. Asserting this bit starts the QSPI. The QSPI control registers are reset to a defined state and can then be changed by the CPU. Reset values are shown below each register.

Refer to the following memory map of the QSPI.

Address	Name	Usage
$YFFC18	SPCR0	QSPI Control Register 0
$YFFC1A	SPCR1	QSPI Control Register 1
$YFFC1C	SPCR2	QSPI Control Register 2
$YFFC1E	SPCR3	QSPI Control Register 3
$YFFC1F	SPSR	QSPI Status Register
$YFFD00	RAM	QSPI Receive Data (16 Words)
$YFFD20	RAM	QSPI Transmit Data (16 Words)
$YFFD40	RAM	QSPI Command Control (8 Words)

Writing a different value into any control register except SPCR2 while the QSPI is enabled disrupts operation. SPCR2 is buffered to prevent disruption of the current serial transfer. After completion of the current serial transfer, the new SPCR2 values become effective.

Writing the same value into any control register except SPCR2 while the QSPI is enabled has no effect on QSPI operation. Rewriting NEWQP in SPCR2 causes execution to restart at the designated location.

SPCR0 — QSPI Control Register 0 **$YFFC18**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSTR	WOMQ	BITS				CPOL	CPHA	SPBR							
RESET:															
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

SPCR0 contains parameters for configuring the QSPI before it is enabled. The CPU can read and write this register. The QSM has read-only access.

MSTR — Master/Slave Mode Select

0 = QSPI is a slave device and only responds to externally generated serial data.

1 = QSPI is system master and can initiate transmission to external SPI devices.

MSTR configures the QSPI for either master or slave mode operation. This bit is cleared on reset and may only be written by the CPU.

WOMQ — Wired-OR Mode for QSPI Pins

0 = Outputs have normal MOS drivers.

1 = Pins designated for output by DDRQS have open-drain drivers.

WOMQ allows the wired-OR function to be used on QSPI pins, regardless of whether they are used as general-purpose outputs or as QSPI outputs. WOMQ affects the QSPI pins regardless of whether the QSPI is enabled or disabled.

BITS — Bits Per Transfer

In master mode, when BITSE in a command is set, the BITS field determines the number of data bits transferred. When BITSE is cleared, eight bits are transferred. Reserved values default to eight bits. BITSE is not used in slave mode.

The following table shows the number of bits per transfer.

BITS	Bits per Transfer
0000	16
0001	Reserved
0010	Reserved
0011	Reserved
0100	Reserved
0101	Reserved
0110	Reserved
0111	Reserved
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

CPOL — Clock Polarity

0 = The inactive state value of SCK is logic level zero.

1 = The inactive state value of SCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SCK). It is used with CPHA to produce a desired clock/data relationship between master and slave devices.

CPHA — Clock Phase

0 = Data is captured on the leading edge of SCK and changed on the following edge of SCK.

1 = Data is changed on the leading edge of SCK and captured on the following edge of SCK.

CPHA determines which edge of SCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices. CPHA is set at reset.

SPBR — Serial Clock Baud Rate

The QSPI uses a modulus counter to derive SCK baud rate from the MCU system clock. Baud rate is selected by writing a value from 2 to 255 into the SPBR field. The following equation determines the SCK baud rate:

SCK Baud Rate = System Clock/(2SPBR)

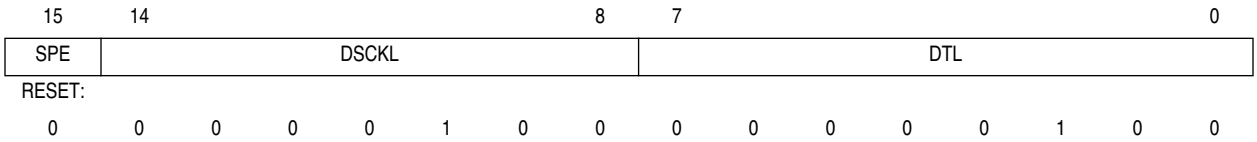
or

SPBR = System Clock/(2SCK)(Baud Rate Desired)

where SPBR equals {2, 3, 4,..., 255}

Giving SPBR a value of zero or one disables the baud rate generator. SCK is disabled and assumes its inactive state value. No serial transfers occur. At reset, baud rate is initialized to one eighth of the system clock frequency.

SPCR1 — QSPI Control Register 1 **$YFFC1A**



SPCR1 contains parameters for configuring the QSPI before it is enabled. The CPU can read and write this register, but the QSM has read access only, except for SPE, which is automatically cleared by the QSPI after completing all serial transfers, or when a mode fault occurs.

SPE — QSPI Enable

0 = QSPI is disabled. QSPI pins can be used for general-purpose I/O.

1 = QSPI is enabled. Pins allocated by PQSPAR are controlled by the QSPI.

DSCKL — Delay before SCK

When the DSCK bit in command RAM is set, this field determines the length of delay from PCS valid to SCK transition. PCS can be any of the four peripheral chip-select pins. The following equation determines the actual delay before SCK:

PCS to SCK Delay = [DSCKL/System Clock]

where DSCKL equals {1, 2, 3,..., 127}.

When the DSCK value of a queue entry equals zero, then DSCKL is not used. Instead, the PCS valid-to-SCK transition is one-half SCK period.

DTL — Length of Delay after Transfer

When the DT bit in command RAM is set, this field determines the length of delay after serial transfer. The following equation is used to calculate the delay:

Delay after Transfer = [(32DTL)/System Clock]

where DTL equals {1, 2, 3,..., 255}.

A zero value for DTL causes a delay-after-transfer value of 8192/System Clock.

If DT equals zero, a standard delay is inserted.

Standard Delay after Transfer = [17/System Clock]

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion.

SPCR2 — QSPI Control Register 2 **$YFFC1C**

15	14	13	12	11 … 8	7	6	5	4	3 … 0
SPIFIE	WREN	WRTO	0	ENDQP	0	0	0	0	NEWQP
RESET:									
0	0	0	0	0 0 0 0	0	0	0	0	0 0 0 0

SPCR2 contains QSPI configuration parameters. The CPU can read and write this register; the QSM has read access only. Writes to SPCR2 are buffered. A write to SPCR2 that changes a bit value while the QSPI is operating is ineffective on the current serial transfer, but becomes effective on the next serial transfer. Reads of SPCR2 return the current value of the register, not of the buffer.

SPIFIE — SPI Finished Interrupt Enable
0 = QSPI interrupts disabled
1 = QSPI interrupts enabled
SPIFIE enables the QSPI to generate a CPU interrupt upon assertion of the status flag SPIF.

WREN — Wrap Enable
0 = Wraparound mode disabled
1 = Wraparound mode enabled
WREN enables or disables wraparound mode.

WRTO — Wrap To
When wraparound mode is enabled, after the end of queue has been reached, WRTO determines which address the QSPI executes.

Bit 12 — Not Implemented

ENDQP — Ending Queue Pointer
This field contains the last QSPI queue address.

Bits [7:4] — Not Implemented

NEWQP — New Queue Pointer Value
This field contains the first QSPI queue address.

SPCR3 — QSPI Control Register 3 **$YFFC1E**

15	14	13	12	11	10	9	8	7 … 0
0	0	0	0	0	LOOPQ	HMIE	HALT	SPSR
RESET:								
0	0	0	0	0	0	0	0	

SPCR3 contains QSPI configuration parameters. The CPU can read and write SPCR3, but the QSM has read-only access.

Bits [15:11] — Not Implemented

LOOPQ — QSPI Loop Mode
0 = Feedback path disabled
1 = Feedback path enabled
LOOPQ controls feedback on the data serializer for testing.

