## Features

- 80C51 Core Architecture
- 256 Bytes of On-chip RAM
- 2048 Bytes of On-chip ERAM
- 64K Bytes of On-chip Flash Memory
  - Data Retention: 10 Years at 85°C
  - Read/Write Cycle: 100K
- 2K Bytes of On-chip Flash for Bootloader
- 2K Bytes of On-chip EEPROM
  Read/Write Cycle: 100K
- Integrated Power Monitor (POR: PFD) To Supervise Internal Power Supply
- 14-sources 4-level Interrupts
- Three 16-bit Timers/Counters
- Full Duplex UART Compatible 80C51
- High-speed Architecture
  - In Standard Mode:
    40 MHz (Vcc 3V to 5.5V, both Internal and external code execution)
    60 MHz (Vcc 4.5V to 5.5V and Internal Code execution only)
  - In X2 mode (6 Clocks/machine cycle)
    20 MHz (Vcc 3V to 5.5V, both Internal and external code execution)
    30 MHz (Vcc 4.5V to 5.5V and Internal Code execution only)
- Five Ports: 32 + 4 Digital I/O Lines
- Five-channel 16-bit PCA with
  - PWM (8-bit)
  - High-speed Output
  - Timer and Edge Capture
- Double Data Pointer
- 21-bit WatchDog Timer (7 Programmable Bits)
- A 10-bit Resolution Analog to Digital Converter (ADC) with 8 Multiplexed Inputs
- SPI Interface, (PLCC52 and VPFP64 packages only)
- Full CAN Controller
  - Fully Compliant with CAN Rev 2.0A and 2.0B
  - Optimized Structure for Communication Management (Via SFR)
  - 15 Independent Message Objects
    - Each Message Object Programmable on Transmission or Reception
    - Individual Tag and Mask Filters up to 29-bit Identifier/Channel
    - 8-byte Cyclic Data Register (FIFO)/Message Object
    - 16-bit Status and Control Register/Message Object
    - 16-bit Time-Stamping Register/Message Object
    - CAN Specification 2.0 Part A or 2.0 Part B Programmable for Each Message Object
    - Access to Message Object Control and Data Registers Via SFR
    - Programmable Reception Buffer Length Up To 15 Message Objects
    - Priority Management of Reception of Hits on Several Message Objects at the Same Time (Basic CAN Feature)
    - Priority Management for Transmission
    - Message Object Overrun Interrupt
  - Supports
    - Time Triggered Communication
    - Autobaud and Listening Mode
    - Programmable Automatic Reply Mode
  - 1-Mbit/s Maximum Transfer Rate at 8 MHz [1] Crystal Frequency in X2 Mode
  - Readable Error Counters
  - Programmable Link to On-chip Timer for Time Stamping and Network Synchronization
  - Independent Baud Rate Prescaler
  - Data, Remote, Error and Overload Frame Handling

1. At BRP = 1 sampling point will be fixed.

# Enhanced 8-bit MCU with CAN Controller and Flash Memory

# AT89C51CC03

Rev. 4182O–CAN–09/08

- **On-chip Emulation Logic (Enhanced Hook System)**
- **Power Saving Modes**
  - **Idle Mode**
  - **Power-down Mode**
- **Power Supply: 3 volts to 5.5 volts**
- **Temperature Range: Industrial (-40° to +85°C), Automotive (-40°C to +125°C)**
- **Packages: VQFP44, PLCC44, VQFP64, PLCC52**

## Description

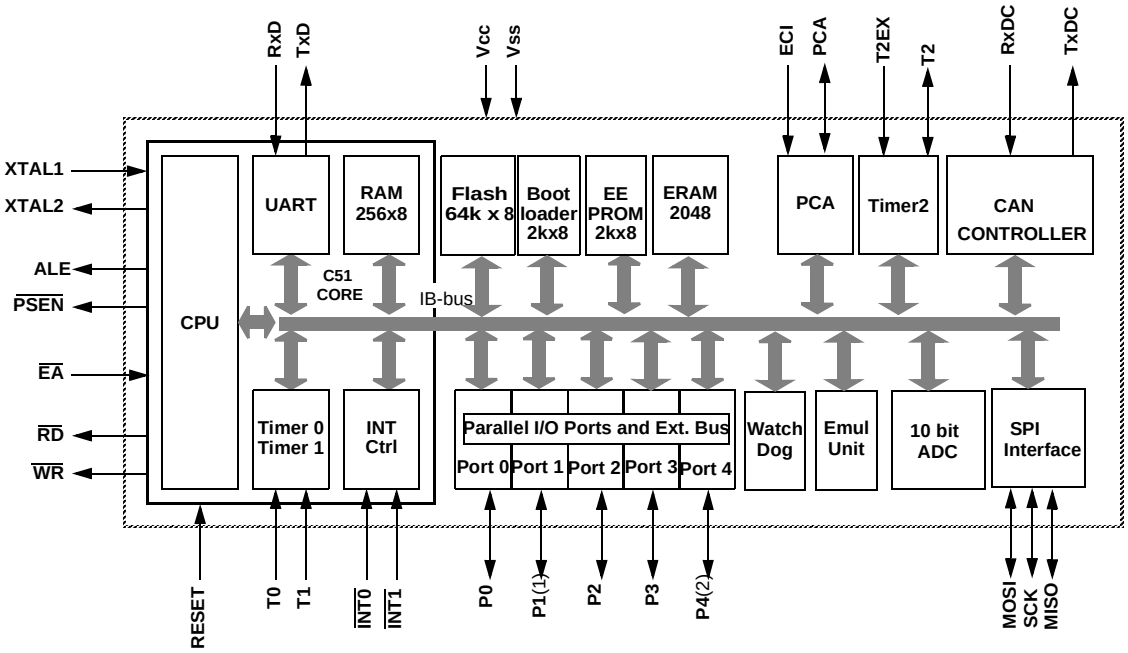
The AT89C51CC03 is a member of the family of 8-bit microcontrollers dedicated to CAN network applications.

In X2 mode a maximum external clock rate of 20 MHz reaches a 300 ns cycle time.

Besides the full CAN controller AT89C51CC03 provides 64K Bytes of Flash memory including In-System Programming (ISP), 2K Bytes Boot Flash Memory, 2K Bytes EEPROM and 2048 byte ERAM.

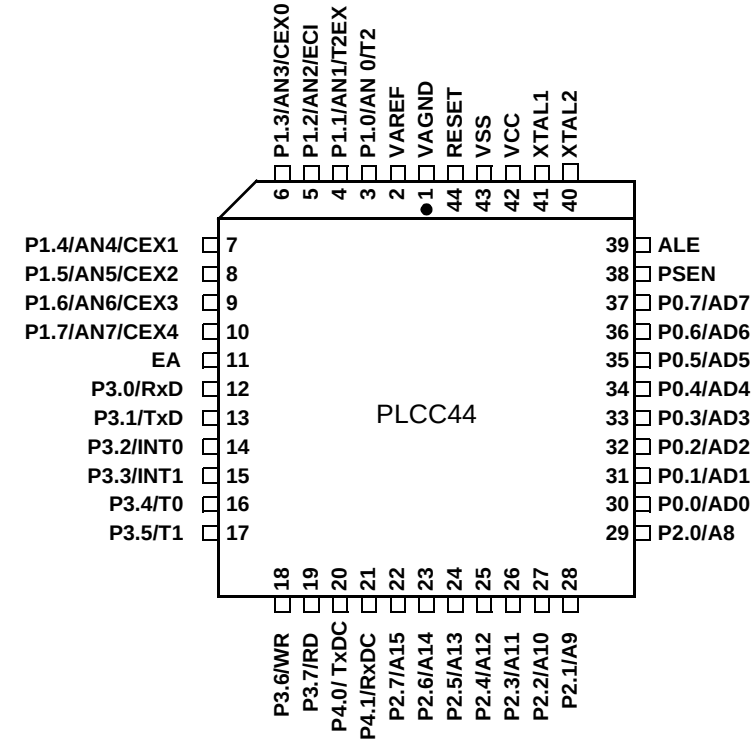
Primary attention is paid to the reduction of the electro-magnetic emission of AT89C51CC03.

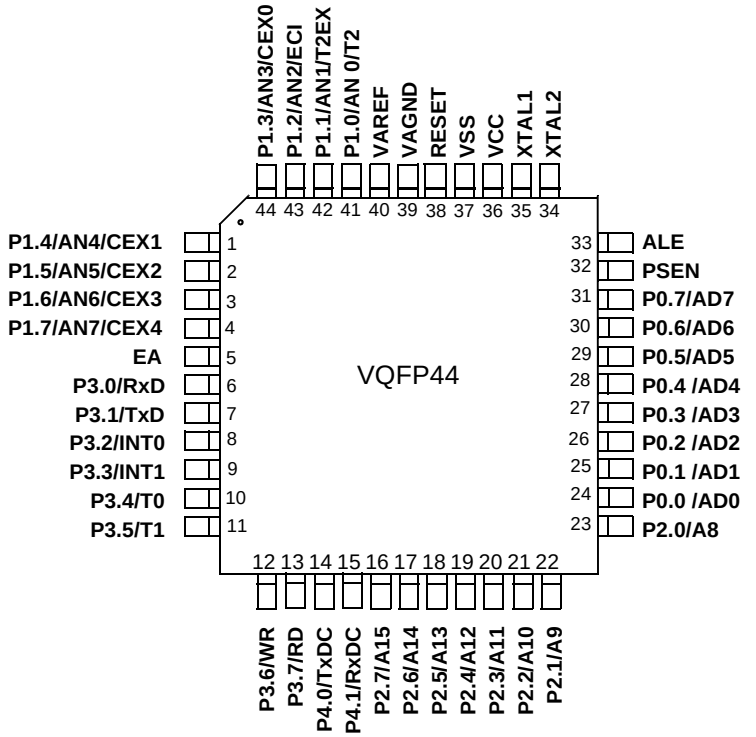
## Block Diagram

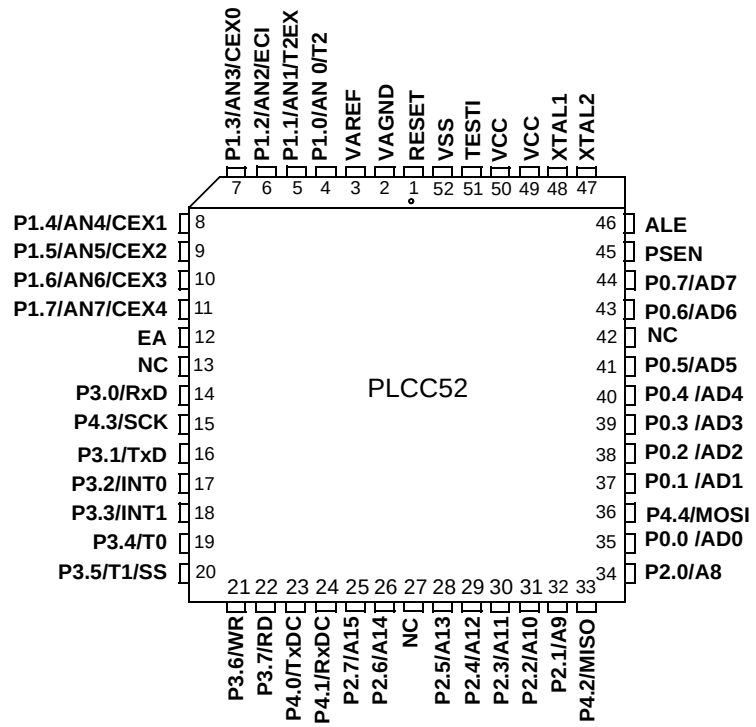



Notes: 1. 8 analog Inputs/8 Digital I/O
2. 5-Bit I/O Port

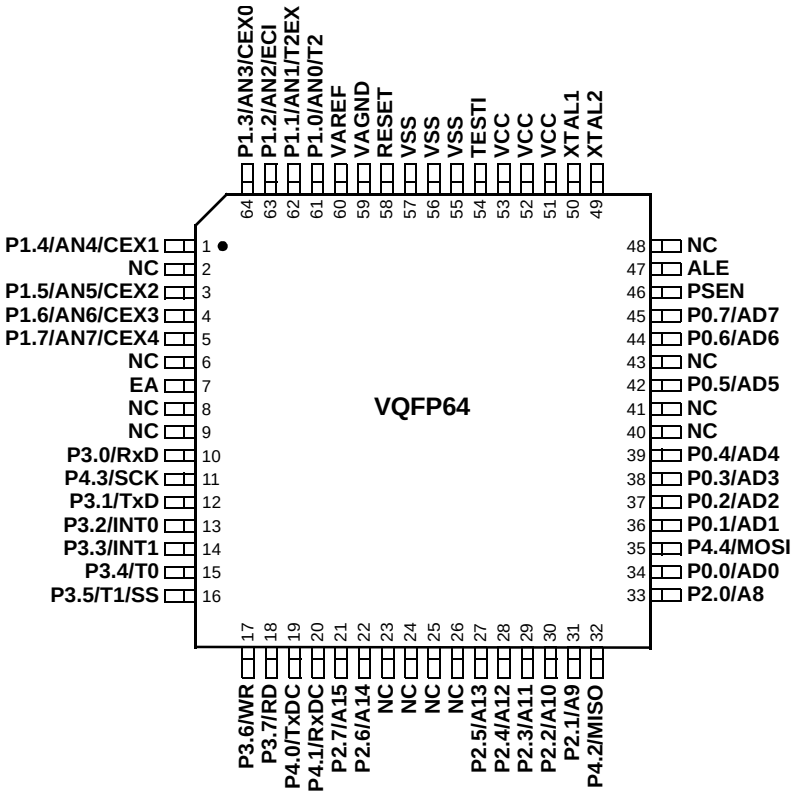
## Pin Configuration






**TESTI must be connected to VSS**




**TESTI must be connected to VSS**

| Pin Name | Type | Description |
|---|---|---|
| VSS | GND | **Circuit ground** |
| TESTI | I | **Must be connected to VSS** |
| VCC | | **Supply Voltage** |
| VAREF | | Reference Voltage for ADC |
| VAGND | | Reference Ground for ADC |
| P0.0:7 | I/O | **Port 0:**<br>Is an 8-bit open drain bi-directional I/O port. Port 0 pins that have 1's written to them float, and in this state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pull-ups when emitting 1's.<br>Port 0 also outputs the code Bytes during program validation. External pull-ups are required during program verification. |
| P1.0:7 | I/O | **Port 1:**<br>Is an 8-bit bi-directional I/O port with internal pull-ups. Port 1 pins can be used for digital input/output or as analog inputs for the Analog Digital Converter (ADC). Port 1 pins that have 1's written to them are pulled high by the internal pull-up transistors and can be used as inputs in this state. As inputs, Port 1 pins that are being pulled low externally will be the source of current ($I_{IL}$, see section "Electrical Characteristic") because of the internal pull-ups. Port 1 pins are assigned to be used as analog inputs via the ADCCF register (in this case the internal pull-ups are disconnected).<br>As a secondary digital function, port 1 contains the Timer 2 external trigger and clock input; the PCA external clock input and the PCA module I/O.<br>P1.0/AN0/T2<br>Analog input channel 0,<br>External clock input for Timer/counter2.<br>P1.1/AN1/T2EX<br>Analog input channel 1,<br>Trigger input for Timer/counter2.<br>P1.2/AN2/ECI<br>Analog input channel 2,<br>PCA external clock input.<br>P1.3/AN3/CEX0<br>Analog input channel 3,<br>PCA module 0 Entry of input/PWM output.<br>P1.4/AN4/CEX1<br>Analog input channel 4,<br>PCA module 1 Entry of input/PWM output.<br>P1.5/AN5/CEX2<br>Analog input channel 5,<br>PCA module 2 Entry of input/PWM output.<br>P1.6/AN6/CEX3<br>Analog input channel 6,<br>PCA module 3 Entry of input/PWM output.<br>P1.7/AN7/CEX4<br>Analog input channel 7,<br>PCA module 4 Entry ot input/PWM output.<br>Port 1 receives the low-order address byte during EPROM programming and program verification.<br>It can drive CMOS inputs without external pull-ups. |
| P2.0:7 | I/O | **Port 2:**<br>Is an 8-bit bi-directional I/O port with internal pull-ups. Port 2 pins that have 1's written to them are pulled high by the internal pull-ups and can be used as inputs in this state. As inputs, Port 2 pins that are being pulled low externally will be a source of current ($I_{IL}$, see section "Electrical Characteristic") because of the internal pull-ups. Port 2 emits the high-order address byte during accesses to the external Program Memory and during accesses to external Data Memory that uses 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1's. During accesses to external Data Memory that use 8 bit addresses (MOVX @Ri), Port 2 transmits the contents of the P2 special function register.<br>It also receives high-order addresses and control signals during program validation.<br>It can drive CMOS inputs without external pull-ups. |

| Pin Name | Type | Description |
|---|---|---|
| P3.0:7 | I/O | **Port 3:**<br>Is an 8-bit bi-directional I/O port with internal pull-ups. Port 3 pins that have 1's written to them are pulled high by the internal pull-up transistors and can be used as inputs in this state. As inputs, Port 3 pins that are being pulled low externally will be a source of current ($I_{IL}$, see section "Electrical Characteristic") because of the internal pull-ups.<br>The output latch corresponding to a secondary function must be programmed to one for that function to operate (except for TxD and $\overline{WR}$). The secondary functions are assigned to the pins of port 3 as follows:<br><br>P3.0/RxD:<br>Receiver data input (asynchronous) or data input/output (synchronous) of the serial interface<br>P3.1/TxD:<br>Transmitter data output (asynchronous) or clock output (synchronous) of the serial interface<br>P3.2/$\overline{INT0}$:<br>External interrupt 0 input/timer 0 gate control input<br>P3.3/$\overline{INT1}$:<br>External interrupt 1 input/timer 1 gate control input<br>P3.4/T0:<br>Timer 0 counter input<br>P3.5/T1/$\overline{SS}$:<br>Timer 1 counter input<br>SPI Slave Select<br>P3.6/$\overline{WR}$:<br>External Data Memory write strobe; latches the data byte from port 0 into the external data memory<br>P3.7/$\overline{RD}$:<br>External Data Memory read strobe; Enables the external data memory.<br>It can drive CMOS inputs without external pull-ups. |
| P4.0:4 | I/O | **Port 4:**<br>Is an 2-bit bi-directional I/O port with internal pull-ups. Port 4 pins that have 1's written to them are pulled high by the internal pull-ups and can be used as inputs in this state. As inputs, Port 4 pins that are being pulled low externally will be a source of current (IIL, on the datasheet) because of the internal pull-up transistor.<br>The output latch corresponding to a secondary function RxDC must be programmed to one for that function to operate. The secondary functions are assigned to the two pins of port 4 as follows:<br><br>P4.0/TxDC:<br>Transmitter output of CAN controller<br>P4.1/RxDC:<br>Receiver input of CAN controller.<br>P4.2/MISO:<br>Master Input Slave Output of SPI controller<br>P4.3/SCK:<br>Serial Clock of SPI controller<br>P4.4/MOSI:<br>Master Ouput Slave Input of SPI controller<br><br>It can drive CMOS inputs without external pull-ups. |

| Pin Name | Type | Description |
|---|---|---|
| RESET | I/O | **Reset:**<br>A high level on this pin during two machine cycles while the oscillator is running resets the device. An internal pull-down resistor to VSS permits power-on reset using only an external capacitor to VCC. |
| ALE | O | **ALE:**<br>An Address Latch Enable output for latching the low byte of the address during accesses to the external memory. The ALE is activated every 1/6 oscillator periods (1/3 in X2 mode) except during an external data memory access. When instructions are executed from an internal Flash ($\overline{EA}$ = 1), ALE generation can be disabled by the software. |
| PSEN | O | **$\overline{PSEN}$:**<br>The Program Store Enable output is a control signal that enables the external program memory of the bus during external fetch operations. It is activated twice each machine cycle during fetches from the external program memory. However, when executing from of the external program memory two activations of PSEN are skipped during each access to the external Data memory. The PSEN is not activated for internal fetches. |
| EA | I | **$\overline{EA}$:**<br>When External Access is held at the high level, instructions are fetched from the internal Flash. When held at the low level, AT89C51CC03 fetches all instructions from the external program memory. |
| XTAL1 | I | **XTAL1:**<br>Input of the inverting oscillator amplifier and input of the internal clock generator circuits.<br>To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left unconnected. To operate above a frequency of 16 MHz, a duty cycle of 50% should be maintained. |
| XTAL2 | O | **XTAL2:**<br>Output from the inverting oscillator amplifier. |

## I/O Configurations

Each Port SFR operates via type-D latches, as illustrated in Figure 1 for Ports 3 and 4. A CPU "write to latch" signal initiates transfer of internal bus data into the type-D latch. A CPU "read latch" signal transfers the latched Q output onto the internal bus. Similarly, a "read pin" signal transfers the logical level of the Port pin. Some Port data instructions activate the "read latch" signal while others activate the "read pin" signal. Latch instructions are referred to as Read-Modify-Write instructions. Each I/O line may be independently programmed as input or output.

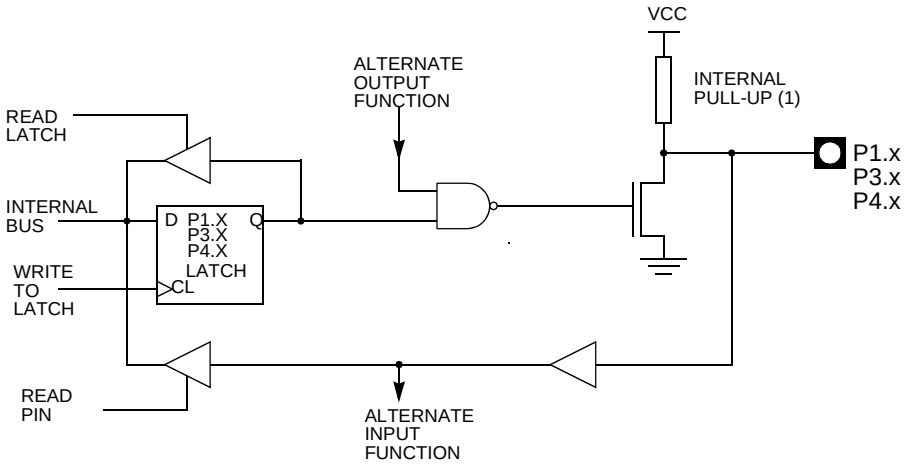
## Port 1, Port 3 and Port 4

Figure 1 shows the structure of Ports 1 and 3, which have internal pull-ups. An external source can pull the pin low. Each Port pin can be configured either for general-purpose I/O or for its alternate input output function.

To use a pin for general-purpose output, set or clear the corresponding bit in the Px register (x = 1,3 or 4). To use a pin for general-purpose input, set the bit in the Px register. This turns off the output FET drive.

To configure a pin for its alternate function, set the bit in the Px register. When the latch is set, the "alternate output function" signal controls the output level (see Figure 1). The operation of Ports 1, 3 and 4 is discussed further in the "quasi-Bidirectional Port Operation" section.

**Figure 1.** Port 1, Port 3 and Port 4 Structure



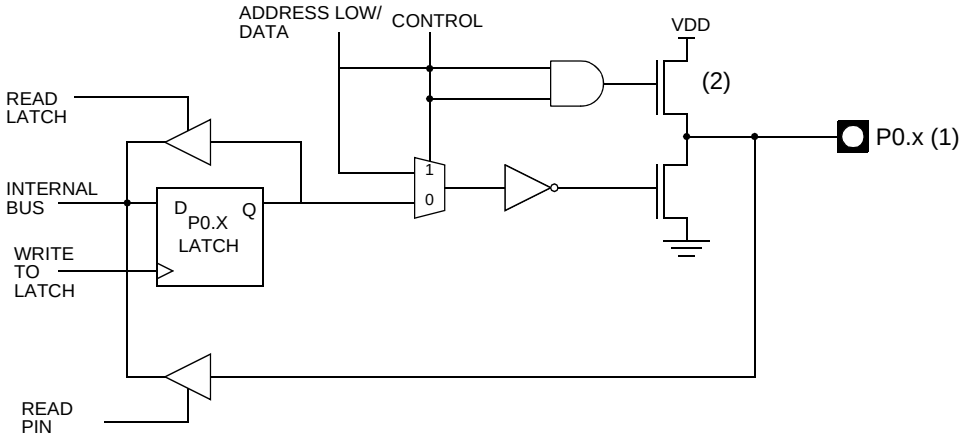

Note: The internal pull-up can be disabled on P1 when analog function is selected.

## Port 0 and Port 2

Ports 0 and 2 are used for general-purpose I/O or as the external address/data bus. Port 0, shown in Figure 3, differs from the other Ports in not having internal pull-ups. Figure 3 shows the structure of Port 2. An external source can pull a Port 2 pin low.

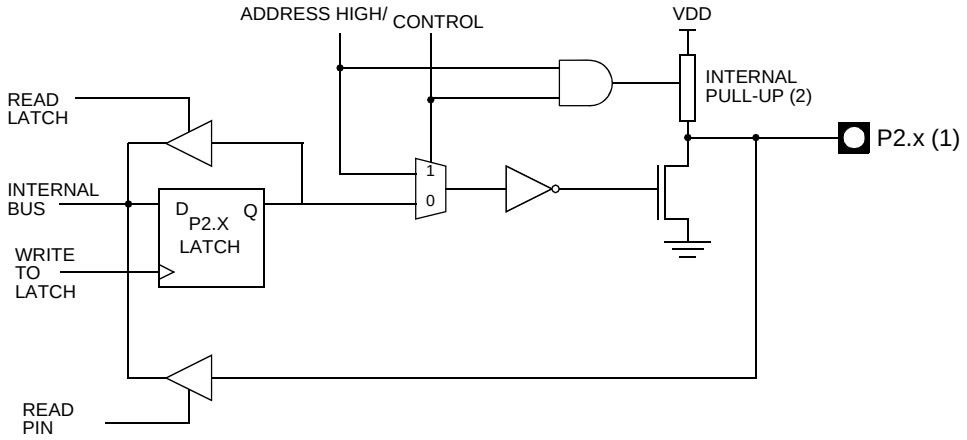
To use a pin for general-purpose output, set or clear the corresponding bit in the Px register (x = 0 or 2). To use a pin for general-purpose input, set the bit in the Px register to turn off the output driver FET.

**Figure 2.** Port 0 Structure




Notes: 1. Port 0 is precluded from use as general-purpose I/O Ports when used as address/data bus drivers.
2. Port 0 internal strong pull-ups assist the logic-one output for memory bus cycles only. Except for these bus cycles, the pull-up FET is off, Port 0 outputs are open-drain.

**Figure 3.** Port 2 Structure




Notes: 1. Port 2 is precluded from use as general-purpose I/O Ports when as address/data bus drivers.
2. Port 2 internal strong pull-ups FET (P1 in FiGURE) assist the logic-one output for memory bus cycle.

When Port 0 and Port 2 are used for an external memory cycle, an internal control signal switches the output-driver input from the latch output to the internal address/data line.

## Read-Modify-Write Instructions

Some instructions read the latch data rather than the pin data. The latch based instructions read the data, modify the data and then rewrite the latch. These are called "Read-Modify-Write" instructions. Below is a complete list of these special instructions (see Table ). When the destination operand is a Port or a Port bit, these instructions read the latch rather than the pin:

| Instruction | Description | Example |
|---|---|---|
| ANL | logical AND | ANL P1, A |
| ORL | logical OR | ORL P2, A |
| XRL | logical EX-OR | XRL P3, A |
| JBC | jump if bit = 1 and clear bit | JBC P1.1, LABEL |
| CPL | complement bit | CPL P3.0 |
| INC | increment | INC P2 |
| DEC | decrement | DEC P2 |
| DJNZ | decrement and jump if not zero | DJNZ P3, LABEL |
| MOV Px.y, C | move carry bit to bit y of Port x | MOV P1.5, C |
| CLR Px.y | clear bit y of Port x | CLR P2.4 |
| SET Px.y | set bit y of Port x | SET P3.3 |

It is not obvious the last three instructions in this list are Read-Modify-Write instructions. These instructions read the port (all 8 bits), modify the specifically addressed bit and

write the new byte back to the latch. These Read-Modify-Write instructions are directed to the latch rather than the pin in order to avoid possible misinterpretation of voltage (and therefore, logic) levels at the pin. For example, a Port bit used to drive the base of an external bipolar transistor can not rise above the transistor's base-emitter junction voltage (a value lower than VIL). With a logic one written to the bit, attempts by the CPU to read the Port at the pin are misinterpreted as logic zero. A read of the latch rather than the pins returns the correct logic-one value.

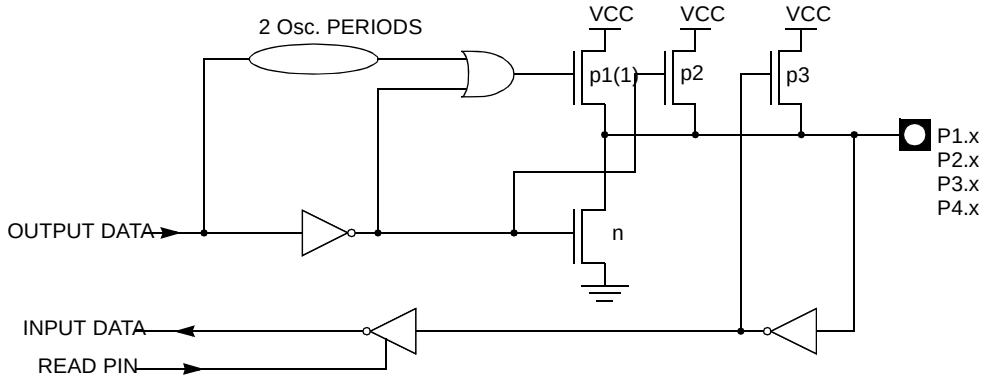
## Quasi-Bidirectional Port Operation

Port 1, Port 2, Port 3 and Port 4 have fixed internal pull-ups and are referred to as "quasi-bidirectional" Ports. When configured as an input, the pin impedance appears as logic one and sources current in response to an external logic zero condition. Port 0 is a "true bidirectional" pin. The pins float when configured as input. Resets write logic one to all Port latches. If logical zero is subsequently written to a Port latch, it can be returned to input conditions by a logical one written to the latch.

Note: Port latch values change near the end of Read-Modify-Write instruction cycles. Output buffers (and therefore the pin state) update early in the instruction after Read-Modify-Write instruction cycle.

Logical zero-to-one transitions in Port 1, Port 2, Port 3 and Port 4 use an additional pull-up (p1) to aid this logic transition (see Figure 4.). This increases switch speed. This extra pull-up sources 100 times normal internal circuit current during 2 oscillator clock periods. The internal pull-ups are field-effect transistors rather than linear resistors. Pull-ups consist of three p-channel FET (pFET) devices. A pFET is on when the gate senses logical zero and off when the gate senses logical one. pFET #1 is turned on for two oscillator periods immediately after a zero-to-one transition in the Port latch. A logical one at the Port pin turns on pFET #3 (a weak pull-up) through the inverter. This inverter and pFET pair form a latch to drive logical one. pFET #2 is a very weak pull-up switched on whenever the associated nFET is switched off. This is traditional CMOS switch convention. Current strengths are 1/10 that of pFET #3.

**Figure 4.** Internal Pull-Up Configurations




Note: Port 2 p1 assists the logic-one output for memory bus cycles.

## SFR Mapping

The Special Function Registers (SFRs) of the AT89C51CC03 fall into the following categories:

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| ACC | E0h | Accumulator | – | – | – | – | – | – | – | – |
| B | F0h | B Register | – | – | – | – | – | – | – | – |
| PSW | D0h | Program Status Word | CY | AC | F0 | RS1 | RS0 | OV | F1 | P |
| SP | 81h | Stack Pointer | – | – | – | – | – | – | – | – |
| DPL | 82h | Data Pointer Low byte<br>LSB of DPTR | – | – | – | – | – | – | – | – |
| DPH | 83h | Data Pointer High byte<br>MSB of DPTR | – | – | – | – | – | – | – | – |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| P0 | 80h | Port 0 | – | – | – | – | – | – | – | – |
| P1 | 90h | Port 1 | – | – | – | – | – | – | – | – |
| P2 | A0h | Port 2 | – | – | – | – | – | – | – | – |
| P3 | B0h | Port 3 | – | – | – | – | – | – | – | – |
| P4 | C0h | Port 4 (x5) | – | – | – | P4.4 / MOSI | P4.3 / SCK | P4.2 / MISO | P4.1 / RxDC | P4.0 / TxDC |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| TH0 | 8Ch | Timer/Counter 0 High byte | – | – | – | – | – | – | – | – |
| TL0 | 8Ah | Timer/Counter 0 Low byte | – | – | – | – | – | – | – | – |
| TH1 | 8Dh | Timer/Counter 1 High byte | – | – | – | – | – | – | – | – |
| TL1 | 8Bh | Timer/Counter 1 Low byte | – | – | – | – | – | – | – | – |
| TH2 | CDh | Timer/Counter 2 High byte | – | – | – | – | – | – | – | – |
| TL2 | CCh | Timer/Counter 2 Low byte | – | – | – | – | – | – | – | – |
| TCON | 88h | Timer/Counter 0 and 1 control | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
| TMOD | 89h | Timer/Counter 0 and 1 Modes | GATE1 | C/T1# | M11 | M01 | GATE0 | C/T0# | M10 | M00 |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| T2CON | C8h | Timer/Counter 2 control | TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/T2# | CP/RL2# |
| T2MOD | C9h | Timer/Counter 2 Mode | – | – | – | – | – | – | T2OE | DCEN |
| RCAP2H | CBh | Timer/Counter 2 Reload/Capture High byte | – | – | – | – | – | – | – | – |
| RCAP2L | CAh | Timer/Counter 2 Reload/Capture Low byte | – | – | – | – | – | – | – | – |
| WDTRST | A6h | WatchDog Timer Reset | – | – | – | – | – | – | – | – |
| WDTPRG | A7h | WatchDog Timer Program | – | – | – | – | – | S2 | S1 | S0 |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| SCON | 98h | Serial Control | FE/SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |
| SBUF | 99h | Serial Data Buffer | – | – | – | – | – | – | – | – |
| SADEN | B9h | Slave Address Mask | – | – | – | – | – | – | – | – |
| SADDR | A9h | Slave Address | – | – | – | – | – | – | – | – |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| CCON | D8h | PCA Timer/Counter Control | CF | CR | – | CCF4 | CCF3 | CCF2 | CCF1 | CCF0 |
| CMOD | D9h | PCA Timer/Counter Mode | CIDL | WDTE | – | – | – | CPS1 | CPS0 | ECF |
| CL | E9h | PCA Timer/Counter Low byte | – | – | – | – | – | – | – | – |
| CH | F9h | PCA Timer/Counter High byte | – | – | – | – | – | – | – | – |
| CCAPM0 | DAh | PCA Timer/Counter Mode 0 | | ECOM0 | CAPP0 | CAPN0 | MAT0 | TOG0 | PWM0 | ECCF0 |
| CCAPM1 | DBh | PCA Timer/Counter Mode 1 | | ECOM1 | CAPP1 | CAPN1 | MAT1 | TOG1 | PWM1 | ECCF1 |
| CCAPM2 | DCh | PCA Timer/Counter Mode 2 | – | ECOM2 | CAPP2 | CAPN2 | MAT2 | TOG2 | PWM2 | ECCF2 |
| CCAPM3 | DDh | PCA Timer/Counter Mode 3 | | ECOM3 | CAPP3 | CAPN3 | MAT3 | TOG3 | PWM3 | ECCF3 |
| CCAPM4 | DEh | PCA Timer/Counter Mode 4 | | ECOM4 | CAPP4 | CAPN4 | MAT4 | TOG4 | PWM4 | ECCF4 |
| CCAP0H | FAh | PCA Compare Capture Module 0 H | CCAP0H7 | CCAP0H6 | CCAP0H5 | CCAP0H4 | CCAP0H3 | CCAP0H2 | CCAP0H1 | CCAP0H0 |
| CCAP1H | FBh | PCA Compare Capture Module 1 H | CCAP1H7 | CCAP1H6 | CCAP1H5 | CCAP1H4 | CCAP1H3 | CCAP1H2 | CCAP1H1 | CCAP1H0 |
| CCAP2H | FCh | PCA Compare Capture Module 2 H | CCAP2H7 | CCAP2H6 | CCAP2H5 | CCAP2H4 | CCAP2H3 | CCAP2H2 | CCAP2H1 | CCAP2H0 |
| CCAP3H | FDh | PCA Compare Capture Module 3 H | CCAP3H7 | CCAP3H6 | CCAP3H5 | CCAP3H4 | CCAP3H3 | CCAP3H2 | CCAP3H1 | CCAP3H0 |
| CCAP4H | FEh | PCA Compare Capture Module 4 H | CCAP4H7 | CCAP4H6 | CCAP4H5 | CCAP4H4 | CCAP4H3 | CCAP4H2 | CCAP4H1 | CCAP4H0 |
| CCAP0L | EAh | PCA Compare Capture Module 0 L | CCAP0L7 | CCAP0L6 | CCAP0L5 | CCAP0L4 | CCAP0L3 | CCAP0L2 | CCAP0L1 | CCAP0L0 |
| CCAP1L | EBh | PCA Compare Capture Module 1 L | CCAP1L7 | CCAP1L6 | CCAP1L5 | CCAP1L4 | CCAP1L3 | CCAP1L2 | CCAP1L1 | CCAP1L0 |
| CCAP2L | ECh | PCA Compare Capture Module 2 L | CCAP2L7 | CCAP2L6 | CCAP2L5 | CCAP2L4 | CCAP2L3 | CCAP2L2 | CCAP2L1 | CCAP2L0 |
| CCAP3L | EDh | PCA Compare Capture Module 3 L | CCAP3L7 | CCAP3L6 | CCAP3L5 | CCAP3L4 | CCAP3L3 | CCAP3L2 | CCAP3L1 | CCAP3L0 |
| CCAP4L | EEh | PCA Compare Capture Module 4 L | CCAP4L7 | CCAP4L6 | CCAP4L5 | CCAP4L4 | CCAP4L3 | CCAP4L2 | CCAP4L1 | CCAP4L0 |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| IEN0 | A8h | Interrupt Enable Control 0 | EA | EC | ET2 | ES | ET1 | EX1 | ET0 | EX0 |
| IEN1 | E8h | Interrupt Enable Control 1 | – | – | – | – | ESPI | ETIM | EADC | ECAN |
| IPL0 | B8h | Interrupt Priority Control Low 0 | – | PPC | PT2 | PS | PT1 | PX1 | PT0 | PX0 |
| IPH0 | B7h | Interrupt Priority Control High 0 | – | PPCH | PT2H | PSH | PT1H | PX1H | PT0H | PX0H |
| IPL1 | F8h | Interrupt Priority Control Low 1 | – | – | – | – | SPIL | POVRL | PADCL | PCANL |
| IPH1 | F7h | Interrupt Priority Control High1 | – | – | – | – | SPIH | POVRH | PADCH | PCANH |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| ADCON | F3h | ADC Control | – | PSIDLE | ADEN | ADEOC | ADSST | SCH2 | SCH1 | SCH0 |
| ADCF | F6h | ADC Configuration | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| ADCLK | F2h | ADC Clock | – | – | – | PRS4 | PRS3 | PRS2 | PRS1 | PRS0 |
| ADDH | F5h | ADC Data High byte | ADAT9 | ADAT8 | ADAT7 | ADAT6 | ADAT5 | ADAT4 | ADAT3 | ADAT2 |
| ADDL | F4h | ADC Data Low byte | – | – | – | – | – | – | ADAT1 | ADAT0 |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| CANGCON | ABh | CAN General Control | ABRQ | OVRQ | TTC | SYNCTTC | AUT–BAUD | TEST | ENA | GRES |
| CANGSTA | AAh | CAN General Status | – | OVFG | – | TBSY | RBSY | ENFG | BOFF | ERRP |
| CANGIT | 9Bh | CAN General Interrupt | CANIT | – | OVRTIM | OVRBUF | SERG | CERG | FERG | AERG |
| CANBT1 | B4h | CAN Bit Timing 1 | – | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 | – |
| CANBT2 | B5h | CAN Bit Timing 2 | – | SJW1 | SJW0 | – | PRS2 | PRS1 | PRS0 | – |
| CANBT3 | B6h | CAN Bit Timing 3 | – | PHS22 | PHS21 | PHS20 | PHS12 | PHS11 | PHS10 | SMP |
| CANEN1 | CEh | CAN Enable Channel byte 1 | – | ENCH14 | ENCH13 | ENCH12 | ENCH11 | ENCH10 | ENCH9 | ENCH8 |
| CANEN2 | CFh | CAN Enable Channel byte 2 | ENCH7 | ENCH6 | ENCH5 | ENCH4 | ENCH3 | ENCH2 | ENCH1 | ENCH0 |
| CANGIE | C1h | CAN General Interrupt Enable | – | – | ENRX | ENTX | ENERCH | ENBUF | ENERG | – |
| CANIE1 | C2h | CAN Interrupt Enable Channel byte 1 | – | IECH14 | IECH13 | IECH12 | IECH11 | IECH10 | IECH9 | IECH8 |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| CANIE2 | C3h | CAN Interrupt Enable Channel byte 2 | IECH7 | IECH6 | IECH5 | IECH4 | IECH3 | IECH2 | IECH1 | IECH0 |
| CANSIT1 | BAh | CAN Status Interrupt Channel byte1 | – | SIT14 | SIT13 | SIT12 | SIT11 | SIT10 | SIT9 | SIT8 |
| CANSIT2 | BBh | CAN Status Interrupt Channel byte2 | SIT7 | SIT6 | SIT5 | SIT4 | SIT3 | SIT2 | SIT1 | SIT0 |
| CANTCON | A1h | CAN Timer Control | TPRESC 7 | TPRESC 6 | TPRESC 5 | TPRESC 4 | TPRESC 3 | TPRESC 2 | TPRESC 1 | TPRESC 0 |
| CANTIMH | ADh | CAN Timer high | CANTIM 15 | CANTIM 14 | CANTIM 13 | CANTIM 12 | CANTIM 11 | CANTIM 10 | CANTIM 9 | CANTIM 8 |
| CANTIML | ACh | CAN Timer low | CANTIM 7 | CANTIM 6 | CANTIM 5 | CANTIM 4 | CANTIM 3 | CANTIM 2 | CANTIM 1 | CANTIM 0 |
| CANSTMPH | AFh | CAN Timer Stamp high | TIMSTMP 15 | TIMSTMP 14 | TIMSTMP 13 | TIMSTMP 12 | TIMSTMP 11 | TIMSTMP 10 | TIMSTMP 9 | TIMSTMP 8 |
| CANSTMPL | AEh | CAN Timer Stamp low | TIMSTMP7 | TIMSTMP 6 | TIMSTMP 5 | TIMSTMP 4 | TIMSTMP 3 | TIMSTMP 2 | TIMSTMP 1 | TIMSTMP 0 |
| CANTTCH | A5h | CAN Timer TTC high | TIMTTC 15 | TIMTTC 14 | TIMTTC 13 | TIMTTC 12 | TIMTTC 11 | TIMTTC 10 | TIMTTC 9 | TIMTTC 8 |
| CANTTCL | A4h | CAN Timer TTC low | TIMTTC 7 | TIMTTC 6 | TIMTTC 5 | TIMTTC 4 | TIMTTC 3 | TIMTTC 2 | TIMTTC 1 | TIMTTC 0 |
| CANTEC | 9Ch | CAN Transmit Error Counter | TEC7 | TEC6 | TEC5 | TEC4 | TEC3 | TEC2 | TEC1 | TEC0 |
| CANREC | 9Dh | CAN Receive Error Counter | REC7 | REC6 | REC5 | REC4 | REC3 | REC2 | REC1 | REC0 |
| CANPAGE | B1h | CAN Page | CHNB3 | CHNB2 | CHNB1 | CHNB0 | AINC | INDX2 | INDX1 | INDX0 |
| CANSTCH | B2h | CAN Status Channel | DLCW | TXOK | RXOK | BERR | SERR | CERR | FERR | AERR |
| CANCONCH | B3h | CAN Control Channel | CONCH1 | CONCH0 | RPLV | IDE | DLC3 | DLC2 | DLC1 | DLC0 |
| CANMSG | A3h | CAN Message Data | MSG7 | MSG6 | MSG5 | MSG4 | MSG3 | MSG2 | MSG1 | MSG0 |
| CANIDT1 | BCh | CAN Identifier Tag byte 1(Part A)<br>CAN Identifier Tag byte 1(PartB) | IDT10<br>IDT28 | IDT9<br>IDT27 | IDT8<br>IDT26 | IDT7<br>IDT25 | IDT6<br>IDT24 | IDT5<br>IDT23 | IDT4<br>IDT22 | IDT3<br>IDT21 |
| CANIDT2 | BDh | CAN Identifier Tag byte 2 (PartA)<br>CAN Identifier Tag byte 2 (PartB) | IDT2<br>IDT20 | IDT1<br>IDT19 | IDT0<br>IDT18 | –<br>IDT17 | –<br>IDT16 | –<br>IDT15 | –<br>IDT14 | –<br>IDT13 |
| CANIDT3 | BEh | CAN Identifier Tag byte 3(PartA)<br>CAN Identifier Tag byte 3(PartB) | –<br>IDT12 | –<br>IDT11 | –<br>IDT10 | –<br>IDT9 | –<br>IDT8 | –<br>IDT7 | –<br>IDT6 | –<br>IDT5 |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| CANIDT4 | BFh | CAN Identifier Tag byte 4(PartA)<br>CAN Identifier Tag byte 4(PartB) | –<br>IDT4 | –<br>IDT3 | –<br>IDT2 | –<br>IDT1 | –<br>IDT0 | RTRTAG | –<br>RB1TAG | RB0TAF |
| CANIDM1 | C4h | CAN Identifier Mask byte 1(PartA)<br>CAN Identifier Mask byte 1(PartB) | IDMSK10<br>IDMSK28 | IDMSK9<br>IDMSK27 | IDMSK8<br>IDMSK26 | IDMSK7<br>IDMSK25 | IDMSK6<br>IDMSK24 | IDMSK5<br>IDMSK23 | IDMSK4<br>IDMSK22 | IDMSK3<br>IDMSK21 |
| CANIDM2 | C5h | CAN Identifier Mask byte 2(PartA)<br>CAN Identifier Mask byte 2(PartB) | IDMSK2<br>IDMSK20 | IDMSK1<br>IDMSK19 | IDMSK0<br>IDMSK18 | –<br>IDMSK17 | –<br>IDMSK16 | –<br>IDMSK15 | –<br>IDMSK14 | –<br>IDMSK13 |
| CANIDM3 | C6h | CAN Identifier Mask byte 3(PartA)<br>CAN Identifier Mask byte 3(PartB) | –<br>IDMSK12 | –<br>IDMSK11 | –<br>IDMSK10 | –<br>IDMSK9 | –<br>IDMSK8 | –<br>IDMSK7 | –<br>IDMSK6 | –<br>IDMSK5 |
| CANIDM4 | C7h | CAN Identifier Mask byte 4(PartA)<br>CAN Identifier Mask byte 4(PartB) | –<br>IDMSK4 | –<br>IDMSK3 | –<br>IDMSK2 | –<br>IDMSK1 | –<br>IDMSK0 | RTRMSK | – | IDEMSK |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| SPCON | D4h | SPI Control | SPR2 | SPEN | SSDIS | MSTR | CPOL | CPHA | SPR1 | SPR0 |
| SPSCR | D5h | SPI Status and Control | SPIF | - | OVR | MODF | SPTE | UARTM | SPTEIE | MOFIE |
| SPDAT | D6h | SPI Data | - | - | - | - | - | - | - | - |

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PCON | 87h | Power Control | SMOD1 | SMOD0 | – | POF | GF1 | GF0 | PD | IDL |
| AUXR | 8Eh | Auxiliary Register 0 | DPU | VPFDP | M0 | XRS2 | XRS1 | XRS0 | EXTRAM | A0 |
| AUXR1 | A2h | Auxiliary Register 1 | – | – | ENBOOT | – | GF3 | 0 | – | DPS |
| CKCON0 | 8Fh | Clock Control 0 | CANX2 | WDX2 | PCAX2 | SIX2 | T2X2 | T1X2 | T0X2 | X2 |
| CKCON1 | 9Fh | Clock Control 1 | - | - | - | - | - | - | - | SPIX2 |
| FCON | D1h | Flash Control | FPL3 | FPL2 | FPL1 | FPL0 | FPS | FMOD1 | FMOD0 | FBUSY |
| EECON | D2h | EEPROM Contol | EEPL3 | EEPL2 | EEPL1 | EEPL0 | – | – | EEE | EEBUSY |
| FSTA | D3 | Flash Status | - | - | - | - | - | - | SEQERR | FLOAD |

**Table 1.** SFR Mapping

| | 0/8[2] | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F | |
|---|---|---|---|---|---|---|---|---|---|
| F8h | IPL1<br>xxxx x000 | CH<br>0000 0000 | CCAP0H<br>0000 0000 | CCAP1H<br>0000 0000 | CCAP2H<br>0000 0000 | CCAP3H<br>0000 0000 | CCAP4H<br>0000 0000 | | FFh |
| F0h | B<br>0000 0000 | | ADCLK<br>xxx0 0000 | ADCON<br>x000 0000 | ADDL<br>0000 0000 | ADDH<br>0000 0000 | ADCF<br>0000 0000 | IPH1<br>xxxx x000 | F7h |
| E8h | IEN1<br>xxxx x000 | CL<br>0000 0000 | CCAP0L<br>0000 0000 | CCAP1L<br>0000 0000 | CCAP2L<br>0000 0000 | CCAP3L<br>0000 0000 | CCAP4L<br>0000 0000 | | EFh |
| E0h | ACC<br>0000 0000 | | | | | | | | E7h |
| D8h | CCON<br>0000 0000 | CMOD<br>00xx x000 | CCAPM0<br>x000 0000 | CCAPM1<br>x000 0000 | CCAPM2<br>x000 0000 | CCAPM3<br>x000 0000 | CCAPM4<br>x000 0000 | | DFh |
| D0h | PSW<br>0000 0000 | FCON<br>0000 0000 | EECON<br>xxxx xx00 | FSTA<br>xxxx xx00 | SPCON<br>0001 0100 | SPSCR<br>0000 0000 | SPDAT<br>xxxx xxxx | | D7h |
| C8h | T2CON<br>0000 0000 | T2MOD<br>xxxx xx00 | RCAP2L<br>0000 0000 | RCAP2H<br>0000 0000 | TL2<br>0000 0000 | TH2<br>0000 0000 | CANEN1<br>x000 0000 | CANEN2<br>0000 0000 | CFh |
| C0h | P4<br>xxx1 1111 | CANGIE<br>xx00 000x | CANIE1<br>x000 0000 | CANIE2<br>0000 0000 | CANIDM1<br>xxxx xxxx | CANIDM2<br>xxxx xxxx | CANIDM3<br>xxxx xxxx | CANIDM4<br>xxxx xxxx | C7h |
| B8h | IPL0<br>x000 0000 | SADEN<br>0000 0000 | CANSIT1<br>0000 0000 | CANSIT2<br>0000 0000 | CANIDT1<br>xxxx xxxx | CANIDT2<br>xxxx xxxx | CANIDT3<br>xxxx xxxx | CANIDT4<br>xxxx xxxx | BFh |
| B0h | P3<br>1111 1111 | CANPAGE<br>0000 0000 | CANSTCH<br>xxxx xxxx | CANCONCH<br>xxxx xxxx | CANBT1<br>xxxx xxxx | CANBT2<br>xxxx xxxx | CANBT3<br>xxxx xxxx | IPH0<br>x000 0000 | B7h |
| A8h | IEN0<br>0000 0000 | SADDR<br>0000 0000 | CANGSTA<br>x0x0 0000 | CANGCON<br>0000 0x00 | CANTIML<br>0000 0000 | CANTIMH<br>0000 0000 | CANSTMPL<br>0000 0000 | CANSTMPH<br>0000 0000 | AFh |
| A0h | P2<br>1111 1111 | CANTCON<br>0000 0000 | AUXR1<br>xxxx 00x0 | CANMSG<br>xxxx xxxx | CANTTCL<br>0000 0000 | CANTTCH<br>0000 0000 | WDTRST<br>1111 1111 | WDTPRG<br>xxxx x000 | A7h |
| 98h | SCON<br>0000 0000 | SBUF<br>0000 0000 | | CANGIT<br>0x00 0000 | CANTEC<br>0000 0000 | CANREC<br>0000 0000 | | CKCON1<br>xxxx xxx0 | 9Fh |
| 90h | P1<br>1111 1111 | | | | | | | | 97h |
| 88h | TCON<br>0000 0000 | TMOD<br>0000 0000 | TL0<br>0000 0000 | TL1<br>0000 0000 | TH0<br>0000 0000 | TH1<br>0000 0000 | AUXR<br>x001 0100 | CKCON0<br>0000 0000 | 8Fh |
| 80h | P0<br>1111 1111 | SP<br>0000 0111 | DPL<br>0000 0000 | DPH<br>0000 0000 | | | | PCON<br>00x1 0000 | 87h |
| | 0/8[2] | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F | |

Reserved

Note: 1. Do not read or write Reserved Registers

2. These registers are bit–addressable.
Sixteen addresses in the SFR space are both byte–addressable and bit–addressable. The bit–addressable SFR's are those whose address ends in 0 and 8. The bit addresses, in this area, are 0x80 through to 0xFF.

## Clock

The AT89C51CC03 core needs only 6 clock periods per machine cycle. This feature, called”X2”, provides the following advantages:

- Divides frequency crystals by 2 (cheaper crystals) while keeping the same CPU power.
- Saves power consumption while keeping the same CPU power (oscillator power saving).
- Saves power consumption by dividing dynamic operating frequency by 2 in operating and idle modes.
- Increases CPU power by 2 while keeping the same crystal frequency.

In order to keep the original C51 compatibility, a divider-by-2 is inserted between the XTAL1 signal and the main clock input of the core (phase generator). This divider may be disabled by the software.

An extra feature is available to start after Reset in the X2 mode. This feature can be enabled by a bit X2B in the Hardware Security Byte. This bit is described in the section "In-System Programming".

## Description

The X2 bit in the CKCON register (see Table 2) allows switching from 12 clock cycles per instruction to 6 clock cycles and vice versa. At reset, the standard speed is activated (STD mode).

Setting this bit activates the X2 feature (X2 mode) for the CPU Clock only (see Figure 5.).

The Timers 0, 1 and 2, Uart, PCA, WatchDog or CAN switch in X2 mode only if the corresponding bit is cleared in the CKCON register.

The clock for the whole circuit and peripheral is first divided by two before being used by the CPU core and peripherals. This allows any cyclic ratio to be accepted on the XTAL1 input. In X2 mode, as this divider is bypassed, the signals on XTAL1 must have a cyclic ratio between 40 to 60%. Figure 5. shows the clock generation block diagram. The X2 bit is validated on the XTAL1÷2 rising edge to avoid glitches when switching from the X2 to the STD mode. Figure 6 shows the mode switching waveforms.

**Figure 5.** Clock CPU Generation Diagram

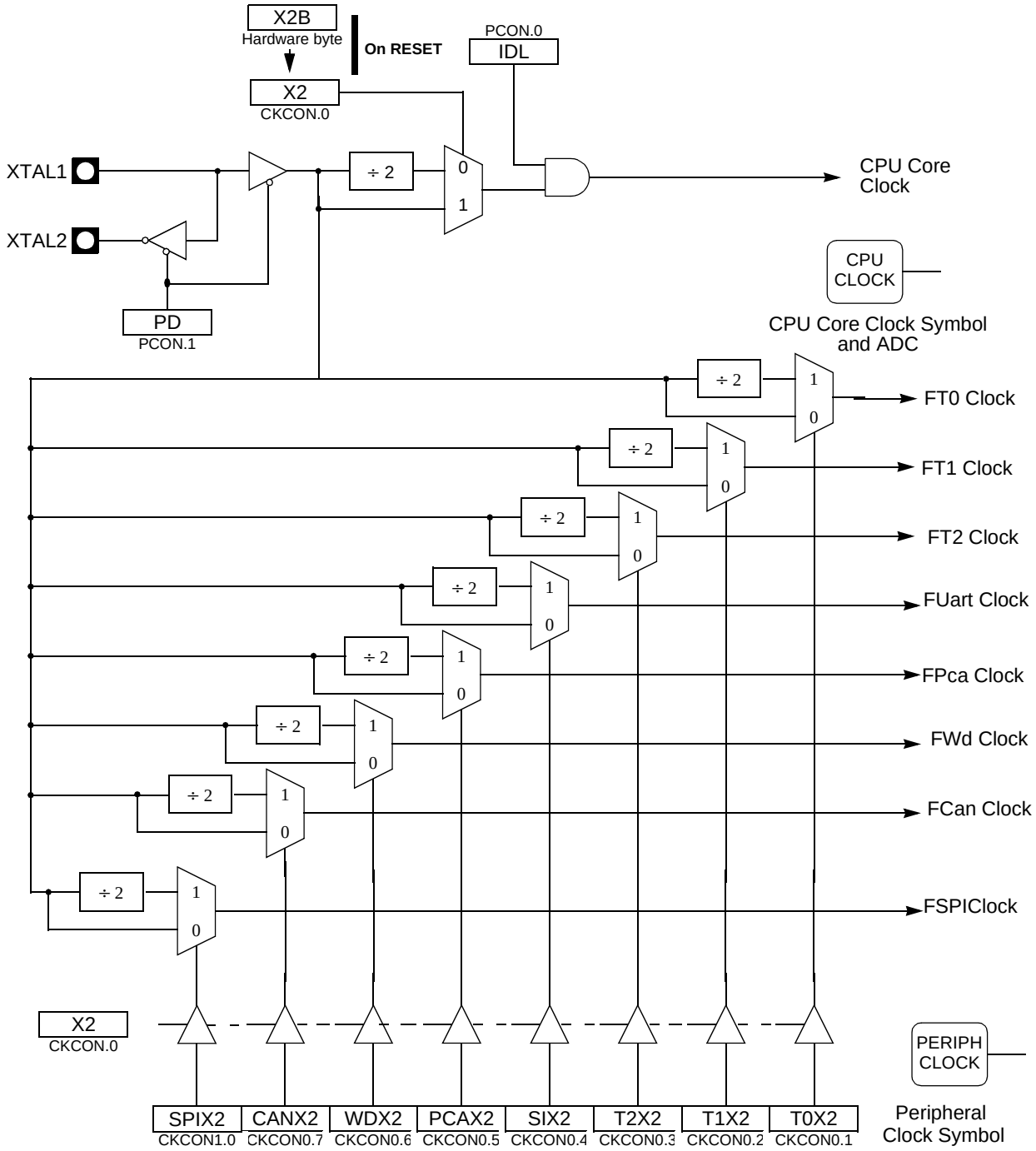
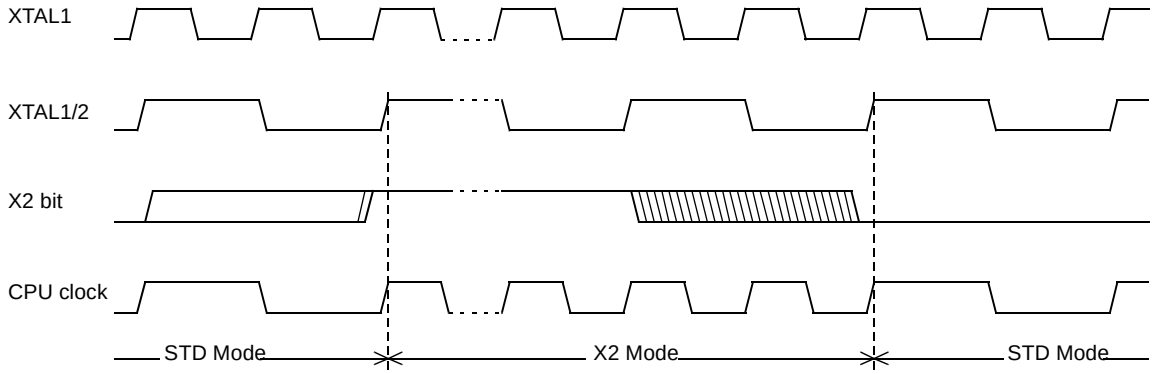


**Figure 6.** Mode Switching Waveforms




Note: In order to prevent any incorrect operation while operating in the X2 mode, users must be aware that all peripherals using the clock frequency as a time reference (UART, timers...) will have their time reference divided by two. For example a free running timer generating an interrupt every 20 ms will then generate an interrupt every 10 ms. A UART with a 4800 baud rate will have a 9600 baud rate.