HMIE — HALTA and MODF Interrupt Enable
0 = HALTA and MODF interrupts disabled
1 = HALTA and MODF interrupts enabled
HMIE controls CPU interrupts caused by the HALTA status flag or the MODF status flag in SPSR.

HALT — Halt
0 = Halt not enabled
1 = Halt enabled
When HALT is asserted, the QSPI stops on a queue boundary. It is in a defined state from which it can later be restarted.

SPSR — QSPI Status Register **$YFFC1F**

15 … 8	7	6	5	4	3 … 0
SPCR3	SPIF	MODF	HALTA	0	CPTQP
RESET:	0	0	0	0	0 0 0 0

SPSR contains QSPI status information. Only the QSPI can assert the bits in this register. The CPU reads this register to obtain status information and writes it to clear status flags.

SPIF — QSPI Finished Flag
0 = QSPI not finished
1 = QSPI finished
SPIF is set after execution of the command at the address in ENDQP.

MODF — Mode Fault Flag
0 = Normal operation
1 = Another SPI node requested to become the network SPI master while the QSPI was enabled in master mode ($\overline{SS}$ input taken low).
The QSPI asserts MODF when the QSPI is the serial master (MSTR = 1) and the $\overline{SS}$ input pin is negated by an external driver.

HALTA — Halt Acknowledge Flag
0 = QSPI not halted
1 = QSPI halted
HALTA is asserted when the QSPI halts in response to CPU assertion of HALT.

Bit 4 — Not Implemented

CPTQP — Completed Queue Pointer
CPTQP points to the last command executed. It is updated when the current command is complete. When the first command in a queue is executing, CPTQP contains either the reset value ($0) or a pointer to the last command completed in the previous queue.

5.4.3 QSPI RAM

The QSPI contains an 80-byte block of dual-access static RAM that is used by both the QSPI and the CPU. The RAM is divided into three segments: receive data, transmit data, and command control data. Receive data is information received from a serial device external to the MCU. Transmit data is information stored by the CPU for transmission to an external peripheral. Command control data is used to perform the transfer.

Refer to the following illustration of the organization of the RAM.

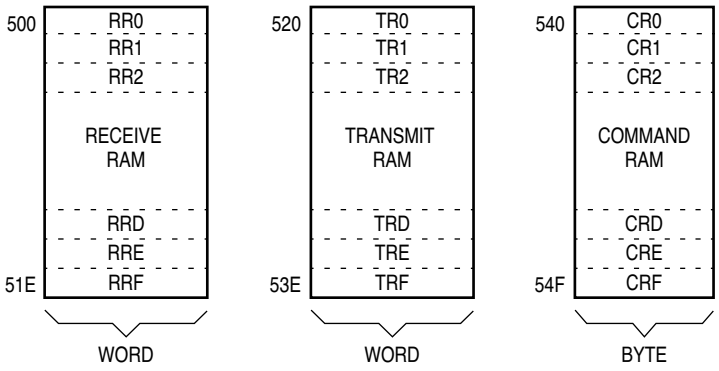


Figure 14 QSPI RAM

Once the CPU has set up the queue of QSPI commands and enabled the QSPI, the QSPI can operate independently of the CPU. The QSPI executes all of the commands in its queue, sets a flag indicating that it is finished, and then either interrupts the CPU or waits for CPU intervention. It is possible to execute a queue of commands repeatedly without CPU intervention.

RR[0:F] — Receive Data RAM **$YFFD00**

Data received by the QSPI is stored in this segment. The CPU reads this segment to retrieve data from the QSPI. Data stored in receive RAM is right-justified. Unused bits in a receive queue entry are set to zero by the QSPI upon completion of the individual queue entry. The CPU can access the data using byte, word, or long-word addressing.

The CPTQP value in SPSR shows which queue entries have been executed. The CPU uses this information to determine which locations in receive RAM contain valid data before reading them.

TR[0:F] — Transmit Data RAM **$YFFD20**

Data that is to be transmitted by the QSPI is stored in this segment. The CPU usually writes one word of data into this segment for each queue command to be executed.

Information to be transmitted must be written to transmit data RAM in a right-justified format. The QSPI cannot modify information in the transmit data RAM. The QSPI copies the information to its data serializer for transmission. Information remains in transmit RAM until overwritten.

CR[0:F] — Command RAM **$YFFD40**

7	6	5	4	3	2	1	0
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*
–	–	–	–	–	–	–	–
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*

*The PCS0 bit represents the dual-function PCS0/$\overline{SS}$.

Command RAM is used by the QSPI when in master mode. The CPU writes one byte of control information to this segment for each QSPI command to be executed. The QSPI cannot modify information in command RAM.

Command RAM consists of 16 bytes. Each byte is divided into two fields. The peripheral chip-select field enables peripherals for transfer. The command control field provides transfer options.

A maximum of 16 commands can be in the queue. Queue execution by the QSPI proceeds from the address in NEWQP through the address in ENDQP. (Both of these fields are in SPCR2.)

CONT — Continue

0 = Control of chip selects returned to PORTQS after transfer is complete.
1 = Peripheral chip selects remain asserted after transfer is complete.

BITSE — Bits per Transfer Enable

0 = 8 bits
1 = Number of bits set in BITS field of SPCR0

DT — Delay after Transfer

The QSPI provides a variable delay at the end of serial transfer to facilitate the interface with peripherals that have a latency requirement. The delay between transfers is determined by the SPCR1 DTL field.

DSCK — PCS to SCK Delay

0 = PCS valid to SCK transition is one-half SCK.
1 = SPCR1 DSCKL field specifies delay from PCS valid to SCK.

PCS[3:0] — Peripheral Chip Select

Use peripheral chip-select bits to select an external device for serial data transfer. More than one peripheral chip select can be activated at a time, and more than one peripheral chip can be connected to each PCS pin, provided that proper fanout is observed.

$\overline{SS}$ — Slave Mode Select

Initiates slave mode serial transfer. If $\overline{SS}$ is taken low when the QSPI is in master mode, a mode fault will be generated.

5.4.4 Operating Modes

The QSPI operates in either master or slave mode. Master mode is used when the MCU originates data transfers. Slave mode is used when an external device initiates serial transfers to the MCU through the QSPI. Switching between the modes is controlled by MSTR in SPCR0. Before entering either mode, appropriate QSM and QSPI registers must be properly initialized.

In master mode, the QSPI executes a queue of commands defined by control bits in each command RAM queue entry. Chip-select pins are activated, data is transmitted from transmit RAM and received into receive RAM.

In slave mode, operation proceeds in response to $\overline{SS}$ pin activation by an external bus master. Operation is similar to master mode, but no peripheral chip selects are generated, and the number of bits transferred is controlled in a different manner. When the QSPI is selected, it automatically executes the next queue transfer to exchange data with the external device correctly.

Although the QSPI inherently supports multimaster operation, no special arbitration mechanism is provided. A mode fault flag (MODF) indicates a request for SPI master arbitration. System software must provide arbitration. Note that unlike previous SPI systems, MSTR is not cleared by a mode fault being set, nor are the QSPI pin output drivers disabled. The QSPI and associated output drivers must be disabled by clearing SPE in SPCR1.

5.5 SCI Submodule

The SCI submodule is used to communicate with external devices through an asynchronous serial bus. The SCI is fully compatible with the SCI systems found on other Motorola MCUs, such as the M68HC11 and M68HC05 Families.

5.5.1 SCI Pins

There are two unidirectional pins associated with the SCI. The SCI controls the transmit data (TXD) pin when enabled, whereas the receive data (RXD) pin remains a dedicated input pin to the SCI. TXD is available as a general-purpose I/O pin when the SCI transmitter is disabled. When used for I/O, TXD can be configured either as input or output, as determined by QSM register DDRQS.