## Registers

**Table 2.** CKCON0 Register

CKCON0 (S:8Fh)
Clock Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **CANX2** | **WDX2** | **PCAX2** | **SIX2** | **T2X2** | **T1X2** | **T0X2** | **X2** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | CANX2 | **CAN clock** [1]<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 6 | WDX2 | **WatchDog clock** [1]<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 5 | PCAX2 | **Programmable Counter Array clock** [1]<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 4 | SIX2 | **Enhanced UART clock (MODE 0 and 2)** [1]<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 3 | T2X2 | **Timer2 clock** [1]<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 2 | T1X2 | **Timer1 clock** [1]<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 1 | T0X2 | **Timer0 clock** [1]<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 0 | X2 | **CPU clock**<br>Clear to select 12 clock periods per machine cycle (STD mode) for CPU and all the peripherals.<br>Set to select 6 clock periods per machine cycle (X2 mode) and to enable the individual peripherals "X2"bits. |

Note: 1. This control bit is validated when the CPU clock bit X2 is set; when X2 is low, this bit has no effect.

Reset Value = 0000 0000b

**Table 3.** CKCON1 Register

CKCON1 (S:9Fh)
Clock Control Register 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | SPIX2 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-1 | - | **Reserved**<br>The value read from these bits is indeterminate. Do not set these bits. |
| 0 | SPIX2 | **SPI clock** (1)<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |

Note: 1. This control bit is validated when the CPU clock bit X2 is set; when X2 is low, this bit has no effect.

Reset Value = 0000 0000b

## Data Memory

The AT89C51CC03 provides data memory access in two different spaces:

1. The internal space mapped in three separate segments:
- the lower 128 Bytes RAM segment.
- the upper 128 Bytes RAM segment.
- the expanded 2048 Bytes RAM segment (ERAM).
2. The external space.

A fourth internal segment is available but dedicated to Special Function Registers, SFRs, (addresses 80h to FFh) accessible by direct addressing mode.

Figure 8 shows the internal and external data memory spaces organization.

**Figure 7.** Internal Memory - RAM

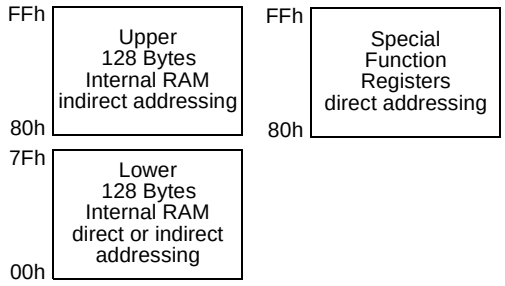
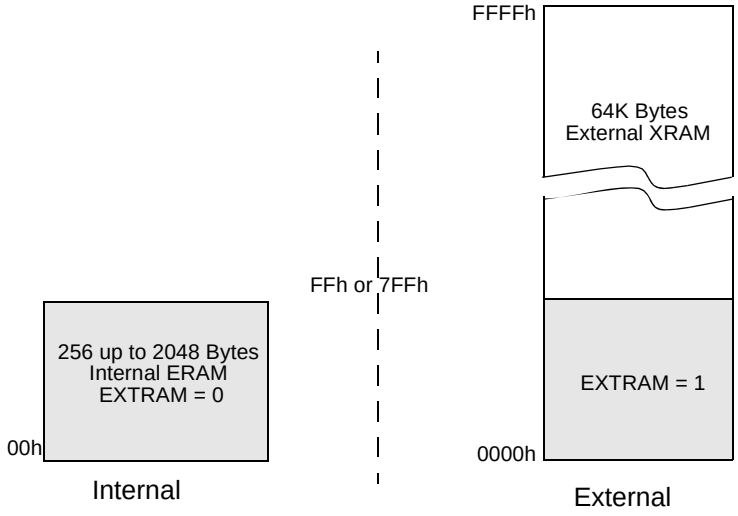



**Figure 8.** Internal and External Data Memory Organization ERAM-XRAM



## Internal Space

### Lower 128 Bytes RAM

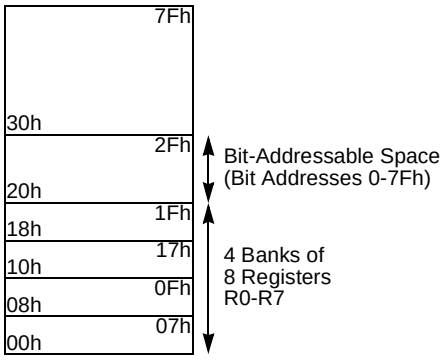
The lower 128 Bytes of RAM (see Figure 8) are accessible from address 00h to 7Fh using direct or indirect addressing modes. The lowest 32 Bytes are grouped into 4 banks of 8 registers (R0 to R7). Two bits RS0 and RS1 in PSW register (see Figure 6) select which bank is in use according to Table 4. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing, and can be used for context switching in interrupt service routines.

**Table 4.** Register Bank Selection

| RS1 | RS0 | Description |
|---|---|---|
| 0 | 0 | Register bank 0 from 00h to 07h |
| 0 | 1 | Register bank 0 from 08h to 0Fh |
| 1 | 0 | Register bank 0 from 10h to 17h |
| 1 | 1 | Register bank 0 from 18h to 1Fh |

The next 16 Bytes above the register banks form a block of bit-addressable memory space. The C51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00h to 7Fh.

**Figure 9.** Lower 128 Bytes Internal RAM Organization




### Upper 128 Bytes RAM

The upper 128 Bytes of RAM are accessible from address 80h to FFh using only indirect addressing mode.

### Expanded RAM

The on-chip 2048 Bytes of expanded RAM (ERAM) are accessible from address 0000h to 07FFh using indirect addressing mode through MOVX instructions. In this address range, the bit EXTRAM in AUXR register is used to select the ERAM (default) or the XRAM. As shown in Figure 8 when EXTRAM = 0, the ERAM is selected and when EXTRAM = 1, the XRAM is selected.

The size of ERAM can be configured by XRS2-0 bit in AUXR register (default size is 2048 Bytes).

Note: Lower 128 Bytes RAM, Upper 128 Bytes RAM, and expanded RAM are made of volatile memory cells. This means that the RAM content is indeterminate after power-up and must then be initialized properly.

## External Space

### Memory Interface

The external memory interface comprises the external bus (port 0 and port 2) as well as the bus control signals (RD#, WR#, and ALE).

Figure 10 shows the structure of the external address bus. P0 carries address A7:0 while P2 carries address A15:8. Data D7:0 is multiplexed with A7:0 on P0. Table 5 describes the external memory interface signals.

**Figure 10.** External Data Memory Interface Structure

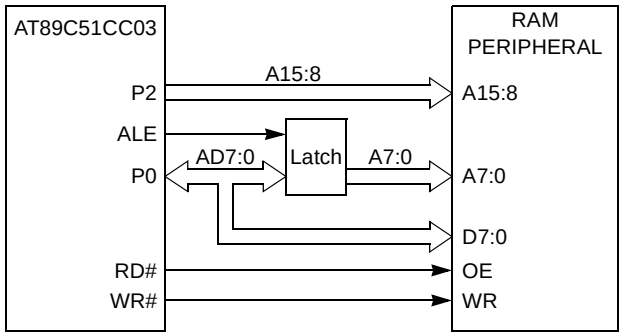



**Table 5.** External Data Memory Interface Signals

| Signal Name | Type | Description | Alternative Function |
|---|---|---|---|
| A15:8 | O | **Address Lines**<br>Upper address lines for the external bus. | P2.7:0 |
| AD7:0 | I/O | **Address/Data Lines**<br>Multiplexed lower address lines and data for the external memory. | P0.7:0 |
| ALE | O | **Address Latch Enable**<br>ALE signals indicates that valid address information are available on lines AD7:0. | - |
| RD# | O | **Read**<br>Read signal output to external data memory. | P3.7 |
| WR# | O | **Write**<br>Write signal output to external memory. | P3.6 |

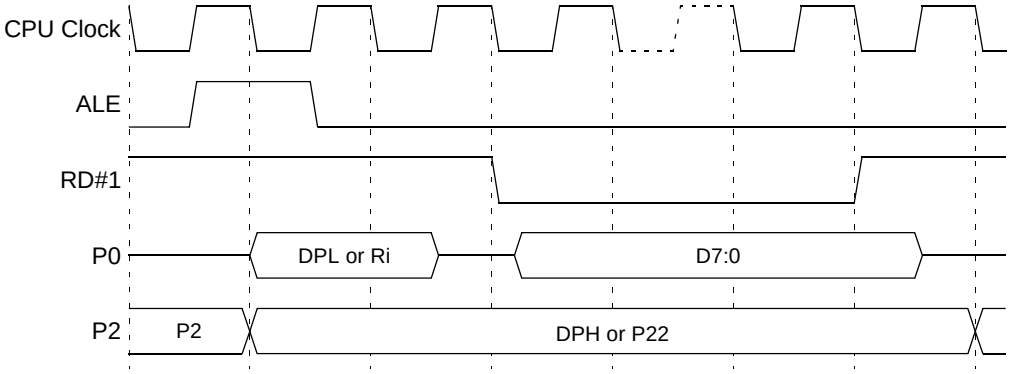
### External Bus Cycles

This section describes the bus cycles the AT89C51CC03 executes to read (see Figure 11), and write data (see Figure 12) in the external data memory.
External memory cycle takes 6 CPU clock periods. This is equivalent to 12 oscillator clock period in standard mode or 6 oscillator clock periods in X2 mode. For further information on X2 mode.

Slow peripherals can be accessed by stretching the read and write cycles. This is done using the M0 bit in AUXR register. Setting this bit changes the width of the RD# and WR# signals from 3 to 15 CPU clock periods.

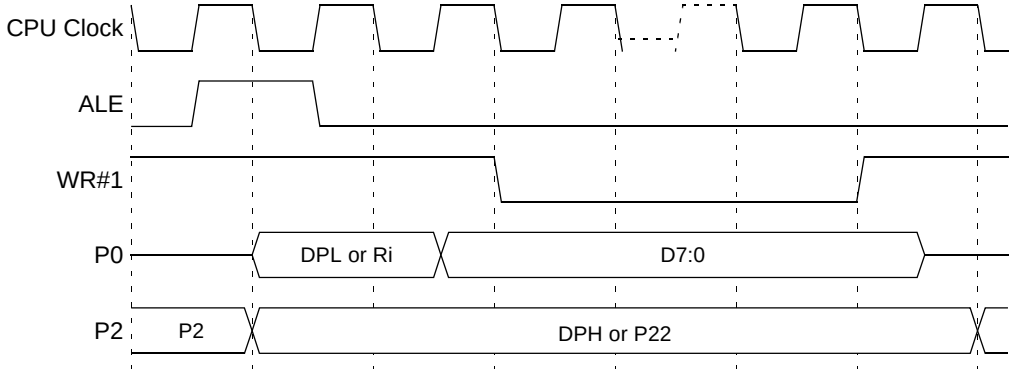
For simplicity, the accompanying figures depict the bus cycle waveforms in idealized form and do not provide precise timing information. For bus cycle timing parameters refer to the Section “AC Characteristics” of the AT89C51CC03 datasheet.

**Figure 11.** External Data Read Waveforms




Notes: 1. RD# signal may be stretched using M0 bit in AUXR register.
2. When executing MOVX @Ri instruction, P2 outputs SFR content.

**Figure 12.** External Data Write Waveforms



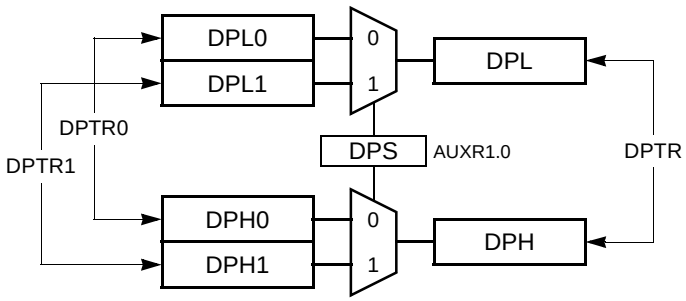

Notes: 1. WR# signal may be stretched using M0 bit in AUXR register.
2. When executing MOVX @Ri instruction, P2 outputs SFR content.

## Dual Data Pointer

### Description

The AT89C51CC03 implements a second data pointer for speeding up code execution and reducing code size in case of intensive usage of external memory accesses.
DPTR 0 and DPTR 1 are seen by the CPU as DPTR and are accessed using the SFR addresses 83h and 84h that are the DPH and DPL addresses. The DPS bit in AUXR1 register (see Figure 8) is used to select whether DPTR is the data pointer 0 or the data pointer 1 (see Figure 13).

**Figure 13.** Dual Data Pointer Implementation




### Application

Software can take advantage of the additional data pointers to both increase speed and reduce code size, for example, block operations (copy, compare…) are well served by using one data pointer as a “source” pointer and the other one as a “destination” pointer. Hereafter is an example of block move implementation using the two pointers and coded in assembler. The latest C compiler takes also advantage of this feature by providing enhanced algorithm libraries.

The INC instruction is a short (2 Bytes) and fast (6 machine cycle) way to manipulate the DPS bit in the AUXR1 register. However, note that the INC instruction does not directly force the DPS bit to a particular state, but simply toggles it. In simple routines, such as the block move example, only the fact that DPS is toggled in the proper sequence matters, not its actual value. In other words, the block move routine works the same whether DPS is '0' or '1' on entry.

```
; ASCII block move using dual data pointers
; Modifies DPTR0, DPTR1, A and PSW
; Ends when encountering NULL character
; Note: DPS exits opposite to the entry state unless an extra INC AUXR1 is added

AUXR1EQU0A2h

move:movDPTR,#SOURCE ; address of SOURCE
   incAUXR1 ; switch data pointers
   movDPTR,#DEST ; address of DEST
mv_loop:incAUXR1; switch data pointers
   movxA,@DPTR; get a byte from SOURCE
   incDPTR; increment SOURCE address
   incAUXR1; switch data pointers
   movx@DPTR,A; write the byte to DEST
   incDPTR; increment DEST address
   jnzmv_loop; check for NULL terminator
end_move:
```

## Registers

**Table 6.** PSW Register

PSW (S:8Eh)
Program Status Word Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **CY** | **AC** | **F0** | **RS1** | **RS0** | **OV** | **F1** | **P** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | CY | **Carry Flag**<br>Carry out from bit 1 of ALU operands. |
| 6 | AC | **Auxiliary Carry Flag**<br>Carry out from bit 1 of addition operands. |
| 5 | F0 | **User Definable Flag 0.** |
| 4-3 | RS1:0 | **Register Bank Select Bits**<br>Refer to Table 4 for bits description. |
| 2 | OV | **Overflow Flag**<br>Overflow set by arithmetic operations. |
| 1 | F1 | **User Definable Flag 1** |
| 0 | P | **Parity Bit**<br>Set when ACC contains an odd number of 1's.<br>Cleared when ACC contains an even number of 1's. |

Reset Value = 0000 0000b

**Table 7.** AUXR Register

AUXR (S:8Eh)
Auxiliary Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **-** | **-** | **M0** | **XRS2** | **XRS1** | **XRS0** | **EXTRAM** | **A0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-6 | - | **Reserved**<br>The value read from these bits are indeterminate. Do not set this bit. |
| 5 | M0 | **Stretch MOVX control:**<br>the RD/ and the WR/ pulse length is increased according to the value of M0.<br>**M0** **Pulse length in clock period**<br>0 6<br>1 30 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 4-2 | XRS1-0 | **ERAM size:**<br>Accessible size of the ERAM<br>**XRS 2:0 ERAM size**<br>000 256 Bytes<br>001 512 Bytes<br>010 768 Bytes<br>011 1024 Bytes<br>100 1792 Bytes<br>101 2048 Bytes (default configuration after reset)<br>110 Reserved<br>111 Reserved |
| 1 | EXTRAM | **Internal/External RAM (00h - FFh)**<br>access using MOVX @ Ri/@ DPTR<br>0 - Internal ERAM access using MOVX @ Ri/@ DPTR.<br>1 - External data memory access. |
| 0 | A0 | **Disable/Enable ALE)**<br>0 - ALE is emitted at a constant rate of 1/6 the oscillator frequency (or 1/3 if X2 mode is used)<br>1 - ALE is active only during a MOVX or MOVC instruction. |

Reset Value = X001 0100b
Not bit addressable

**Table 8.** AUXR1 Register

AUXR1 (S:A2h)
Auxiliary Control Register 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | ENBOOT | - | GF3 | 0 | - | DPS |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-6 | - | **Reserved**<br>The value read from these bits is indeterminate. Do not set these bits. |
| 5 | ENBOOT | **Enable Boot Flash**<br>Set this bit for map the boot Flash between F800h -FFFFh<br>Clear this bit for disable boot Flash. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | GF3 | **General-purpose Flag 3** |
| 2 | 0 | **Always Zero**<br>This bit is stuck to logic 0 to allow INC AUXR1 instruction without affecting GF3 flag. |
| 1 | - | **Reserved for Data Pointer Extension.** |
| 0 | DPS | **Data Pointer Select Bit**<br>Set to select second dual data pointer: DPTR1.<br>Clear to select first dual data pointer: DPTR0. |

Reset Value = XXXX 00X0b

## Power Monitor

The POR/PFD function monitors the internal power-supply of the CPU core memories and the peripherals, and if needed, suspends their activity when the internal power supply falls below a safety threshold. This is achieved by applying an internal reset to them.

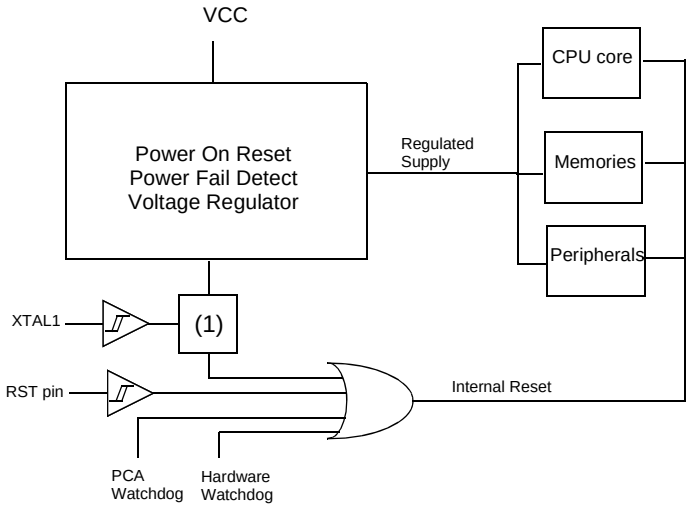
By generating the Reset the Power Monitor insures a correct start up when AT89C51CC03 is powered up.

## Description

In order to startup and maintain the microcontroller in correct operating mode, $V_{CC}$ has to be stabilized in the $V_{CC}$ operating range and the oscillator has to be stabilized with a nominal amplitude compatible with logic level VIH/VIL.

These parameters are controlled during the three phases: power-up, normal operation and power going down. See Figure 14.

**Figure 14.** Power Monitor Block Diagram

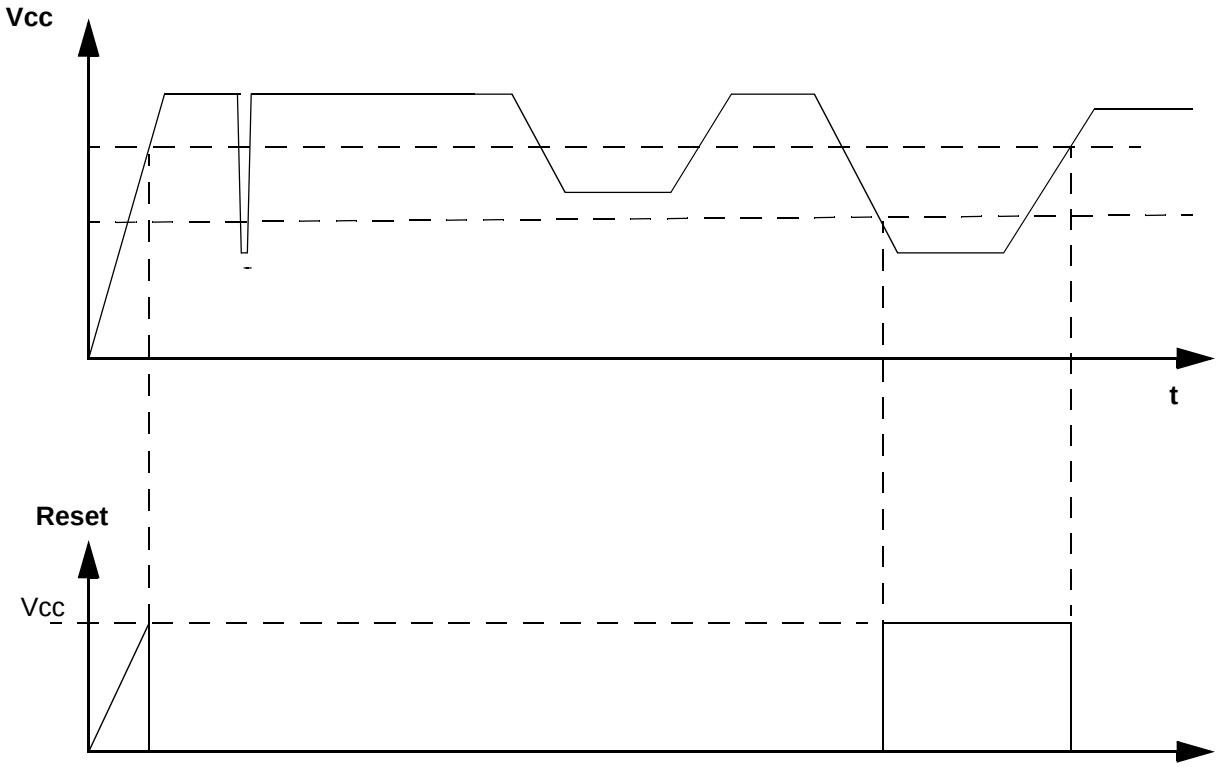



Note: 1. Once XTAL1 high and low levels reach above and below VIH/VIL a 1024 clock period delay will extend the reset coming from the Power Fail Detect. If the power falls below the Power Fail Detect thresthold level, the reset will be applied immediately.

The Voltage regulator generates a regulated internal supply for the CPU core the memories and the peripherals. Spikes on the external Vcc are smoothed by the voltage regulator.

The Power fail detect monitor the supply generated by the voltage regulator and generate a reset if this supply falls below a safety threshold as illustrated in the Figure 15.

**Figure 15.** Power Fail Detect




When the power is applied, the Power Monitor immediately asserts a reset. Once the internal supply after the voltage regulator reach a safety level, the power monitor then looks at the XTAL clock input. The internal reset will remain asserted until the Xtal1 levels are above and below VIH and VIL. Further more. An internal counter will count 1024 clock periods before the reset is de-asserted.

If the internal power supply falls below a safety level, a reset is immediately asserted.

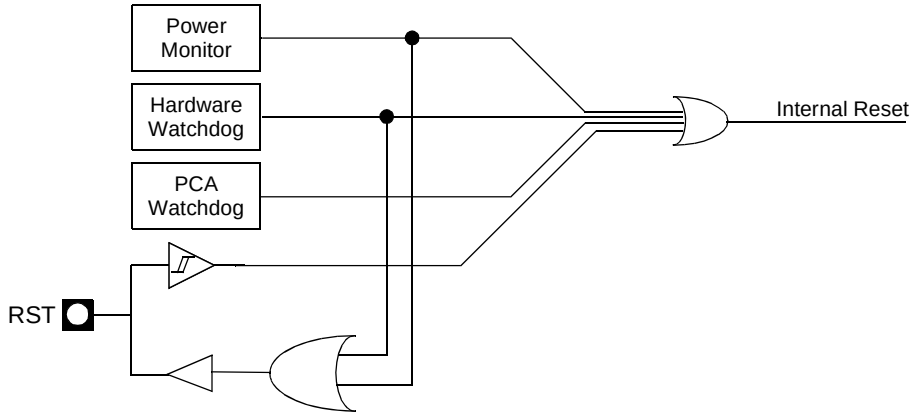
.

# Reset

## Introduction

The reset sources are : Power Management, Hardware Watchdog, PCA Watchdog and Reset input.

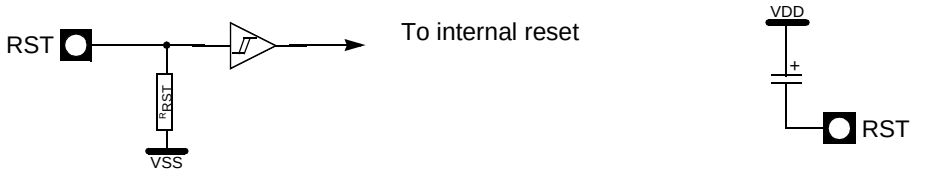
**Figure 16.** Reset Schematic




## Reset Input

The Reset input can be used to force a reset pulse longer than the internal reset controlled by the Power Monitor. RST input has a pull-down resistor allowing power-on reset by simply connecting an external capacitor to $V_{CC}$ as shown in Figure 17. Resistor value and input characteristics are discussed in the Section “DC Characteristics” of the AT89C51CC03 datasheet. The status of the Port pins during reset is detailed in Table 9.

**Figure 17.** Reset Circuitry and Power-On Reset



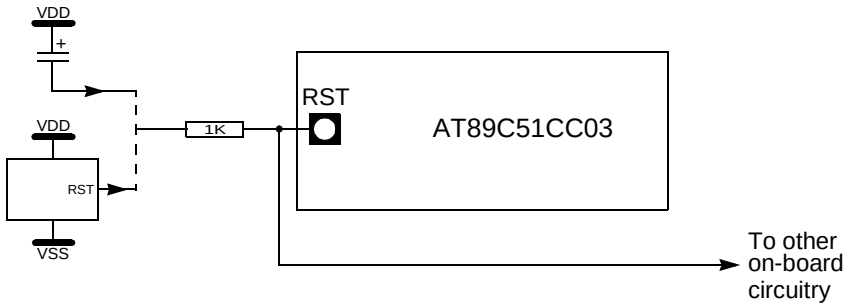

a. RST input circuitry

b. Power-on Reset

## Reset Output

As detailed in Section “Watchdog Timer”, page 81, the WDT generates a 96-clock period pulse on the RST pin. In order to properly propagate this pulse to the rest of the application in case of external capacitor or power-supply supervisor circuit, a 1 kΩ resistor must be added as shown Figure 18.

**Figure 18.** Recommended Reset Output Schematic



# Power Management

## Introduction

Two power reduction modes are implemented in the AT89C51CC03. The Idle mode and the Power-Down mode. These modes are detailed in the following sections. In addition to these power reduction modes, the clocks of the core and peripherals can be dynamically divided by 2 using the X2 mode detailed in Section “Clock”, page 17.

## Idle Mode

Idle mode is a power reduction mode that reduces the power consumption. In this mode, program execution halts. Idle mode freezes the clock to the CPU at known states while the peripherals continue to be clocked. The CPU status before entering Idle mode is preserved, i.e., the program counter and program status word register retain their data for the duration of Idle mode. The contents of the SFRs and RAM are also retained. The status of the Port pins during Idle mode is detailed in Table 9.

### Entering Idle Mode

To enter Idle mode, set the IDL bit in PCON register (see Table 10). The AT89C51CC03 enters Idle mode upon execution of the instruction that sets IDL bit. The instruction that sets IDL bit is the last instruction executed.

Note: If IDL bit and PD bit are set simultaneously, the AT89C51CC03 enters Power-Down mode. Then it does not go in Idle mode when exiting Power-Down mode.

### Exiting Idle Mode

There are two ways to exit Idle mode:

1. Generate an enabled interrupt.
   - Hardware clears IDL bit in PCON register which restores the clock to the CPU. Execution resumes with the interrupt service routine. Upon completion of the interrupt service routine, program execution resumes with the instruction immediately following the instruction that activated Idle mode. The general purpose flags (GF1 and GF0 in PCON register) may be used to indicate whether an interrupt occurred during normal operation or during Idle mode. When Idle mode is exited by an interrupt, the interrupt service routine may examine GF1 and GF0.
2. Generate a reset.
   - A logic high on the RST pin clears IDL bit in PCON register directly and asynchronously. This restores the clock to the CPU. Program execution momentarily resumes with the instruction immediately following the instruction that activated the Idle mode and may continue for a number of clock cycles before the internal reset algorithm takes control. Reset initializes the AT89C51CC03 and vectors the CPU to address C:0000h.

Note: During the time that execution resumes, the internal RAM cannot be accessed; however, it is possible for the Port pins to be accessed. To avoid unexpected outputs at the Port pins, the instruction immediately following the instruction that activated Idle mode should not write to a Port pin or to the external RAM.

## Power-Down Mode

The Power-Down mode places the AT89C51CC03 in a very low power state. Power-Down mode stops the oscillator, freezes all clock at known states. The CPU status prior to entering Power-Down mode is preserved, i.e., the program counter, program status word register retain their data for the duration of Power-Down mode. In addition, the SFR and RAM contents are preserved. The status of the Port pins during Power-Down mode is detailed in Table 9.

Note: VCC may be reduced to as low as $V_{RET}$ during Power-Down mode to further reduce power dissipation. Take care, however, that VDD is not reduced until Power-Down mode is invoked.

**Entering Power-Down Mode**

To enter Power-Down mode, set PD bit in PCON register. The AT89C51CC03 enters the Power-Down mode upon execution of the instruction that sets PD bit. The instruction that sets PD bit is the last instruction executed.

**Exiting Power-Down Mode**

Note: If VCC was reduced during the Power-Down mode, do not exit Power-Down mode until VCC is restored to the normal operating level.

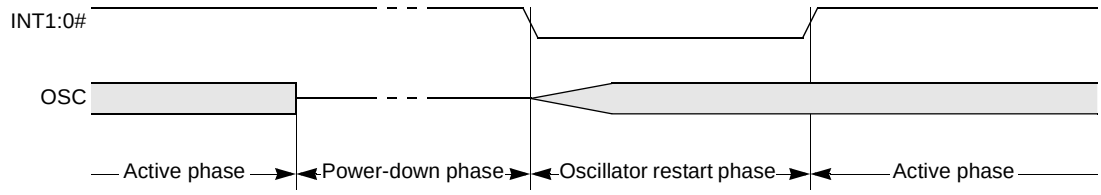
There are two ways to exit the Power-Down mode:

1. Generate an enabled external interrupt.
   – The AT89C51CC03 provides capability to exit from Power-Down using INT0#, INT1#.
   Hardware clears PD bit in PCON register which starts the oscillator and restores the clocks to the CPU and peripherals. Using INTx# input, execution resumes when the input is released (see Figure 19). Execution resumes with the interrupt service routine. Upon completion of the interrupt service routine, program execution resumes with the instruction immediately following the instruction that activated Power-Down mode.

Note: The external interrupt used to exit Power-Down mode must be configured as level sensitive (INT0# and INT1#) and must be assigned the highest priority. In addition, the duration of the interrupt must be long enough to allow the oscillator to stabilize. The execution will only resume when the interrupt is deasserted.

Note: Exit from power-down by external interrupt does not affect the SFRs nor the internal RAM content.

**Figure 19.** Power-Down Exit Waveform Using INT1:0#




2. Generate a reset.
   – A logic high on the RST pin clears PD bit in PCON register directly and asynchronously. This starts the oscillator and restores the clock to the CPU and peripherals. Program execution momentarily resumes with the instruction immediately following the instruction that activated Power-Down mode and may continue for a number of clock cycles before the internal reset algorithm takes control. Reset initializes the AT89C51CC03 and vectors the CPU to address 0000h.

Note: During the time that execution resumes, the internal RAM cannot be accessed; however, it is possible for the Port pins to be accessed. To avoid unexpected outputs at the Port pins, the instruction immediately following the instruction that activated the Power-Down mode should not write to a Port pin or to the external RAM.

Note: Exit from power-down by reset redefines all the SFRs, but does not affect the internal RAM content.

**Table 9.** Pin Conditions in Special Operating Modes

| Mode | Port 0 | Port 1 | Port 2 | Port 3 | Port 4 | ALE | PSEN# |
|---|---|---|---|---|---|---|---|
| Reset | Floating | High | High | High | High | High | High |
| Idle (internal code) | Data | Data | Data | Data | Data | High | High |
| Idle (external code) | Floating | Data | Data | Data | Data | High | High |
| Power-Down(internal code) | Data | Data | Data | Data | Data | Low | Low |
| Power-Down (external code) | Floating | Data | Data | Data | Data | Low | Low |

## Registers

**Table 10.** PCON Register

PCON (S87:h) Power configuration Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | GF1 | GF0 | PD | IDL |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-4 | - | **Reserved**<br>The value read from these bits is indeterminate. Do not set these bits. |
| 3 | GF1 | **General Purpose flag 1**<br>One use is to indicate whether an interrupt occurred during normal operation or during Idle mode. |
| 2 | GF0 | **General Purpose flag 0**<br>One use is to indicate whether an interrupt occurred during normal operation or during Idle mode. |
| 1 | PD | **Power-Down Mode bit**<br>Cleared by hardware when an interrupt or reset occurs.<br>Set to activate the Power-Down mode.<br>If IDL and PD are both set, PD takes precedence. |
| 0 | IDL | **Idle Mode bit**<br>Cleared by hardware when an interrupt or reset occurs.<br>Set to activate the Idle mode.<br>If IDL and PD are both set, PD takes precedence. |

Reset Value= XXXX 0000b

## EEPROM Data Memory

The 2-Kbyte on-chip EEPROM memory block is located at addresses 0000h to 07FFh of the XRAM/ERAM memory space and is selected by setting control bits in the EECON register. A read in the EEPROM memory is done with a MOVX instruction.

A physical write in the EEPROM memory is done in two steps: write data in the column latches and transfer of all data latches into an EEPROM memory row (programming).

The number of data written on the page may vary from 1 up to 128 Bytes (the page size). When programming, only the data written in the column latch is programmed and a ninth bit is used to obtain this feature. This provides the capability to program the whole memory by Bytes, by page or by a number of Bytes in a page. Indeed, each ninth bit is set when the writing the corresponding byte in a row and all these ninth bits are reset after the writing of the complete EEPROM row.

### Write Data in the Column Latches

Data is written by byte to the column latches as for an external RAM memory. Out of the 11 address bits of the data pointer, the 4 MSBs are used for page selection (row) and 7 are used for byte selection. Between two EEPROM programming sessions, all the addresses in the column latches must stay on the same page, meaning that the 4 MSB must no be changed.

The following procedure is used to write to the column latches:

- Save and disable interrupt.
- Set bit EEE of EECON register
- Load DPTR with the address to write
- Store A register with the data to be written
- Execute a MOVX @DPTR, A
- If needed loop the three last instructions until the end of a 128 Bytes page
- Restore interrupt.

Note: The last page address used when loading the column latch is the one used to select the page programming address.

### Programming

The EEPROM programming consists of the following actions:

- writing one or more Bytes of one page in the column latches. Normally, all Bytes must belong to the same page; if not, the first page address will be latched and the others discarded.
- launching programming by writing the control sequence (50h followed by A0h) to the EECON register.
- EEBUSY flag in EECON is then set by hardware to indicate that programming is in progress and that the EEPROM segment is not available for reading.
- The end of programming is indicated by a hardware clear of the EEBUSY flag.

Note: The sequence 5xh and Axh must be executed without instructions between then otherwise the programming is aborted.

### Read Data

The following procedure is used to read the data stored in the EEPROM memory:

- Save and disable interrupt
- Set bit EEE of EECON register
- Load DPTR with the address to read
- Execute a MOVX A, @DPTR
- Restore interrupt

## Examples

```
;*F*****************************************************************************;* NAME: api_rd_eeprom_byte
;* DPTR contain address to read.
;* Acc contain the reading value
;* NOTE: before execute this function, be sure the EEPROM is not BUSY
;*****************************************************************************
api_rd_eeprom_byte:
 MOV   EECON, #02h; map EEPROM in XRAM space
 MOVX A, @DPTR
 MOV   EECON, #00h; unmap EEPROM
ret

;*F*****************************************************************************
;* NAME: api_ld_eeprom_cl
;* DPTR contain address to load
;* Acc contain value to load
;* NOTE: in this example we load only 1 byte, but it is possible upto
;* 128 Bytes.
;* before execute this function, be sure the EEPROM is not BUSY
;*****************************************************************************
api_ld_eeprom_cl:
 MOV   EECON, #02h ; map EEPROM in XRAM space
 MOVX @DPTR, A
 MOVEECON, #00h; unmap EEPROM
ret

;*F*****************************************************************************
;* NAME: api_wr_eeprom
;* NOTE: before execute this function, be sure the EEPROM is not BUSY
;*****************************************************************************
api_wr_eeprom:
 MOV   EECON, #050h
 MOV   EECON, #0A0h
ret
```

## Registers

**Table 11.** EECON Register

EECON (S:0D2h)
EEPROM Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **EEPL3** | **EEPL2** | **EEPL1** | **EEPL0** | **-** | **-** | **EEE** | **EEBUSY** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-4 | EEPL3-0 | **Programming Launch command bits**<br>Write 5Xh followed by AXh to EEPL to launch the programming. |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 2 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 1 | EEE | **Enable EEPROM Space bit**<br>Set to map the EEPROM space during MOVX instructions (Write in the column latches)<br>Clear to map the XRAM space during MOVX. |
| 0 | EEBUSY | **Programming Busy flag**<br>Set by hardware when programming is in progress.<br>Cleared by hardware when programming is done.<br>Can not be set or cleared by software. |

Reset Value = XXXX XX00b
Not bit addressable

## Program/Code Memory

The AT89C51CC03 implement 64K Bytes of on-chip program/code memory. Figure 20 shows the partitioning of internal and external program/code memory spaces depending on the product.