The following table shows SCI pins and their functions.

Pin Names	Mnemonics	Mode	Function
Receive Data	RXD	Receiver Disabled Receiver Enabled	Not Used Serial Data Input to SCI
Transmit Data	TXD	Transmitter Disabled Transmitter Enabled	General-Purpose I/O Serial Data Output from SCI

5.5.2 SCI Registers

The SCI programming model includes QSM global and pin control registers, and four SCI registers. There are two SCI control registers, one status register, and one data register. All registers can be read or written at any time by the CPU.

Changing the value of SCI control bits during a transfer operation may disrupt operation. Before changing register values, allow the transmitter to complete the current transfer, then disable the receiver and transmitter. Status flags in the SCSR may be cleared at any time.

SCCR0 — SCI Control Register 0 **$YFFC08**

15	14	13	12												0
0	0	0	SCBR												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

SCCR0 contains a baud rate selection parameter. Baud rate must be set before the SCI is enabled. The CPU can read and write this register at any time.

Bits [15:13] — Not Implemented

SCBR — Baud Rate

SCI baud rate is programmed by writing a 13-bit value to BR. The baud rate is derived from the MCU system clock by a modulus counter.

The SCI receiver operates asynchronously. An internal clock is necessary to synchronize with an incoming data stream. The SCI baud rate generator produces a receiver sampling clock with a frequency 16 times that of the expected baud rate of the incoming data. The SCI determines the position of bit boundaries from transitions within the received waveform, and adjusts sampling points to the proper positions within the bit period. Receiver sampling rate is always 16 times the frequency of the SCI baud rate, which is calculated as follows:

$$\text{SCI Baud Rate} = \text{System Clock}/(32\text{SCBR})$$

or

$$\text{SCBR} = \text{System Clock}(32\text{SCK})(\text{Baud Rate Desired})$$

where SCBR is in the range {1, 2, 3, ..., 8191}

Writing a value of zero to SCBR disables the baud rate generator.

The following table lists the SCBR settings for standard and maximum baud rates using 16.78-MHz and 20.97-MHz system clocks.

Table 25 SCI Baud Rates

Nominal Baud Rate	Actual Rate with 16.78-MHz Clock	SCBR Value	Actual Rate with 20.97-MHz Clock	SCBR Value
64*	64.0	$1FFF	—	—
110	110.0	$129E	110.0	$1745
300	299.9	$06D4	300.1	$0888
600	599.9	$036A	600.1	$0444
1200	1199.7	$0165	1200.3	$0222
2400	2405.0	$00DA	2400.6	$0111
4800	4810.0	$006D	4783.6	$0089
9600	9532.5	$0037	9637.6	$0044
19200	19418.1	$0016	19275.3	$0022
38400	37449.1	$000E	38550.6	$0011
76800	74898.3	$0007	72817.8	$0009
Maximum Rate	524288.0	$0001	655360.0	$0001

*A rate of 64 baud is not available with a 20.97-MHz system clock. To achieve this rate, the SYNCR can be programmed to generate a lower system clock rate.

SCCR1 — SCI Control Register 1 **$YFFC0A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	SCBR												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SCCR1 contains SCI configuration parameters. The CPU can read and write this register at any time. The SCI can modify RWU in some circumstances. In general, interrupts enabled by these control bits are cleared by reading SCSR, then reading (receiver status bits) or writing (transmitter status bits) SCDR.

Bit 15 — Not Implemented

LOOPS — Loop Mode

0 = Normal SCI operation, no looping, feedback path disabled
1 = Test SCI operation, looping, feedback path enabled

LOOPS controls a feedback path on the data serial shifter. When loop mode is enabled, SCI transmitter output is fed back into the receive serial shifter. TXD is asserted (idle line). Both transmitter and receiver must be enabled before entering loop mode.

WOMS — Wired-OR Mode for SCI Pins

0 = If configured as an output, TXD is a normal CMOS output.
1 = If configured as an output, TXD is an open-drain output.

WOMS determines whether the TXD pin is an open-drain output or a normal CMOS output. This bit is used only when TXD is an output. If TXD is used as a general-purpose input pin, WOMS has no effect.

ILT — Idle-Line Detect Type

0 = Short idle-line detect (start count on first one)
1 = Long idle-line detect (start count on first one after stop bit(s))

PT — Parity Type
0 = Even parity
1 = Odd parity
When parity is enabled, PT determines whether parity is even or odd for both the receiver and the transmitter.

PE — Parity Enable
0 = SCI parity disabled
1 = SCI parity enabled
PE determines whether parity is enabled or disabled for both the receiver and the transmitter. If the received parity bit is not correct, the SCI sets the PF error flag in SCSR.
When PE is set, the most significant bit (MSB) of the data field is used for the parity function, which results in either seven or eight bits of user data, depending on the condition of M bit. The following table lists the available choices.

M	PE	Result
0	0	8 Data Bits
0	1	7 Data Bits, 1 Parity Bit
1	0	9 Data Bits
1	1	8 Data Bits, 1 Parity Bit

M — Mode Select
0 = SCI frame: 1 start bit, 8 data bits, 1 stop bit (10 bits total)
1 = SCI frame: 1 start bit, 9 data bits, 1 stop bit (11 bits total)

WAKE — Wakeup by Address Mark
0 = SCI receiver awakened by idle-line detection
1 = SCI receiver awakened by address mark (last bit set)

TIE — Transmit Interrupt Enable
0 = SCI TDRE interrupts inhibited
1 = SCI TDRE interrupts enabled

TCIE — Transmit Complete Interrupt Enable
0 = SCI TC interrupts inhibited
1 = SCI TC interrupts enabled

RIE — Receiver Interrupt Enable
0 = SCI RDRF interrupt inhibited
1 = SCI RDRF interrupt enabled

ILIE — Idle-Line Interrupt Enable
0 = SCI IDLE interrupts inhibited
1 = SCI IDLE interrupts enabled

TE — Transmitter Enable
0 = SCI transmitter disabled (TXD pin may be used as I/O)
1 = SCI transmitter enabled (TXD pin dedicated to SCI transmitter)
The transmitter retains control of the TXD pin until completion of any character transfer in progress when TE is cleared.

RE — Receiver Enable
0 = SCI receiver disabled (status bits inhibited)
1 = SCI receiver enabled

RWU — Receiver Wakeup

0 = Normal receiver operation (received data recognized)

1 = Wakeup mode enabled (received data ignored until awakened)

Setting RWU enables the wakeup function, which allows the SCI to ignore received data until awakened by either an idle line or address mark (as determined by WAKE). When in wakeup mode, the receiver status flags are not set, and interrupts are inhibited. This bit is cleared automatically (returned to normal mode) when the receiver is awakened.

SBK — Send Break

0 = Normal operation

1 = Break frame(s) transmitted after completion of current frame

SBK provides the ability to transmit a break code from the SCI. If the SCI is transmitting when SBK is set, it will transmit continuous frames of zeros after it completes the current frame, until SBK is cleared. If SBK is toggled (one to zero in less than one frame interval), the transmitter sends only one or two break frames before reverting to idle line or beginning to send data.

SCSR — SCI Status Register **$YFFC0C**

15 … 9	8	7	6	5	4	3	2	1	0
NOT USED	TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF
RESET:	1	1	0	0	0	0	0	0	0

SCSR contains flags that show SCI operational conditions. These flags can be cleared either by hardware or by a special acknowledgment sequence. The sequence consists of SCSR read with flags set, followed by SCDR read (write in the case of TDRE and TC). A long-word read can consecutively access both SCSR and SCDR. This action clears receive status flag bits that were set at the time of the read, but does not clear TDRE or TC flags.