The Flash memory increases EPROM and ROM functionality by in-circuit electrical erasure and programming. Thanks to the internal charge pump, the high voltage needed for programming or erasing Flash cells is generated on-chip using the standard VDD voltage. Thus, the Flash Memory can be programmed using only one voltage and allows In-System Programming commonly known as ISP. Hardware programming mode is also available using specific programming tool.

**Figure 20.** Program/Code Memory Organization

## External Code Memory Access

### Memory Interface

The external memory interface comprises the external bus (port 0 and port 2) as well as the bus control signals (PSEN#, and ALE).

Figure 21 shows the structure of the external address bus. P0 carries address A7:0 while P2 carries address A15:8. Data D7:0 is multiplexed with A7:0 on P0. Table 21 describes the external memory interface signals.

**Figure 21.** External Code Memory Interface Structure

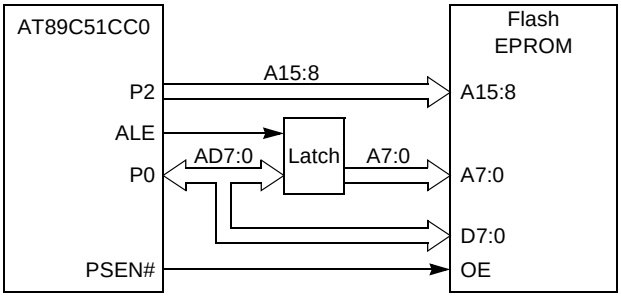



**Table 12.** External Code Memory Interface Signals

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| A15:8 | O | **Address Lines**<br>Upper address lines for the external bus. | P2.7:0 |
| AD7:0 | I/O | **Address/Data Lines**<br>Multiplexed lower address lines and data for the external memory. | P0.7:0 |
| ALE | O | **Address Latch Enable**<br>ALE signals indicates that valid address information are available on lines AD7:0. | - |
| PSEN# | O | **Program Store Enable Output**<br>This signal is active low during external code fetch or external code read (MOVC instruction). | - |

### External Bus Cycles

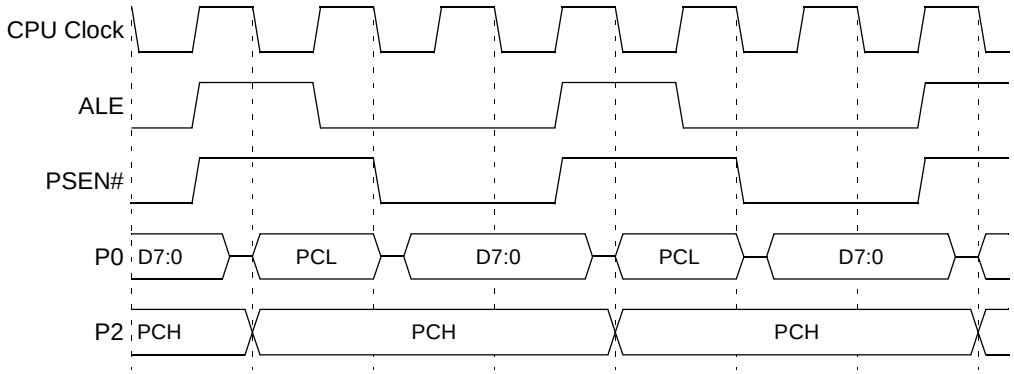
This section describes the bus cycles the AT89C51CC03 executes to fetch code (see Figure 22) in the external program/code memory.

External memory cycle takes 6 CPU clock periods. This is equivalent to 12 oscillator clock period in standard mode or 6 oscillator clock periods in X2 mode. For further information on X2 mode see section “Clock “.

For simplicity, the accompanying figure depicts the bus cycle waveforms in idealized form and do not provide precise timing information.

For bus cycling parameters refer to the ‘AC-DC parameters’ section.

**Figure 22.** External Code Fetch Waveforms




## Flash Memory Architecture

AT89C51CC03 features two on-chip Flash memories:

- Flash memory FM0:
  containing 64K Bytes of program memory (user space) organized into 128 byte pages,
- Flash memory FM1:
  2K Bytes for boot loader and Application Programming Interfaces (API).

The FM0 can be program by both parallel programming and Serial In-System Programming (ISP) whereas FM1 supports only parallel programming by programmers. The ISP mode is detailed in the "In-System Programming" section.

All Read/Write access operations on Flash Memory by user application are managed by a set of API described in the "In-System Programming" section.

The bit ENBOOT in AUXR1 register is used to map FM1 from F800h to FFFFh. Figure 23 and Figure 24 show the Flash memory configuration with ENBOOT=1 and ENBOOT=0.

**Figure 23.** Flash Memory Architecture with ENBOOT=1 (boot mode)

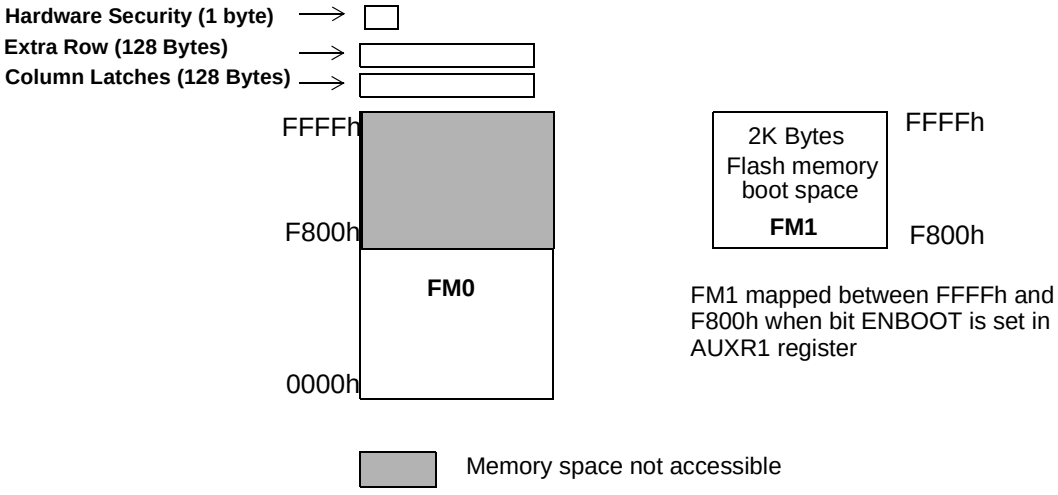
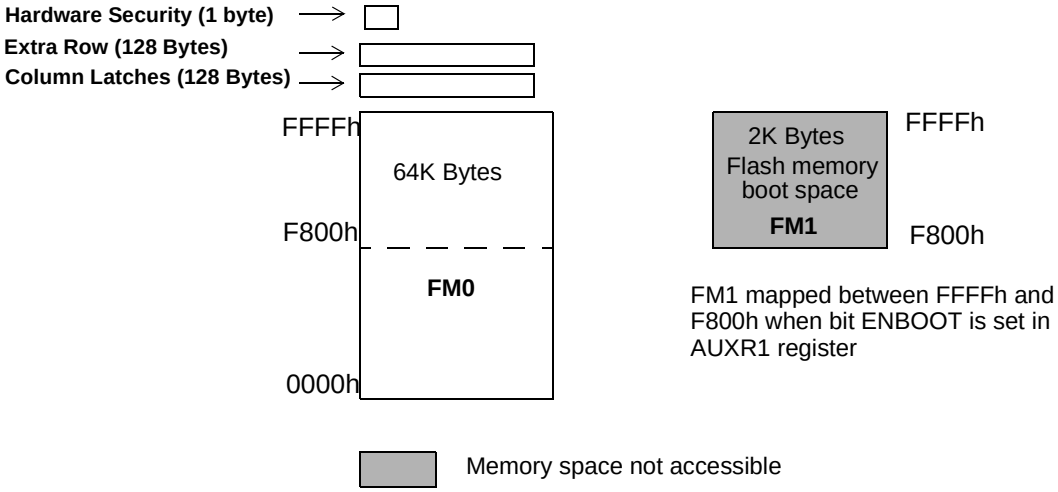


**Figure 24.** Flash Memory Architecture with ENBOOT=0 (user modemode)



**FM0 Memory Architecture**

The Flash memory is made up of 4 blocks (see Figure 23):

- The memory array (user space) 64K Bytes
- The Extra Row
- The Hardware security bits
- The column latch registers

*User Space*

This space is composed of a 64K Bytes Flash memory organized in 512 pages of 128 Bytes. It contains the user's application code.

*Extra Row (XRow)*

This row is a part of FM0 and has a size of 128 Bytes. The extra row may contain information for boot loader usage.

*Hardware security Byte (HSB)*

The Hardware security Byte space is a part of FM0 and has a size of 1 byte.
The 4 MSB can be read/written by software (from FM0 and , the 4 LSB can only be read by software and written by hardware in parallel mode.

H Hardware Security Byte (HSB)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **X2** | **BLJB** | **-** | **-** | **-** | **LB2** | **LB1** | **LB0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | X2 | **X2 Mode**<br>Programmed (='0') to force X2 mode (6 clocks per instruction) after reset<br>Unprogrammed to force X1 mode, Standard Mode, afetr reset (Default) |
| 6 | BLJB | **Boot Loader Jump Bit**<br>When unprogrammed (='1'), at the next reset :<br>-ENBOOT=0 (see code space memory configuration)<br>-Start address is 0000h (PC=0000h)<br>When programmed (='0')at the nex reset:<br>-ENBOOT=1 (see code space memory configuration)<br>-Start address is F800h (PC=F800h) |
| 5 | - | **Reserved** |
| 4 | - | **Reserved** |
| 3 | - | **Reserved** |
| 2-0 | LB2-0 | **General Memory Lock Bits (only programmable by programmer tools)**<br>Section "Flash Protection from Parallel Programming", page 53 |

*Column Latches*

The column latches, also part of FM0, have a size of full page (128 Bytes).
The column latches are the entrance buffers of the three previous memory locations (user array, XROW and Hardware security byte). The column latches are write only and can be accessed only from FM1 (boot mode) and from external memory

**Cross Flash Memory Access Description**

The FM0 memory can be program only from FM1. Programming FM0 from FM0 or from external memory is impossible.

The FM1 memory can be program only by parallel programming.

The Table  show all software Flash access allowed.

Cross Flash Memory Access

| Code executing from | | Action | FM0 (user Flash) | FM1 (boot Flash) |
|---|---|---|---|---|
| | FM0 (user Flash) | Read | ok | - |
| | | Load column latch | ok | - |
| | | Write | - | - |
| | FM1 (boot Flash) | Read | ok | ok |
| | | Load column latch | ok | - |
| | | Write | ok | - |
| | External memory EA = 0 | Read | (a) | - |
| | | Load column latch | - | - |
| | | Write | - | - |

(a) Depend upon general lock bit configuration.

## Overview of FM0 Operations

### Flash Registers (SFR)

The CPU interfaces to the flash memory through the FCON register, AUXR1 register and FSTA register.

These registers are used to map the column latches, HSB, extra row and EEDATA in the working data or code space.

*FCON Register*

**Table 13.** FCON Register

FCON Register (S:D1h)
Flash Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **FPL3** | **FPL2** | **FPL1** | **FPL0** | **FPS** | **FMOD1** | **FMOD0** | **FBUSY** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-4 | FPL3:0 | **Programming Launch Command Bits**<br>Write 5Xh followed by AXh to launch the programming according to FMOD1:0. (see Table 16.) |
| 3 | FPS | **Flash Map Program Space**<br>When this bit is set:<br>The MOVX @DPTR, A instruction writes in the columns latches space<br>When this bit is cleared:<br>The MOVX @DPTR, A instruction writes in the regular XDATA memory space |
| 2-1 | FMOD1:0 | **Flash Mode**<br>See Table 16. |
| 0 | FBUSY | **Flash Busy**<br>Set by hardware when programming is in progress.<br>Clear by hardware when programming is done.<br>Can not be changed by software. |

Reset Value= 0000 0000b

*FSTA Register*

**Table 14.** FSTA Register

FSTA Register (S:D3h)
Flash Status Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | **SEQERR** | **FLOAD** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-2 | | **unusesd** |
| 1 | SEQERR | **Flash activation sequence error**<br>Set by hardware when the flash activation sequence(MOV FCON 5X and MOV FCON AX )is not correct (See Error Repport Section)<br>Clear by software or clear by hardware if the last activation sequence was correct (previous error are canceled) |
| 0 | FLOAD | **Flash Colums latch loaded**<br>Set by hardware when the first data is loaded in the column latches.<br>Clear by hardware when the activation sequence suceed (flash write sucess, or reset column latch success) |

Reset Value= 0000 0000b

**Mapping of the Memory Space**

By default, the user space is accessed by MOVC A, @DPTR instruction for read only. The column latches space is made accessible by setting the FPS bit in FCON register. Writing is possible from 0000h to FFFFh, address bits 6 to 0 are used to select an address within a page while bits 15 to 7 are used to select the programming address of the page.
Setting FPS bit takes precedence on the EXTRAM bit in AUXR register.

The other memory spaces (user, extra row, hardware security) are made accessible in the code segment by programming bits FMOD0 and FMOD1 in FCON register in accordance with Table 15. A MOVC instruction is then used for reading these spaces.

**Table 15.** FM0 Blocks Select Bits

| FMOD1 | FMOD0 | FM0 Adressable space |
|---|---|---|
| 0 | 0 | User (0000h-FFFFh) |
| 0 | 1 | Extra Row(FF80h-FFFFh) |
| 1 | 0 | Hardware Security Byte (0000h) |
| 1 | 1 | Column latches reset (note1) |

Notes: 1. The column latches reset is a new option introduced in the AT89C51CC03, and is not available in T89C51CC01/2

**Launching Programming**

FPL3:0 bits in FCON register are used to secure the launch of programming. A specific sequence must be written in these bits to unlock the write protection and to launch the programming. This sequence is 5xh followed by Axh. Table 16 summarizes the memory spaces to program according to FMOD1:0 bits.

**Table 16.** Programming Spaces

| | Write to FCON | | | | |
|---|---|---|---|---|---|
| | **FPL3:0** | **FPS** | **FMOD1** | **FMOD0** | **Operation** |
| **User** | 5 | X | 0 | 0 | No action |
| | A | X | 0 | 0 | Write the column latches in user space |
| **Extra Row** | 5 | X | 0 | 1 | No action |
| | A | X | 0 | 1 | Write the column latches in extra row space |
| **Hardware Security Byte** | 5 | X | 1 | 0 | No action |
| | A | X | 1 | 0 | Write the fuse bits space |
| **Reset Columns Latches** | 5 | X | 1 | 1 | No action |
| | A | X | 1 | 1 | Reset the column latches |

Notes:
1. The sequence 5xh and Axh must be executing without instructions between them otherwise the programming is not executed (see Flash Status Register)
2. The sequence 5xh and Axh must be executed with the same FMOD0 FMOD1 configuration.
3. Interrupts that may occur during programming time must be disabled to avoid any spurious exit of the programming mode.

**Status of the Flash Memory**

The bit FBUSY in FCON register is used to indicate the status of programming.

FBUSY is set when programming is in progress.

The flash programming process is launched the second machine cycle following the sequence 5xh and Axh in FCON. Thus the FBUSY flag should be read by sofware not during the insctruction after the 5xh, Axh sequence but the the second instruction after the 5xh, Axh sequence in FCON (See next example). FBUSY is cleared when the programming is completed.

```
;*F*************************************************************************
;* NAME: launch_prog
;;*************************************************************************
launch_prog:
        MOV   FCON, #050h
        MOV   FCON #0A0h  ; Flash Write Sequence
        NOP               ;Required time before reading busy flag
wait_busy:
        MOV A,FCON
        JB ACC.0,wait_busy
        RET
```

**Selecting FM1**

The bit ENBOOT in AUXR1 register is used to map FM1 from F800h to FFFFh.

**Loading the Column Latches**

Any number of data from 1-byte to 128 Bytes can be loaded in the column latches. This provides the capability to program the whole memory by byte, by page or by any number of Bytes in a page. Data written in the column latches do not have to be in consecutive

order. The page address of the last address loaded in the column latches will be used for the whole page.

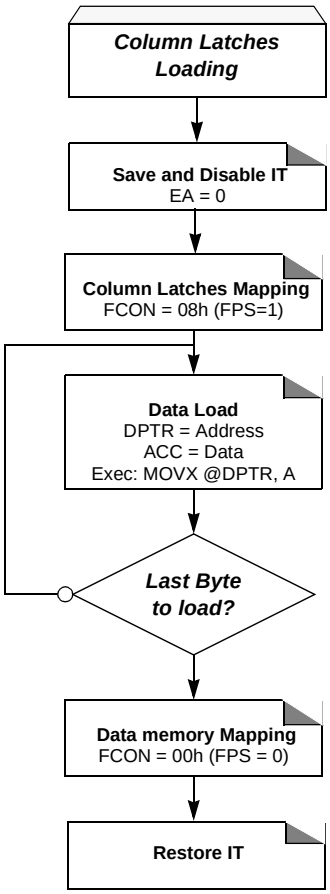
When programming is launched, an automatic erase of the locations loaded in the column latches is first performed, then programming is effectively done. Thus no page or block erase is needed and only the loaded data are programmed in the corresponding page

Notes:
1. : If no bytes are written in the column latches the SEQERR bit in the FSTA register will be set.
2. When a flash write sequence is in progress (FBUSY is set) a write sequence to the column latches will be ignored and the content of the column latches at the time of the launch write sequence will be preserved.
3. MOVX @DPTR, A instruction must be used to load the column latches. Never use MOVX @Ri, A instructions.
4. When a programming sequence is launched, Flash bytes corresponding to activated bytes in the column latches are first erased then the bytes in the column latches are copied into the Flash bytes. Flash bytes corresponding to bytes in the column latches not activated (not loaded during the load column latches sequence) will not be erased and written.

The following procedure is used to load the column latches and is summarized in Figure 25:

- Save and Disable interrupt and map the column latch space by setting FPS bit.
- Load the DPTR with the address to load.
- Load Accumulator register with the data to load.
- Execute the MOVX @DPTR, A instruction.
- If needed loop the three last instructions until the page is completely loaded.
- unmap the column latch.
- Restore Interrupt

**Figure 25.** Column Latches Loading Procedure




Note: The last page address used when loading the column latch is the one used to select the page programming address.

### Programming the Flash Spaces

*User*

The following procedure is used to program the User space and is summarized in Figure 26:

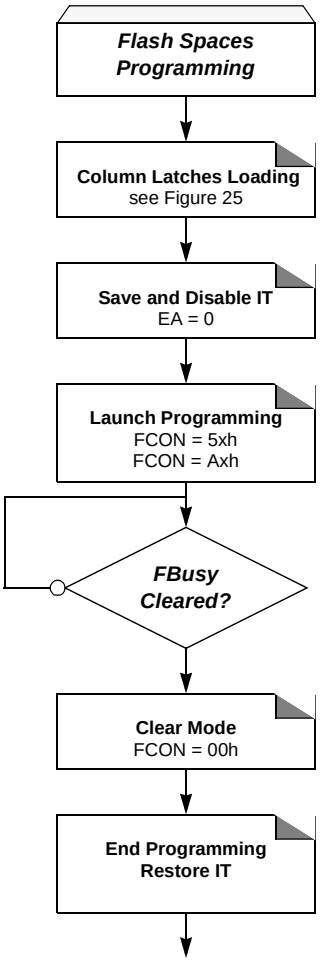
- Load up to one page of data in the column latches from address 0000h to FFFFh.
- Save and Disable the interrupts.
- Launch the programming by writing the data sequence 50h followed by A0h in FCON register (only from FM1).
  The end of the programming indicated by the FBUSY flag cleared.
- Restore the interrupts.