If an internal SCI signal for setting a status bit comes after the CPU has read the asserted status bits, but before the CPU has written or read register SCDR, the newly set status bit is not cleared. SCSR must be read again with the bit set. Also, SCDR must be written or read before the status bit is cleared.

Reading either byte of SCSR causes all 16 bits to be accessed. Any status bit already set in either byte will be cleared on a subsequent read or write of register SCDR.

TDRE — Transmit Data Register Empty Flag

0 = Register TDR still contains data to be sent to the transmit serial shifter.

1 = A new character can now be written to register TDR.

TDRE is set when the byte in register TDR is transferred to the transmit serial shifter. If TDRE is zero, transfer has not occurred and a write to TDR will overwrite the previous value. New data is not transmitted if TDR is written without first clearing TDRE.

TC — Transmit Complete Flag

0 = SCI transmitter is busy

1 = SCI transmitter is idle

TC is set when the transmitter finishes shifting out all data, queued preambles (mark/idle line), or queued breaks (logic zero). The interrupt can be cleared by reading SCSR when TC is set and then by writing the transmit data register (TDR) of SCDR.

RDRF — Receive Data Register Full Flag

0 = Register RDR is empty or contains previously read data.

1 = Register RDR contains new data.

RDRF is set when the content of the receive serial shifter is transferred to the RDR. If one or more errors are detected in the received word, flag(s) NF, FE, and/or PF are set within the same clock cycle.

RAF — Receiver Active Flag

0 = SCI receiver is idle

1 = SCI receiver is busy

RAF indicates whether the SCI receiver is busy. It is set when the receiver detects a possible start bit and is cleared when the chosen type of idle line is detected. RAF can be used to reduce collisions in systems with multiple masters.

IDLE — Idle-Line Detected Flag

0 = SCI receiver did not detect an idle-line condition.

1 = SCI receiver detected an idle-line condition.

IDLE is disabled when RWU in SCCR1 is set. IDLE is set when the SCI receiver detects the idle-line condition specified by ILT in SCCR1. If cleared, IDLE will not set again until after RDRF is set. RDRF is set when a break is received, so that a subsequent idle line can be detected.

OR — Overrun Error Flag

0 = RDRF is cleared before new data arrives.

1 = RDRF is not cleared before new data arrives.

OR is set when a new byte is ready to be transferred from the receive serial shifter to the RDR, and RDRF is still set. Data transfer is inhibited until OR is cleared. Previous data in RDR remains valid, but data received during overrun condition (including the byte that set OR) is lost.

NF — Noise Error Flag

0 = No noise detected on the received data

1 = Noise occurred on the received data

NF is set when the SCI receiver detects noise on a valid start bit, on any data bit, or on a stop bit. It is not set by noise on the idle line or on invalid start bits. Each bit is sampled three times. If none of the three samples are the same logic level, the majority value is used for the received data value, and NF is set. NF is not set until an entire frame is received and RDRF is set.

FE — Framing Error Flag

0 = No framing error on the received data.

1 = Framing error or break occurred on the received data.

FE is set when the SCI receiver detects a zero where a stop bit was to have occurred. FE is not set until the entire frame is received and RDRF is set. A break can also cause FE to be set. It is possible to miss a framing error if RXD happens to be at logic level one at the time the stop bit is expected.

PF — Parity Error Flag

0 = No parity error on the received data

1 = Parity error occurred on the received data

PF is set when the SCI receiver detects a parity error. PF is not set until the entire frame is received and RDRF is set.

SCDR — SCI Data Register **$YFFC0E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:															
0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U

SCDR contains two data registers at the same address. Receive data register (RDR) is a read-only register that contains data received by the SCI. The data comes into the receive serial shifter and is transferred to RDR. Transmit data register (TDR) is a write-only register that contains data to be transmitted. The data is first written to TDR, then transferred to the transmit serial shifter, where additional format bits are added before transmission. R[7:0]/T[7:0] contain either the first eight data bits received when SCDR is read, or the first eight data bits to be transmitted when SCDR is written. R8/T8 are used when the SCI is configured for 9-bit operation. When it is configured for 8-bit operation, they have no meaning or effect.

6 General-Purpose Timer Module

The 11-channel general-purpose timer (GPT) is used in systems where a moderate level of CPU control is required. The GPT consists of a capture/compare unit, a pulse accumulator, and two pulse-width modulators. A bus interface unit connects the GPT to the intermodule bus.

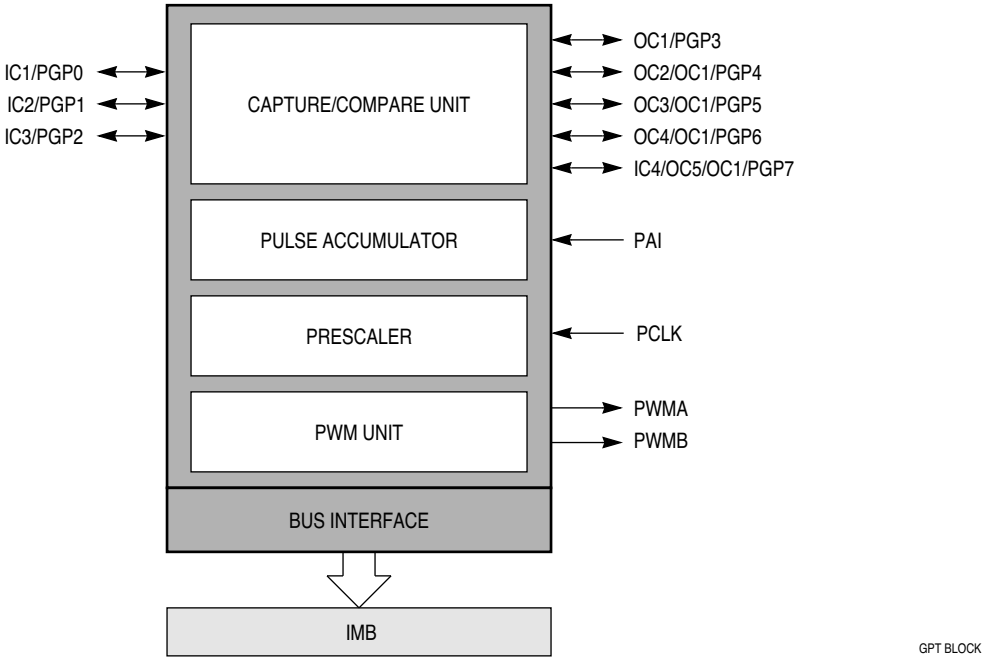


Figure 15 GPT Block Diagram

6.1 Overview

The capture/compare unit features three input capture channels, four output compare channels, and one channel that can be selected as an input capture or output compare channel. These channels share a 16-bit free-running counter (TCNT) which derives its clock from a nine-stage prescaler or from the external clock input signal, PCLK.

Pulse accumulator channel logic includes an 8-bit counter; the pulse accumulator can operate in either event counting mode or gated time accumulation mode.

Pulse-width modulator outputs are periodic waveforms whose duty cycles can be independently selected and modified by user software. The PWM circuits share a 16-bit free-running counter that can be clocked by the same nine-stage prescaler used by the capture/compare unit or by the PCLK input.

All GPT pins can also be used for general-purpose input/output. The input capture and output compare pins form a bidirectional 8-bit parallel port (PORTGP). PWM pins are outputs only. PAI and PCLK pins are inputs only.

GPT input capture/output compare pins are bidirectional and can be used to form an 8-bit parallel port. The pulse-width modulator outputs can be used as general-purpose outputs. The PAI and PCLK inputs can be used as general-purpose inputs.

The GPT control register address map is shown below. The “Access” column in the GPT address map indicates which registers are accessible only at the supervisor privilege level and which can be assigned to either the supervisor or user privilege level, according to the value of the SUPV bit in the GPTMCR.

Table 26 GPT Address Map

Access	Address	15 8	7 0
S	$YFF900	GPT MODULE CONFIGURATION (GPTMCR)	
S	$YFF902	(RESERVED FOR TEST)	
S	$YFF904	INTERRUPT CONFIGURATION (ICR)	
U	$YFF906	PGP DATA DIRECTION (DDRGP)	PGP DATA (PORTGP)
U	$YFF908	OC1 ACTION MASK (OC1M)	OC1 ACTION DATA (OC1D)
U	$YFF90A	TIMER COUNTER (TCNT)	
U	$YFF90C	PA CONTROL (PACTL)	PA COUNTER (PACNT)
U	$YFF90E	INPUT CAPTURE 1 (TIC1)	
U	$YFF910	INPUT CAPTURE 2 (TIC2)	
U	$YFF912	INPUT CAPTURE 3 (TIC3)	
U	$YFF914	OUTPUT COMPARE 1 (TOC1)	
U	$YFF916	OUTPUT COMPARE 2 (TOC2)	
U	$YFF918	OUTPUT COMPARE 3 (TOC3)	
U	$YFF91A	OUTPUT COMPARE 4 (TOC4)	
U	$YFF91C	INPUT CAPTURE 4/OUTPUT COMPARE 5 (TI4/O5)	
U	$YFF91E	TIMER CONTROL 1 (TCTL1)	TIMER CONTROL 2 (TCTL2)
U	$YFF920	TIMER MASK 1 (TMSK1)	TIMER MASK 2 (TMSK2)
U	$YFF922	TIMER FLAG 1 (TFLG1)	TIMER FLAG 2 (TFLG2)
U	$YFF924	FORCE COMPARE (CFORC)	PWM CONTROL C (PWMC)
U	$YFF926	PWM CONTROL A (PWMA)	PWM CONTROL B (PWMB)
U	$YFF928	PWM COUNT (PWMCNT)	
U	$YFF92A	PWMA BUFFER (PWMBUFA)	PWMB BUFFER (PWMBUFB)
U	$YFF92C	GPT PRESCALER (PRESCL)	
	$YFF92E–$YFF93F	NOT USED	

Y = M111, where M is the logic state of the modmap (MM) bit in SIMCR.

6.2 Capture/Compare Unit

The capture/compare unit features three input capture channels, four output compare channels, and one input capture/output compare channel (function selected by control register). These channels share a 16-bit free-running counter (TCNT), which derives its clock from seven stages of a 9-stage prescaler or from external clock input PCLK. This section, which is similar to the timer found on the MC68HC11F1, also contains one pulse accumulator channel. The pulse accumulator logic includes its own 8-bit counter and can operate in either event counting mode or gated time accumulation mode. Refer to the following block diagrams of the GPT timer and prescaler.