*Extra Row*

The following procedure is used to program the Extra Row space and is summarized in Figure 26:

- Load data in the column latches from address FF80h to FFFFh.
- Save and Disable the interrupts.
- Launch the programming by writing the data sequence 52h followed by A2h in FCON register (only from FM1).
  The end of the programming indicated by the FBUSY flag cleared.
- Restore the interrupts.

**Figure 26.** Flash and Extra Row Programming Procedure

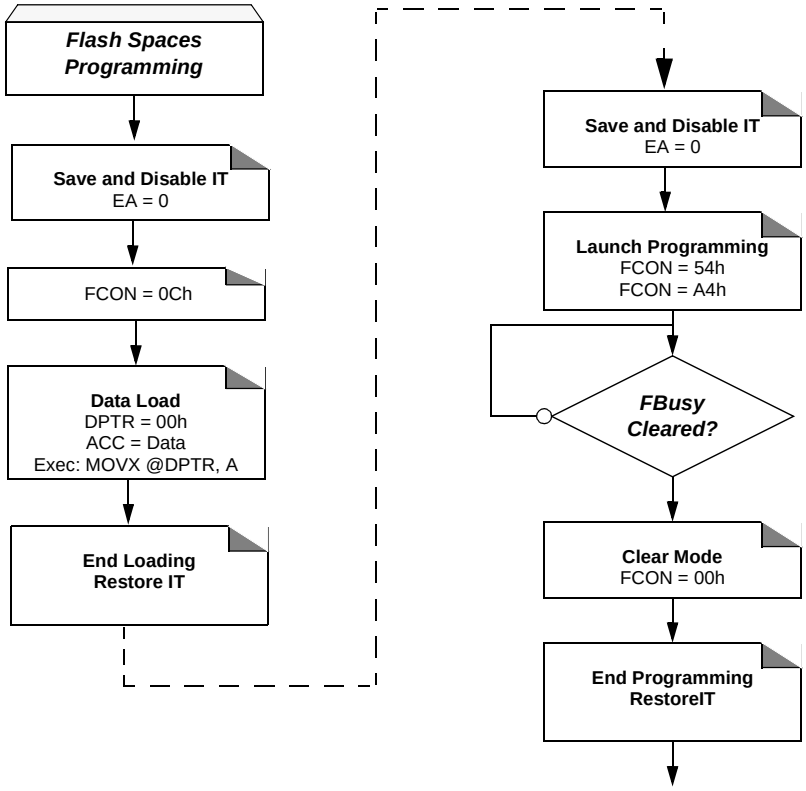



*Hardware Security Byte*

The following procedure is used to program the Hardware Security Byte space and is summarized in Figure 27:

- Set FPS and map Hardware byte (FCON = 0x0C)
- Save and disable the interrupts.
- Load DPTR at address 0000h.
- Load Accumulator register with the data to load.
- Execute the MOVX @DPTR, A instruction.
- Launch the programming by writing the data sequence 54h followed by A4h in FCON register (only from FM1).
  The end of the programming indicated by the FBusy flag cleared.
- Restore the interrupts.

**Figure 27.** Hardware Programming Procedure




*Reset the Column Latches*

An automatic reset of the column latches is performed after a successful Flash write sequence. User can also reset the column latches manually, for instance to reload the column latches before writing the Flash. The following procedure is summarized below.

- Save and disable the interrupts.
- Launch the reset by writing the data sequence 56h followed by A6h in FCON register (only from FM1).
- Restore the interrupts.

**Error Reports**

*Flash Programming Sequence Errors*

When a wrong sequence is detected, the SEQERR bit in FSTA register is set. Possible wrong sequence are :

- MOV FCON, 5xh instruction not immediately followed by a MOV FCON, Ax instruction.
- A write Flash sequence is launched while no data were loaded in the column latches

The SEQERR bit can be cleared

- By software
- By hardware when a correct programming sequence is completed

When multiple pages are written into the Flash, the user should check FSTA for errors after each write page sequences, not only at the end of the multiple write pages.

*Power Down Request*

Before entering in Power Down (Set bit PD in PCON register) the user should check that no write sequence is in progress (check BUSY=0), then check that the column latches are reset (FLOAD=0 in FSTA register. Launch a reset column latches to clear FLOAD if necessary.

**Reading the Flash Spaces**

*User*

The following procedure is used to read the User space:

- Read one byte in Accumulator by executing MOVC A,@A+DPTR with A+DPTR=read@.

Note: FCON is supposed to be reset when not needed.

*Extra Row*

The following procedure is used to read the Extra Row space and is summarized in Figure 28:

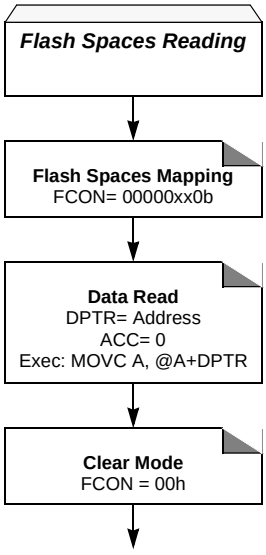
- Map the Extra Row space by writing 02h in FCON register.
- Read one byte in Accumulator by executing MOVC A,@A+DPTR with A = 0 and DPTR = FF80h to FFFFh.
- Clear FCON to unmap the Extra Row.

*Hardware Security Byte*

The following procedure is used to read the Hardware Security space and is summarized in Figure 28:

- Map the Hardware Security space by writing 04h in FCON register.
- Read the byte in Accumulator by executing MOVC A,@A+DPTR with A = 0 and DPTR = 0000h.

**Figure 28.** Clear FCON to unmap the Hardware Security Byte.Reading Procedure




**Flash Protection from Parallel Programming**

The three lock bits in Hardware Security Byte (see "In-System Programming" section) are programmed according to Table 17 provide different level of protection for the on-chip code and data located in FM0 and FM1.

The only way to write this bits are the parallel mode. They are set by default to level 4

**Table 17.** Program Lock Bit

| Program Lock Bits | | | | |
|---|---|---|---|---|
| Security level | LB0 | LB1 | LB2 | **Protection Description** |
| 1 | U | U | U | No program lock features enabled. |
| 2 | P | U | U | MOVC instruction executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further parallel programming of the Flash is disabled.<br>ISP and software programming with API are still allowed.<br>Writing EEprom Data from external parallel programmer is disabled but still allowed from internal code execution. |
| 3 | U | P | U | Same as 2, also verify through parallel programming interface is disabled.<br>Writing And Reading EEPROM Data from external parallel programmer is disabled but still allowed from internal code execution.. |
| 4 | U | U | P | Same as 3, also external execution is disabled |

Program Lock bits

U: unprogrammed

P: programmed

WARNING: Security level 2 and 3 should only be programmed after Flash and Core verification.

## Operation Cross Memory Access

Space addressable in read and write are:

- RAM
- ERAM (Expanded RAM access by movx)
- XRAM (eXternal RAM)
- EEPROM DATA
- FM0 ( user flash )
- Hardware byte
- XROW
- Boot Flash
- Flash Column latch

The table below provide the different kind of memory which can be accessed from different code location.

**Table 18.** Cross Memory Access

| | Action | RAM | XRAM ERAM | Boot FLASH | FM0 | E² Data | Hardware Byte | XROW |
|---|---|---|---|---|---|---|---|---|
| boot FLASH | Read | | | OK | OK | OK | OK | - |
| | Write | | | - | OK(1) | OK(1) | OK(1) | OK(1) |
| FM0 | Read | | | OK | OK | OK | OK | - |
| | Write | | | - | OK (idle) | OK(1) | - | OK |
| External memory EA = 0 or Code Roll Over | Read | | | - | - | OK | - | - |
| | Write | | | - | - | OK(1) | - | - |

Note: 1. RWW: Read While Write

## Sharing Instructions

**Table 19.** Instructions shared

| Action | RAM | XRAM ERAM | EEPROM DATA | Boot FLASH | FM0 | Hardware Byte | XROW |
|---|---|---|---|---|---|---|---|
| Read | MOV | MOVX | MOVX | MOVC | MOVC | MOVC | MOVC |
| Write | MOV | MOVX | MOVX | - | by cl | by cl | by cl |

Note: by cl : using Column Latch

**Table 20.** Read MOVX A, @DPTR

| EEE bit in EECON Register | FPS in FCON Register | ENBOOT | EA | XRAM ERAM | EEPROM DATA | Flash Column Latch |
|---|---|---|---|---|---|---|
| 0 | 0 | X | X | OK | | |
| 0 | 1 | X | X | OK | | |
| 1 | 0 | X | X | | OK | |
| 1 | 1 | X | X | OK | | |

**Table 21.** Write MOVX @DPTR,A

| EEE bit in EECON Register | FPS bit in FCON Register | ENBOOT | EA | XRAM ERAM | EEPROM Data | Flash Column Latch |
|---|---|---|---|---|---|---|
| 0 | 0 | X | X | OK | | |
| 0 | 1 | X | 1 | | | OK |
| | | | 0 | OK | | |
| 1 | 0 | X | X | | OK | |
| 1 | 1 | X | 1 | | | OK |
| | | | 0 | OK | | |

**Table 22.** Read MOVC A, @DPTR

| Code Execution | FCON Register | | | ENBOOT | DPTR | FM1 | FM0 | XROW | Hardware Byte | External Code |
|---|---|---|---|---|---|---|---|---|---|---|
| | FMOD1 | FMOD0 | FPS | | | | | | | |
| From FM0 | 0 | 0 | X | 0 | 0000h to FFFFh | | OK | | | |
| | | | | 1 | 0000h to F7FF | | OK | | | |
| | | | | | F800h to FFFFh | Do not use this configuration | | | | |
| | 0 | 1 | X | X | 0000 to 007Fh See (1) | | | OK | | |
| | 1 | 0 | X | X | X | | | | OK | |
| | 1 | 1 | X | 0 | 000h to FFFFh | | OK | | | |
| | | | | 1 | 0000h to F7FF | | OK | | | |
| | | | | | F800h to FFFFh | Do not use this configuration | | | | |
| From FM1 (ENBOOT =1 | 0 | 0 | 0 | 1 | 0000h to F7FF | | OK | | | |
| | | | | | F800h to FFFFh | OK | | | | |
| | | | | 0 | X | NA | | | | |
| | | | 1 | 1 | X | | OK | | | |
| | | | | 0 | X | NA | | | | |
| | 0 | 1 | X | 1 | 0000h to 007h See (2) | | | OK | | |
| | | | | 0 | | NA | | | | |
| | 1 | 0 | X | 1 | X | | | | OK | |
| | | | | 0 | | NA | | | | |
| | 1 | 1 | X | 1 | 000h to FFFFh | | OK | | | |
| | | | | 0 | | NA | | | | |
| External code : EA=0 or Code Roll Over | X | 0 | X | X | X | | | | | OK |

1. For DPTR higher than 007Fh only lowest 7 bits are decoded, thus the behavior is the same as for addresses from 0000h to 007Fh
2. For DPTR higher than 007Fh only lowest 7 bits are decoded, thus the behavior is the same as for addresses from 0000h to 007Fh

## In-System Programming (ISP)

With the implementation of the User Space (FM0) and the Boot Space (FM1) in Flash technology the AT89C51CC03 allows the system engineer the development of applications with a very high level of flexibility. This flexibility is based on the possibility to alter the customer program at any stages of a product's life:

- Before assembly the 1st personalization of the product by programming in the FM0 and if needed also a customized Boot loader in the FM1.
  Atmel provide also a standard Boot loader by default UART or CAN.
- After assembling on the PCB in its final embedded position by serial mode via the CAN bus or UART.

This In-System Programming (ISP) allows code modification over the total lifetime of the product.

Besides the default Boot loader Atmel provide to the customer also all the needed Application-Programming-Interfaces (API) which are needed for the ISP. The API are located also in the Boot memory.

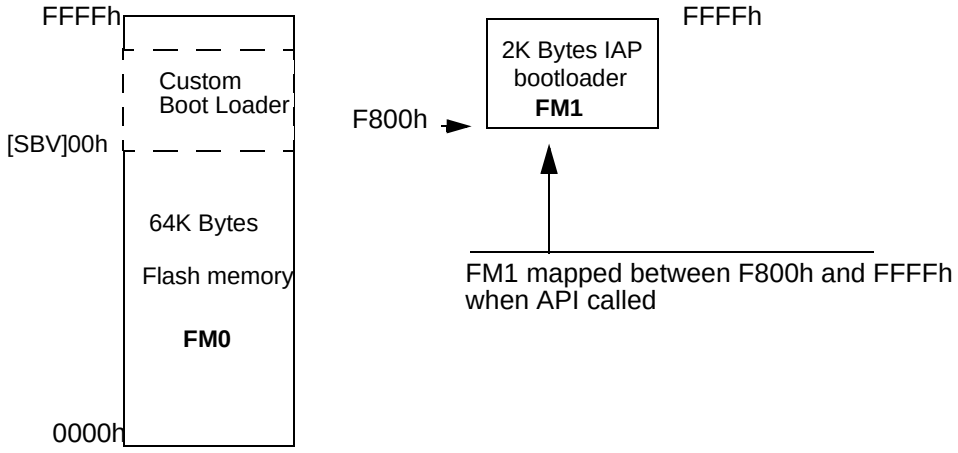
This allow the customer to have a full use of the 64-Kbyte user memory.

## Flash Programming and Erasure

There are three methods of programming the Flash memory:

- The Atmel bootloader located in FM1 is activated by the application. Low level API routines (located in FM1)will be used to program FM0. The interface used for serial downloading to FM0 is the UART or the CAN. API can be called also by the user's bootloader located in FM0 at [SBV]00h.
- A further method exists in activating the Atmel boot loader by hardware activation.
- The FM0 can be programmed also by the parallel mode using a programmer.

**Figure 29.** Flash Memory Mapping




## Boot Process

### Software Boot Process Example

Many algorithms can be used for the software boot process. Before describing them,

The description of the different flags and Bytes is given below:

Boot Loader Jump Bit (BLJB):
- This bit indicates if on RESET the user wants to jump to this application at address @0000h on FM0 or execute the boot loader at address @F800h on FM1.
- BLJB = 0 on parts delivered with bootloader programmed.
- To read or modify this bit, the APIs are used.

Boot Vector Address (SBV):
- This byte contains the MSB of the user boot loader address in FM0.
- The default value of SBV is FCh (no user boot loader in FM0).
- To read or modify this byte, the APIs are used.

Extra Byte (EB) and Boot Status Byte (BSB):
- These Bytes are reserved for customer use.
- To read or modify these Bytes, the APIs are used.

**Hardware Boot Process**

At the falling edge of RESET, the bit ENBOOT in AUXR1 register is initialized with the value of Boot Loader Jump Bit (BLJB).

Further at the falling edge of RESET if the following conditions (called Hardware condition) are detected:

- PSEN low,
- EA high,
- ALE high (or not connected).
  - After Hardware Condition the FCON register is initialized with the value 00h and the PC is initialized with F800h (FM1).

The Hardware condition makes the bootloader to be executed, whatever BLJB value is.

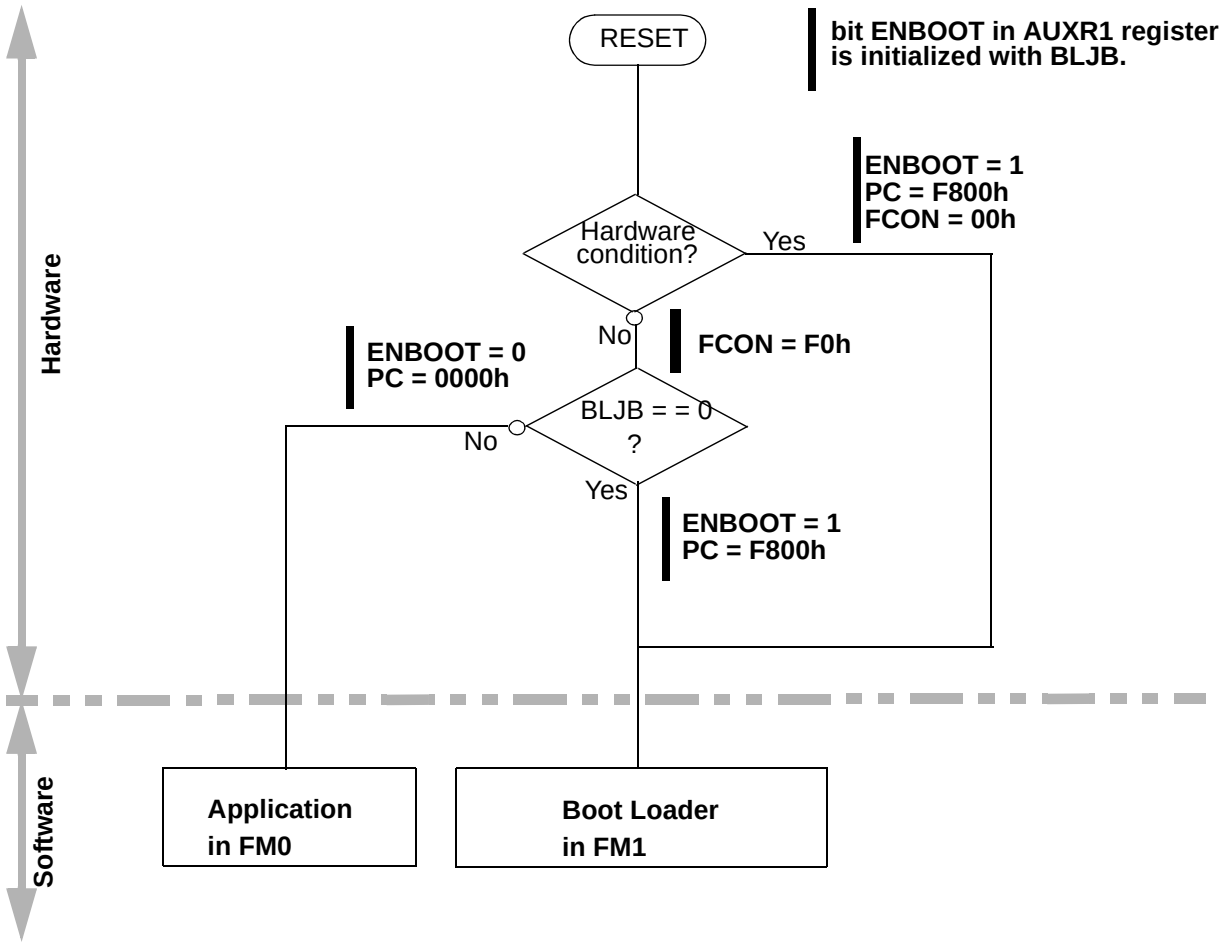
If no hardware condition is detected, the FCON register is initialized with the value F0h.

Check of the BLJB value.

- If bit BLJB = 1:
  User application in FM0 will be started at @0000h (standard reset).
- If bit BLJB = 0:
  Boot loader will be started at @F800h in FM1.

Note:
1. As PSEN is an output port in normal operating mode (running user applications or bootloader applications) after reset it is recommended to release PSEN after the falling edge of Reset is signaled.
   The hardware conditions are sampled at reset signal Falling Edge, thus they can be released at any time when reset input is low.
2. To ensure correct microcontroller startup, the PSEN pin should not be tied to ground during power-on.

**Figure 30.** Hardware Boot Process Algorithm




## Application Programming Interface

Several Application Program Interface (API) calls are available for use by an application program to permit selective erasing and programming of Flash pages. All calls are made by functions.

All these APIs are describe in an documentation: "In-System Programing: Flash Library for AT89C51CC03" available on the Atmel web site.