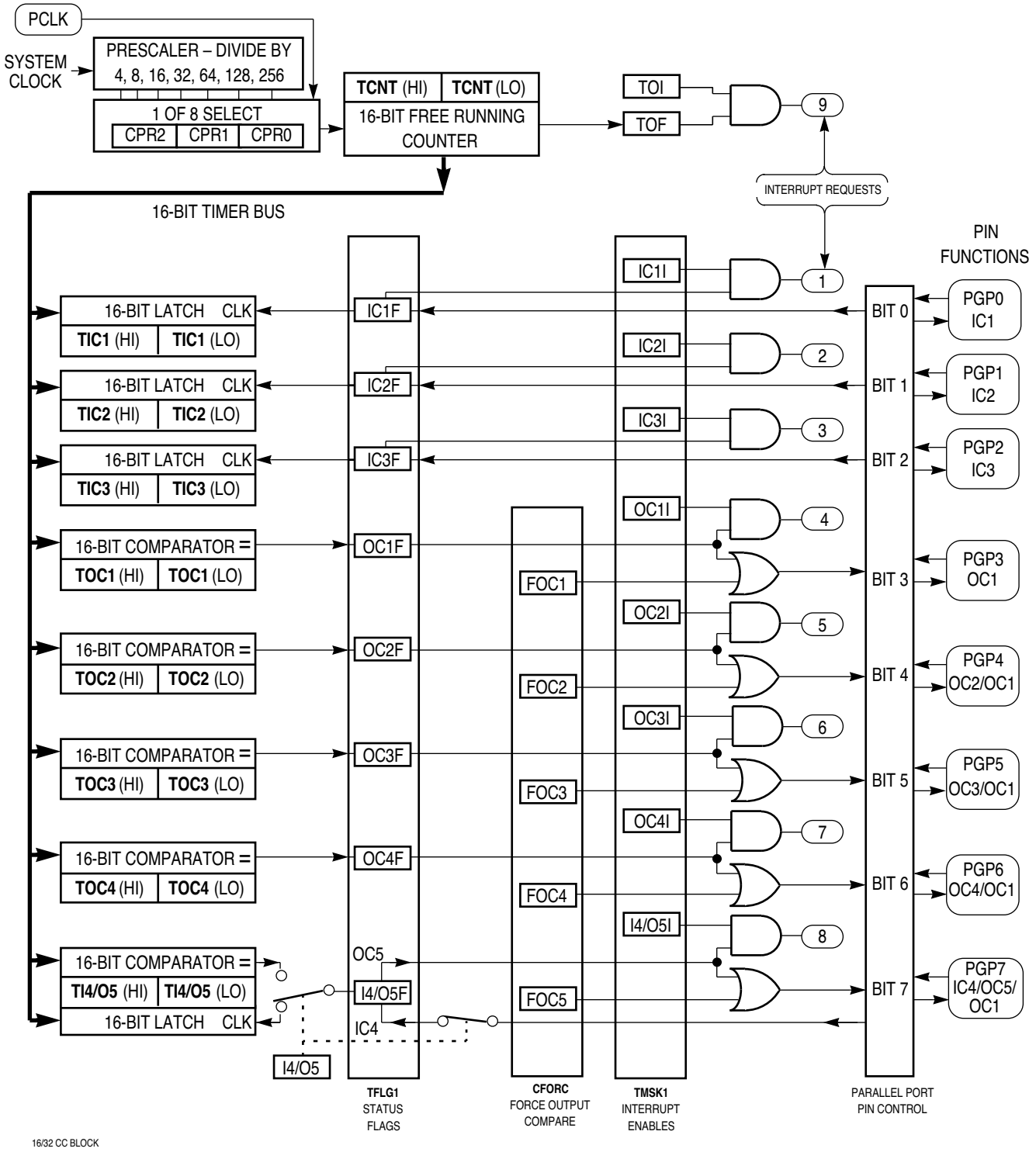


Figure 16 GPT Timer Block Diagram

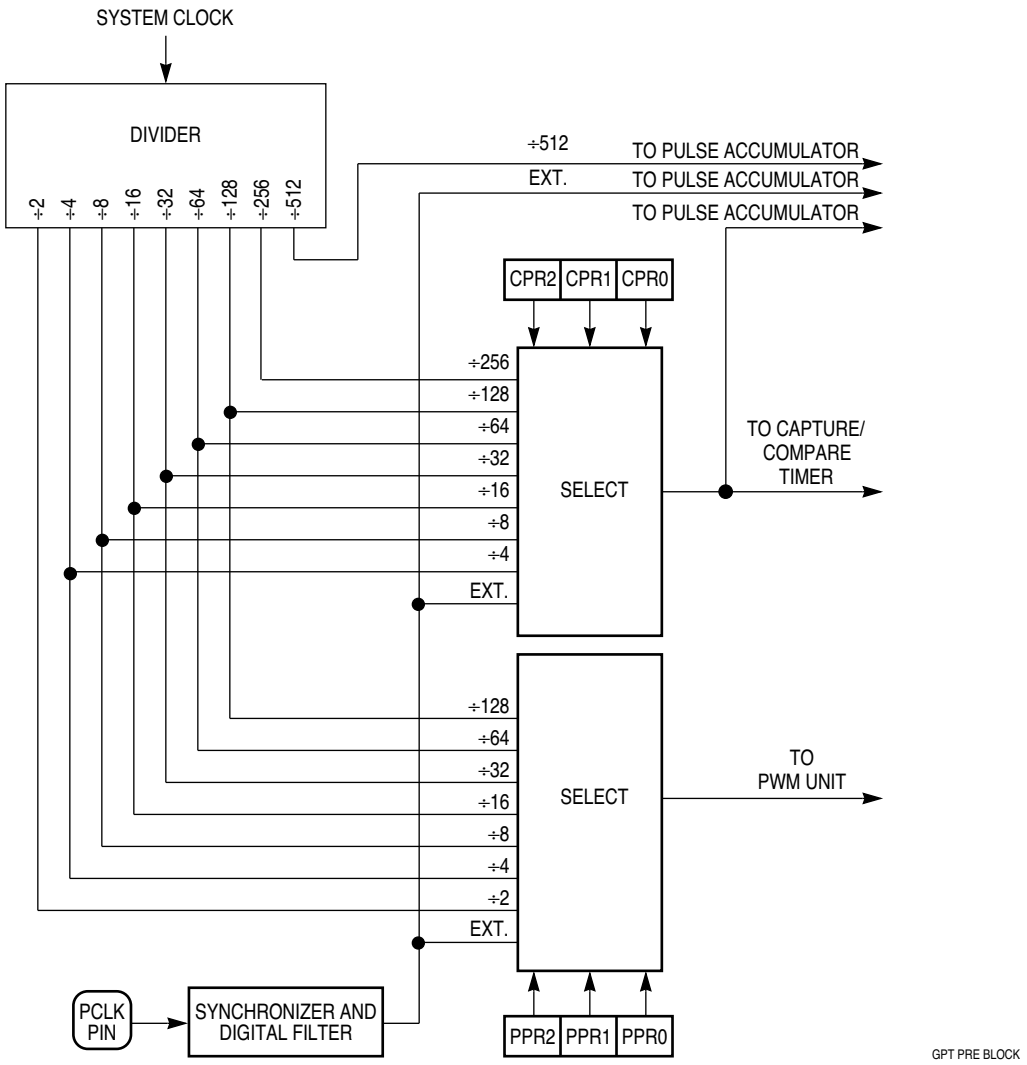


Figure 17 Prescaler Block Diagram

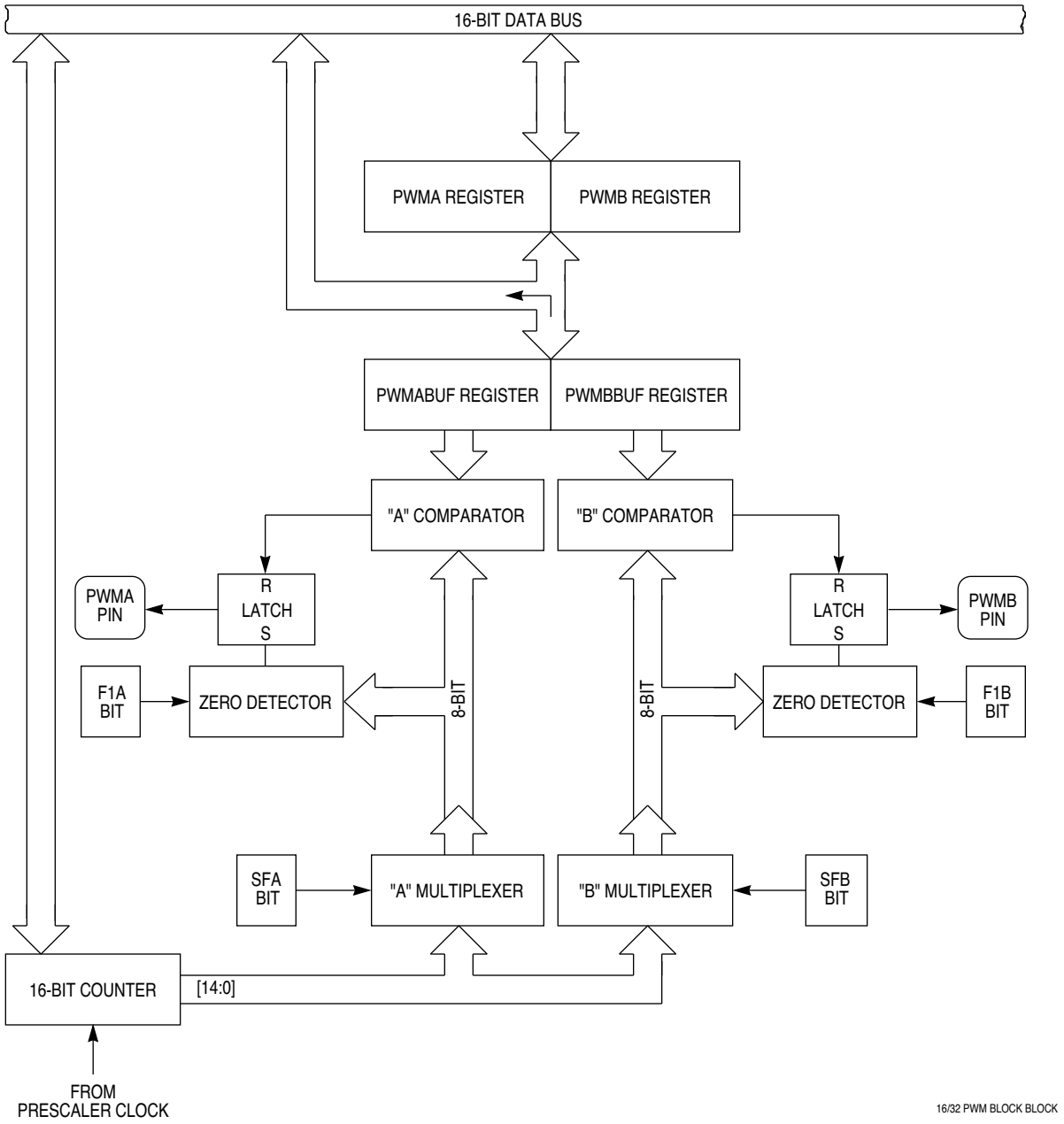


Figure 18 PWM Unit Block Diagram

6.3 Pulse-Width Modulator

The pulse-width modulation submodule has two output pins. The outputs are periodic waveforms controlled by a single frequency whose duty cycles can be independently selected and modified by user software. Each PWM can be independently programmed to run in fast or slow mode. The PWM unit has its own 16-bit free-running counter, which is clocked by an output of the nine-stage prescaler (the same prescaler used by the capture/compare unit) or by the clock input pin, PCLK.

6.4 GPT Registers

GPTMCR — GPT Module Configuration Register **$YFF900**

15	14	13	12	11	10	9	8	7	6	5	4	3			0
STOP	FRZ1	FRZ0	STOPP	INCP	0	0	0	SUPV	0	0	0	IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

The GPTMCR contains parameters for configuring the GPT.

STOP — Stop Clocks
- 0 = Internal clocks not shut down
- 1 = Internal clocks shut down

FRZ1 — Not implemented at this time

FRZ0 — FREEZE Response
- 0 = Ignore FREEZE
- 1 = FREEZE the current state of the GPT

STOPP — Stop Prescaler
- 0 = Normal operation
- 1 = Stop prescaler and pulse accumulator from incrementing. Ignore changes to input pins.

INCP — Increment Prescaler
- 0 = Has no meaning
- 1 = If STOPP is asserted, increment prescaler once and clock input synchronizers once.

SUPV — Supervisor/Unrestricted Data Space
- 0 = Registers with access controlled by SUPV bit are accessible from either user or supervisor privilege level.
- 1 = Registers with access controlled by SUPV bit are restricted to supervisor access only.

IARB — Interrupt Arbitration Field

The IARB field is used to arbitrate between simultaneous interrupt requests of the same priority. Each module that can generate interrupt requests must be assigned a unique, non-zero IARB field value. Refer to **3.8 Interrupts** for more information.

MTR — GPT Module Test Register (Reserved) **$YFF902**

This address is currently unused and returns zeros if read. It is reserved for GPT factory test.

ICR — GPT Interrupt Configuration Register **$YFF904**

15			12	11	10		8	7			4	3	2	1	0
IPA				0	IPL			IVBA				0	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IPA — Interrupt Priority Adjust

Specifies which GPT interrupt source is given highest internal priority

IPL — Interrupt Priority Level

Specifies the priority level of interrupts generated by the GPT.

IVBA — Interrupt Vector Base Address

Most significant nibble of interrupt vector number generated by the GPT when an interrupt service request is acknowledged.

DDRGP/PORTGP — Port GP Data Direction Register/Port GP Data Register **$YFF906**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDGP7	DDGP6	DDGP5	DDGP4	DDGP3	DDGP2	DDGP1	DDGP0	PGP7	PGP6	PGP5	PGP4	PGP3	PGP2	PGP1	PGP0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

When GPT pins are used as an 8-bit port, DDRGP determines whether pins are input or output and PORTGP holds the 8-bit data.

DDRGP[7:0] — Port GP Data Direction Register
0 = Input only
1 = Output
Each bit in DDRGP determines whether the corresponding PORTGP bit is input or output.

OC1M/OC1D — OC1 Action Mask Register/OC1 Action Data Register **$YFF908**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC1M					0	0	0	OC1D					0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

All OC outputs can be controlled by the action of OC1. OC1M contains a mask that determines which pins are affected. OC1D determines what the outputs are.

OC1M[5:1] — OC1 Mask Field
0 = Corresponding output compare pin is not affected by OC1 compare.
1 = Corresponding output compare pin is affected by OC1 compare.
OC1M[5:1] correspond to OC[5:1].

OC1D[5:1] — OC1 Data Field
0 = If OC1 mask bit is set, clear the corresponding output compare pin on OC1 match.
1 = If OC1 mask bit is set, set the corresponding output compare pin on OC1 match.
OC1D[5:1] correspond to OC[5:1].

TCNT — Timer Counter Register **$YFF90A**

TCNT is the 16-bit free-running counter associated with the input capture, output compare, and pulse accumulator functions of the GPT module.

PACTL/PACNT — Pulse Accumulator Control Register/Counter **$YFF90C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAIS	PAEN	PAMOD	PEDGE	PCLKS	I4/O5	PACLK		PACNT							
RESET:															
U	0	0	0	U	0	0	0	0	0	0	0	0	0	0	0

PACTL enables the pulse accumulator and selects either event counting or gated mode. In event counting mode, PACNT is incremented each time an event occurs. In gated mode, it is incremented by an internal clock.

PAIS — PAI Pin State (Read Only)

PAEN — Pulse Accumulator System Enable
0 = Pulse accumulator disabled
1 = Pulse accumulator enabled

PAMOD — Pulse Accumulator Mode
0 = External event counting
1 = Gated time accumulation

PEDGE — Pulse Accumulator Edge Control
The effects of PEDGE and PAMOD are shown in the following table.

PAMOD	PEDGE	Effect
0	0	PAI falling edge increments counter
0	1	PAI rising edge increments counter
1	0	Zero on PAI inhibits counting
1	1	One on PAI inhibits counting

PCLKS — PCLK Pin State (Read Only)

I4/O5 — Input Capture 4/Output Compare 5
0 = Output compare 5 enabled
1 = Input capture 4 enabled

PACLK[1:0] — Pulse Accumulator Clock Select (Gated Mode)

PACLK[1:0]	Pulse Accumulator Clock Selected
00	System Clock Divided by 512
01	Same Clock Used to Increment TCNT
10	TOF Flag from TCNT
11	External Clock, PCLK

PACNT — Pulse Accumulator Counter
8-bit read/write counter used for external event counting or gated time accumulation.

TIC[1:3] — Input Capture Registers 1–3 **$YFF90E, $YFF910, $YFF912**

The input capture registers are 16-bit read-only registers which are used to latch the value of TCNT when a specified transition is detected on the corresponding input capture pin. They are reset to $FFFF.

TOC[1:4] — Output Compare Registers 1–4 **$YFF914, $YFF916, $YFF918, $YFF91A**

The output compare registers are 16-bit read/write registers which can be used as output waveform controls or as elapsed time indicators. For output compare functions, they are written to a desired match value and compared against TCNT to control specified pin actions. They are reset to $FFFF.

TI4/O5 — Input Capture 4/Output Compare 5 Register **$YFF91C**

This register serves either as input capture register 4 or output compare register 5, depending on the state of I4/O5 in PACTL.

TCTL1/TCTL2 — Timer Control Registers 1–2 **$YFF91E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OM5	OL5	OM4	OL4	OM3	OL3	OM2	OL2	EDGE4		EDGE3		EDGE2		EDGE1	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TCTL1 determines output compare mode and output logic level. TCTL2 determines the type of input capture to be performed.

OM/OL[5:2] — Output Compare Mode Bits and Output Compare Level Bits
Each pair of bits specifies an action to be taken when output comparison is successful.

OM/OL[5:2]	Action Taken
00	Timer Disconnected from Output Logic
01	Toggle OCx Output Line
10	Clear OCx Output Line to 0
11	Set OCx Output Line to 1

EDGE[4:1] — Input Capture Edge Control Bits
Each pair of bits configures input sensing logic for the corresponding input capture.

EDGE[4:1]	Configuration
00	Capture Disabled
01	Capture on Rising Edge Only
10	Capture on Falling Edge Only
11	Capture on Any (Rising or Falling) Edge

TMSK1/TMSK2 — Timer Interrupt Mask Registers 1–2 **$YFF920**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I4/O5I	OCI				ICI			TOI	0	PAOVI	PAII	CPROUT	CPR		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMSK1 enables OC and IC interrupts. TMSK2 controls pulse accumulator interrupts and TCNT functions.

I4/O5I — Input Capture 4/Output Compare 5 Interrupt Enable
0 = IC4/OC5 interrupt disabled
1 = IC4/OC5 interrupt requested when I4/O5F flag in TFLG1 is set

OCI[4:1] — Output Compare Interrupt Enable
0 = OC interrupt disabled
1 = OC interrupt requested when OC flag set
OCI[4:1] correspond to OC[4:1].

ICI[3:1] — Input Capture Interrupt Enable
0 = IC interrupt disabled
1 = IC interrupt requested when IC flag set
ICI[3:1] correspond to IC[3:1].

TOI — Timer Overflow Interrupt Enable
0 = Timer overflow interrupt disabled
1 = Interrupt requested when TOF flag is set

PAOVI — Pulse Accumulator Overflow Interrupt Enable
0 = Pulse accumulator overflow interrupt disabled
1 = Interrupt requested when PAOVF flag is set

PAII — Pulse Accumulator Input Interrupt Enable
0 = Pulse accumulator interrupt disabled
1 = Interrupt requested when PAIF flag is set

CPROUT — Compare/Capture Unit Clock Output Enable
0 = Normal operation for OC1 pin
1 = TCNT clock driven out OC1 pin

CPR[2:0] — Timer Prescaler/PCLK Select Field

This field selects one of seven prescaler taps or PCLK to be TCNT input.

CPR[2:0]	System Clock Divide-By Factor
000	4
001	8
010	16
011	32
100	64
101	128
110	256
111	PCLK

TFLG1/TFLG2 — Timer Interrupt Flag Registers 1–2 **$YFF922**

15	14	11	10	8	7	6	5	4	3	2	1	0
I4/O5F	OCF		ICF		TOF	0	PAOVF	PAIF	0	0	0	0
RESET:												
0	0	0	0	0	0	0	0	0	0	0	0	0

These registers show condition flags that correspond to various GPT events. If the corresponding interrupt enable bit in TMSK1/TMSK2 is set, an interrupt occurs.

I4/O5F — Input Capture 4/Output Compare 5 Flag

When I4/O5 in PACTL is zero, this flag is set each time TCNT matches the value in TOC5. When I4/O5 in PACTL is one, the flag is set each time a selected edge is detected at the I4/O5 pin.

OCF[4:1] — Output Compare Flags

An output compare flag is set each time TCNT matches the corresponding TOC register. OCF[4:1] correspond to OC[4:1].

ICF[3:1] — Input Capture Flags

A flag is set each time a selected edge is detected at the corresponding input capture pin. ICF[3:1] correspond to IC[3:1].

TOF — Timer Overflow Flag

This flag is set each time TCNT advances from a value of $FFFF to $0000.

PAOVF — Pulse Accumulator Overflow Flag

This flag is set each time the pulse accumulator counter advances from a value of $FF to $00.

PAIF — Pulse Accumulator Flag

In event counting mode, this flag is set when an active edge is detected on the PAI pin. In gated time accumulation mode, PAIF is set at the end of the timed period.

CFORC/PWMC — Compare Force Register/PWM Control Register C **$YFF924**

15	11	10	9	8	7	6	4	3	2	1	0
FOC		0	FPWMA	FPWMB	PPROUT	PPR		SFA	SFB	F1A	F1B
RESET:											
0	0	0	0	0	0	0	0	0	0	0	0

Setting a bit in CFORC causes a specific output on OC or PWM pins. PWMC sets PWM operating conditions.

FOC[5:1] — Force Output Compare
0 = Has no meaning
1 = Causes pin action programmed for corresponding OC pin, but the OC flag is not set.
FOC[5:1] correspond to OC[5:1].

FPWMA — Force PWMA Value
0 = Normal PWMA operation
1 = The value of F1A is driven out on the PWMA pin, regardless of the state of PPROUT.

FPWMB — Force PWMB Value
0 = Normal PWMB operation
1 = The value of F1B is driven out on the PWMB pin.

PPROUT — PWM Clock Output Enable
0 = Normal PWM operation on PWMA
1 = TCNT clock driven out PWMA pin

PPR[2:0] — PWM Prescaler/PCLK Select
This field selects one of seven prescaler taps, or PCLK, to be PWMCNT input.

PPR[2:0]	System Clock Divide-By Factor
000	2
001	4
010	8
011	16
100	32
101	64
110	128
111	PCLK

SFA — PWMA Slow/Fast Select
0 = PWMA period is 256 PWMCNT increments long.
1 = PWMA period is 32768 PWMCNT increments long.

SFB — PWMB Slow/Fast Select
0 = PWMB period is 256 PWMCNT increments long.
1 = PWMB period is 32768 PWMCNT increments long.
The following table shows the effects of SF settings on PWM frequency for a 16.78-MHz system clock and a 20.97-MHz system clock.

PPR[2:0]	Prescaler Tap		SFA/B = 0		SFA/B = 1	
	16.78 MHz	20.97 MHz	16.78 MHz	20.97 MHz	16.78 MHz	20.97 MHz
000	Div 2 = 8.39 MHz	Div 2 = 10.5 MHz	32.8 kHz	41 kHz	256 Hz	320 Hz
001	Div 4 = 4.19 MHz	Div 4 = 5.25 MHz	16.4 kHz	20.5 kHz	128 Hz	160 Hz
010	Div 8 = 2.10 MHz	Div 8 = 2.62 MHz	8.19 kHz	10.2 kHz	64.0 Hz	80.0 Hz
011	Div 16 = 1.05 MHz	Div 16 = 1.31 MHz	4.09 kHz	5.15 kHz	32.0 Hz	40.0 Hz
100	Div 32 = 524 kHz	Div 32 = 655 kHz	2.05 kHz	2.56 kHz	16.0 Hz	20.0 Hz
101	Div 64 = 262 kHz	Div 64 = 328 kHz	1.02 kHz	1.28 kHz	8.0 Hz	10.0 Hz
110	Div 128 = 131 kHz	Div 128 = 164 kHz	512 Hz	641 Hz	4.0 Hz	5.0 Hz
111	PCLK	PCLK	PCLK/256	PCLK/256	PCLK/32768	PCLK/32768

F1A — Force Logic Level One on PWMA
0 = Force logic level zero output on PWMA pin
1 = Force logic level one output on PWMA pin

F1B — Force Logic Level One on PWMB
0 = Force logic level zero output on PWMB pin
1 = Force logic level one output on PWMB pin

PWMA/PWMB — PWM Control Registers A/B **$YFF926, $YFF927**

These registers are associated with the pulse-width value of the PWM output on the corresponding PWM pin. A value of $00 loaded into one of these registers results in a continuously low output on the corresponding pin. A value of $80 results in a 50% duty cycle output. Maximum value ($FF) selects an output that is high for 255/256 of the period.

PWMCNT — PWM Count Register **$YFF928**

PWMCNT is the 16-bit free-running counter associated with the PWM functions of the GPT module.

PWMBUFA/B — PWM Buffer Registers A/B **$YFF92A, $YFF92B**

These read-only registers contain values associated with the duty cycles of the corresponding PWM. Reset state is $0000.

PRESCL — GPT Prescaler **$YFF92C**

The 9-bit prescaler value can be read from bits [8:0] at this address. Bits [15:9] always read as zeros. Reset state is $0000.

7 Summary of Changes

This is a complete revision, with complete reprint. All known errors in the publication have been corrected. The following summary lists significant changes. Typographical errors that do not affect content are not annotated.

Page 2	Revised ordering information.
Page 5	New block diagram drawn.
Page 6	New 132-pin assignment diagram drawn.
Page 7	New 144-pin assignment diagram drawn.
Page 8	New address map drawn.
Page 9-13	Added Signal Description section.
Page 14-46	Expanded and revised SIM section. Made all register diagrams and bit mnemonics consistent. Incorporated new information concerning the system clock, resets, interrupts, and chip-select circuits.
Page 47-55	Expanded and revised CPU section. Made all register diagrams and bit mnemonics consistent. Revised instruction set summary information.
Page 56-76	Expanded and revised QSM section. Made all register diagrams and bit mnemonics consistent. Added information concerning SPI and SCI operation.
Page 77-89	Expanded and revised GPT section. Made all registerdiagrams and bit mnemonics consistent. Added information concerning input capture, output compare, and PWM operation.

Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View MC68331CEH20 on WIN SOURCE
- NXP / Nexperia Information

Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management