## XROW Bytes

**Table 23.** XROW Mapping

| Description | Default Value | Address |
|---|---|---|
| Copy of the Manufacturer Code | 58h | 30h |
| Copy of the Device ID#1: Family code | D7h | 31h |
| Copy of the Device ID#2: Memories size and type | FFh | 60h |
| Copy of the Device ID#3: Name and Revision | FEh | 61h |

## Hardware Security Byte

**Table 24.** Hardware Security Byte

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **X2B** | **BLJB** | **-** | **-** | **-** | **LB2** | **LB1** | **LB0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | X2B | **X2 Bit**<br>Set this bit to start in standard mode<br>Clear this bit to start in X2 mode. |
| 6 | BLJB | **Boot Loader JumpBit**<br>- 1: To start the user's application on next RESET (@0000h) located in FM0,<br>- 0: To start the boot loader(@F800h) located in FM1. |
| 5-3 | - | **Reserved**<br>The value read from these bits are indeterminate. |
| 2-0 | LB2:0 | **Lock Bits** |

After erasing the chip in parallel mode, the default value is : FFh

The erasing in ISP mode (from bootloader) does not modify this byte.

Notes: 1. Only the 4 MSB bits can be accessed by software.
2. The 4 LSB bits can only be accessed by parallel mode.

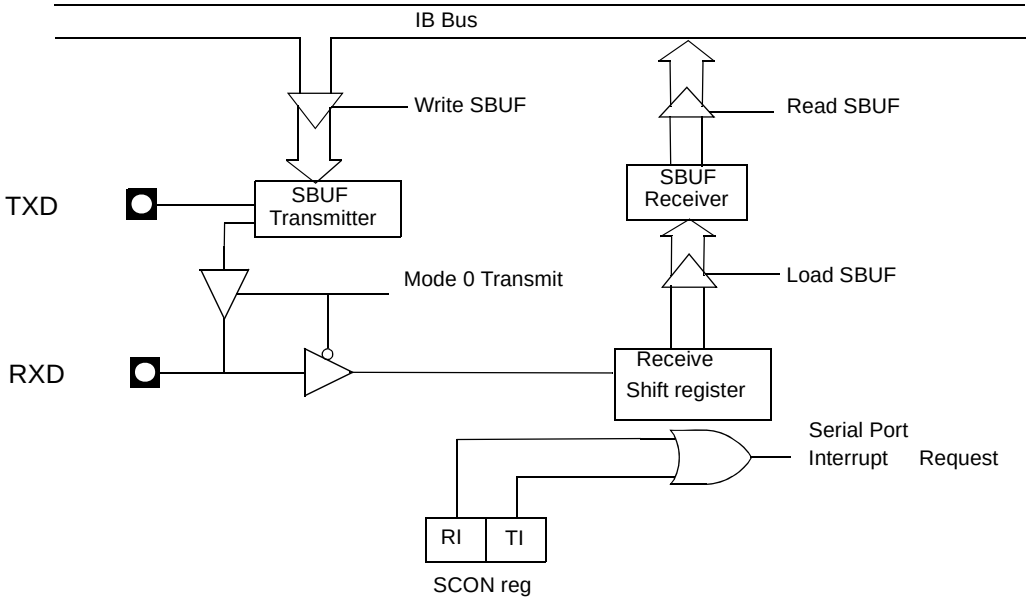
## Serial I/O Port

The AT89C51CC03 I/O serial port is compatible with the I/O serial port in the 80C52. It provides both synchronous and asynchronous communication modes. It operates as a Universal Asynchronous Receiver and Transmitter (UART) in three full-duplex modes (Modes 1, 2 and 3). Asynchronous transmission and reception can occur simultaneously and at different baud rates

Serial I/O port includes the following enhancements:

- Framing error detection
- Automatic address recognition

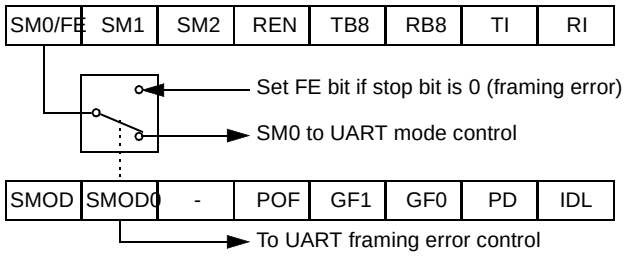
**Figure 31.** Serial I/O Port Block Diagram




### Framing Error Detection

Framing bit error detection is provided for the three asynchronous modes. To enable the framing bit error detection feature, set SMOD0 bit in PCON register.

**Figure 32.** Framing Error Block Diagram




When this feature is enabled, the receiver checks each incoming data frame for a valid stop bit. An invalid stop bit may result from noise on the serial lines or from simultaneous transmission by two CPUs. If a valid stop bit is not found, the Framing Error bit (FE) in SCON register bit is set.

The software may examine the FE bit after each reception to check for data errors. Once set, only software or a reset clears the FE bit. Subsequently received frames with

valid stop bits cannot clear the FE bit. When the FE feature is enabled, RI rises on the stop bit instead of the last data bit (See Figure 33. and Figure 34.).

**Figure 33.** UART Timing in Mode 1

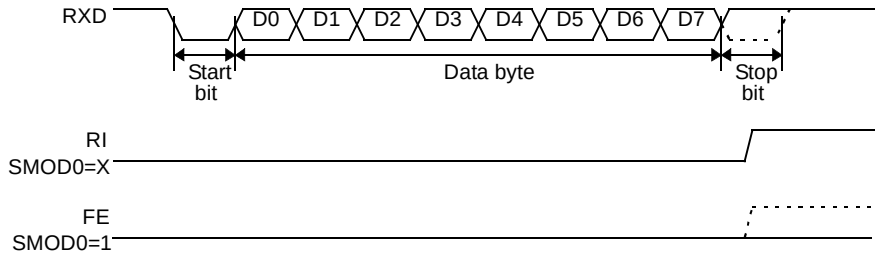
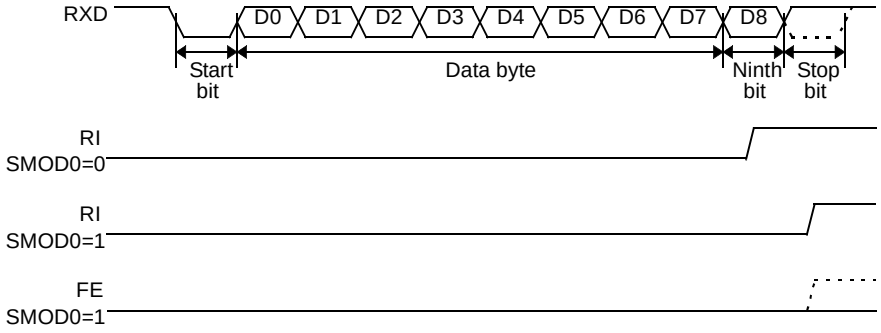



**Figure 34.** UART Timing in Modes 2 and 3




## Automatic Address Recognition

The automatic address recognition feature is enabled when the multiprocessor communication feature is enabled (SM2 bit in SCON register is set).

Implemented in the hardware, automatic address recognition enhances the multiprocessor communication feature by allowing the serial port to examine the address of each incoming command frame. Only when the serial port recognizes its own address will the receiver set the RI bit in the SCON register to generate an interrupt. This ensures that the CPU is not interrupted by command frames addressed to other devices.

If necessary, you can enable the automatic address recognition feature in mode 1. In this configuration, the stop bit takes the place of the ninth data bit. Bit RI is set only when the received command frame address matches the device's address and is terminated by a valid stop bit.

To support automatic address recognition, a device is identified by a given address and a broadcast address.

Note: The multiprocessor communication and automatic address recognition features cannot be enabled in mode 0 (i.e. setting SM2 bit in SCON register in mode 0 has no effect).

## Given Address

Each device has an individual address that is specified in the SADDR register; the SADEN register is a mask byte that contains don't-care bits (defined by zeros) to form the device's given address. The don't-care bits provide the flexibility to address one or more slaves at a time. The following example illustrates how a given address is formed. To address a device by its individual address, the SADEN mask byte must be `1111 1111b`.

For example:

SADDR0101 0110b
SADEN1111 1100b

Given0101 01XXb

Here is an example of how to use given addresses to address different slaves:

Slave A:SADDR1111 0001b
SADEN1111 1010b

Given1111 0X0Xb

Slave B:SADDR1111 0011b
SADEN1111 1001b

Given1111 0XX1b

Slave C:SADDR1111 0011b
SADEN1111 1101b

Given1111 00X1b

The SADEN byte is selected so that each slave may be addressed separately.
For slave A, bit 0 (the LSB) is a don't-care bit; for slaves B and C, bit 0 is a 1. To communicate with slave A only, the master must send an address where bit 0 is clear (e.g. `1111 0000b`).
For slave A, bit 1 is a 0; for slaves B and C, bit 1 is a don't care bit. To communicate with slaves A and B, but not slave C, the master must send an address with bits 0 and 1 both set (e.g. `1111 0011b`).
To communicate with slaves A, B and C, the master must send an address with bit 0 set, bit 1 clear, and bit 2 clear (e.g. `1111 0001b`).

## Broadcast Address

A broadcast address is formed from the logical OR of the SADDR and SADEN registers with zeros defined as don't-care bits, e.g.:

SADDR0101 0110b
SADEN1111 1100b

SADDR OR SADEN1111 111Xb

The use of don't-care bits provides flexibility in defining the broadcast address, however in most applications, a broadcast address is FFh. The following is an example of using broadcast addresses:

Slave A:SADDR1111 0001b
SADEN1111 1010b

Given1111 1X11b,

Slave B:SADDR1111 0011b
SADEN1111 1001b

Given1111 1X11B,

Slave C:SADDR=1111 0010b
SADEN1111 1101b

Given1111 1111b

For slaves A and B, bit 2 is a don't care bit; for slave C, bit 2 is set. To communicate with all of the slaves, the master must send an address FFh. To communicate with slaves A and B, but not slave C, the master can send and address FBh.

## Registers

**Table 25.** SCON Register

SCON (S:98h)
Serial Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **FE/SM0** | **SM1** | **SM2** | **REN** | **TB8** | **RB8** | **TI** | **RI** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | FE | **Framing Error bit (SMOD0=1**)<br>Clear to reset the error state, not cleared by a valid stop bit.<br>Set by hardware when an invalid stop bit is detected. |
| | SM0 | **Serial port Mode bit 0 (SMOD0=0)**<br>Refer to SM1 for serial port mode selection. |
| 6 | SM1 | **Serial port Mode bit 1**<br>SM0 SM1 Mode Baud Rate<br>0 0 Shift Register $F_{XTAL}$/12 (or $F_{XTAL}$ /6 in mode X2)<br>0 1 8-bit UART Variable<br>1 0 9-bit UART $F_{XTAL}$/64 or $F_{XTAL}$/32<br>1 1 9-bit UART Variable |
| 5 | SM2 | **Serial port Mode 2 bit/Multiprocessor Communication Enable bit**<br>Clear to disable multiprocessor communication feature.<br>Set to enable multiprocessor communication feature in mode 2 and 3. |
| 4 | REN | **Reception Enable bit**<br>Clear to disable serial reception.<br>Set to enable serial reception. |
| 3 | TB8 | **Transmitter Bit 8/Ninth bit to transmit in modes 2 and 3**<br>Clear to transmit a logic 0 in the 9th bit.<br>Set to transmit a logic 1 in the 9th bit. |
| 2 | RB8 | **Receiver Bit 8/Ninth bit received in modes 2 and 3**<br>Cleared by hardware if 9th bit received is a logic 0.<br>Set by hardware if 9th bit received is a logic 1. |
| 1 | TI | **Transmit Interrupt flag**<br>Clear to acknowledge interrupt.<br>Set by hardware at the end of the 8th bit time in mode 0 or at the beginning of the stop bit in the other modes. |
| 0 | RI | **Receive Interrupt flag**<br>Clear to acknowledge interrupt.<br>Set by hardware at the end of the 8th bit time in mode 0, see Figure 33. and Figure 34. in the other modes. |

Reset Value = 0000 0000b
Bit addressable

**Table 26.** SADEN Register

SADEN (S:B9h)
Slave Address Mask Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | | **Mask Data for Slave Individual Address** |

Reset Value = 0000 0000b
Not bit addressable

**Table 27.** SADDR Register

SADDR (S:A9h)
Slave Address Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | | **Slave Individual Address** |

Reset Value = 0000 0000b
Not bit addressable

**Table 28.** SBUF Register

SBUF (S:99h)
Serial Data Buffer

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | | **Data sent/received by Serial I/O Port** |

Reset Value = 0000 0000b
Not bit addressable

**Table 29.** PCON Register

PCON (S:87h)
Power Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SMOD1 | SMOD0 | – | POF | GF1 | GF0 | PD | IDL |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | SMOD1 | **Serial port Mode bit 1**<br>Set to select double baud rate in mode 1, 2 or 3. |
| 6 | SMOD0 | **Serial port Mode bit 0**<br>Clear to select SM0 bit in SCON register.<br>Set to select FE bit in SCON register. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | POF | **Power-Off Flag**<br>Clear to recognize next reset type.<br>Set by hardware when VCC rises from 0 to its nominal voltage. Can also be set by software. |
| 3 | GF1 | **General-purpose Flag**<br>Cleared by user for general-purpose usage.<br>Set by user for general-purpose usage. |
| 2 | GF0 | **General-purpose Flag**<br>Cleared by user for general-purpose usage.<br>Set by user for general-purpose usage. |
| 1 | PD | **Power-Down mode bit**<br>Cleared by hardware when reset occurs.<br>Set to enter power-down mode. |
| 0 | IDL | **Idle mode bit**<br>Clear by hardware when interrupt or reset occurs.<br>Set to enter idle mode. |

Reset Value = 00X1 0000b
Not bit addressable

## Timers/Counters

The AT89C51CC03 implements two general-purpose, 16-bit Timers/Counters. Such are identified as Timer 0 and Timer 1, and can be independently configured to operate in a variety of modes as a Timer or an event Counter. When operating as a Timer, the Timer/Counter runs for a programmed length of time, then issues an interrupt request. When operating as a Counter, the Timer/Counter counts negative transitions on an external pin. After a preset number of counts, the Counter issues an interrupt request. The various operating modes of each Timer/Counter are described in the following sections.

## Timer/Counter Operations

A basic operation is Timer registers THx and TLx (x = 0, 1) connected in cascade to form a 16-bit Timer. Setting the run control bit (TRx) in TCON register (see Figure 30) turns the Timer on by allowing the selected input to increment TLx. When TLx overflows it increments THx; when THx overflows it sets the Timer overflow flag (TFx) in TCON register. Setting the TRx does not clear the THx and TLx Timer registers. Timer registers can be accessed to obtain the current count or to enter preset values. They can be read at any time but TRx bit must be cleared to preset their values, otherwise the behavior of the Timer/Counter is unpredictable.

The C/Tx# control bit selects Timer operation or Counter operation by selecting the divided-down peripheral clock or external pin Tx as the source for the counted signal. TRx bit must be cleared when changing the mode of operation, otherwise the behavior of the Timer/Counter is unpredictable.

For Timer operation (C/Tx# = 0), the Timer register counts the divided-down peripheral clock. The Timer register is incremented once every peripheral cycle (6 peripheral clock periods). The Timer clock rate is $F_{PER}/6$, i.e. $F_{OSC}/12$ in standard mode or $F_{OSC}/6$ in X2 mode.

For Counter operation (C/Tx# = 1), the Timer register counts the negative transitions on the Tx external input pin. The external input is sampled every peripheral cycles. When the sample is high in one cycle and low in the next one, the Counter is incremented. Since it takes 2 cycles (12 peripheral clock periods) to recognize a negative transition, the maximum count rate is $F_{PER}/12$, i.e. $F_{OSC}/24$ in standard mode or $F_{OSC}/12$ in X2 mode. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full peripheral cycle.

## Timer 0

Timer 0 functions as either a Timer or event Counter in four modes of operation. Figure 35 to Figure 38 show the logical configuration of each mode.

Timer 0 is controlled by the four lower bits of TMOD register (see Figure 31) and bits 0, 1, 4 and 5 of TCON register (see Figure 30). TMOD register selects the method of Timer gating (GATE0), Timer or Counter operation (T/C0#) and mode of operation (M10 and M00). TCON register provides Timer 0 control functions: overflow flag (TF0), run control bit (TR0), interrupt flag (IE0) and interrupt type control bit (IT0).

For normal Timer operation (GATE0 = 0), setting TR0 allows TL0 to be incremented by the selected input. Setting GATE0 and TR0 allows external pin INT0# to control Timer operation.

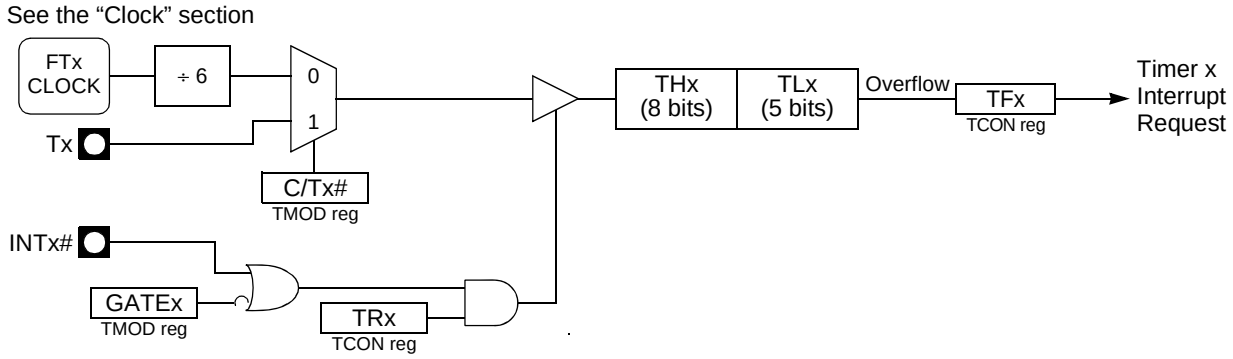
Timer 0 overflow (count rolls over from all 1s to all 0s) sets TF0 flag generating an interrupt request.

It is important to stop Timer/Counter before changing mode.

**Mode 0 (13-bit Timer)**

Mode 0 configures Timer 0 as an 13-bit Timer which is set up as an 8-bit Timer (TH0 register) with a modulo 32 prescaler implemented with the lower five bits of TL0 register (see Figure 35). The upper three bits of TL0 register are indeterminate and should be ignored. Prescaler overflow increments TH0 register.

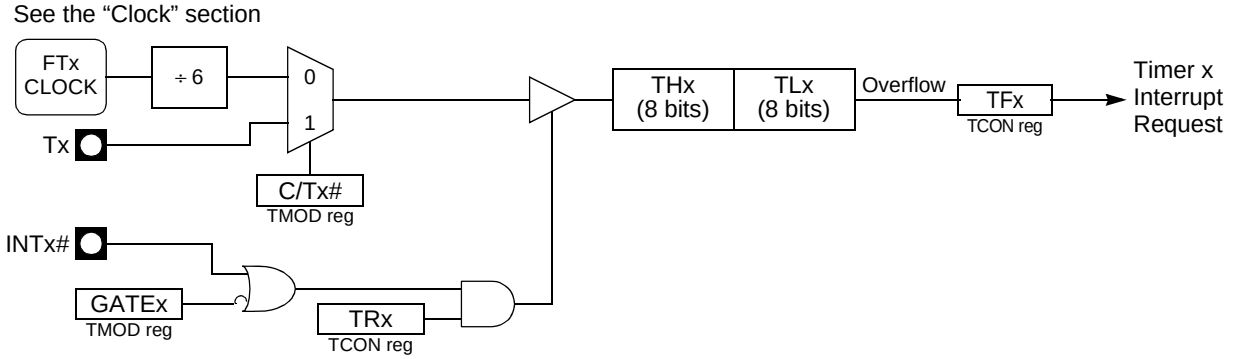
**Figure 35.** Timer/Counter x (x = 0 or 1) in Mode 0




**Mode 1 (16-bit Timer)**

Mode 1 configures Timer 0 as a 16-bit Timer with TH0 and TL0 registers connected in cascade (see Figure 36). The selected input increments TL0 register.

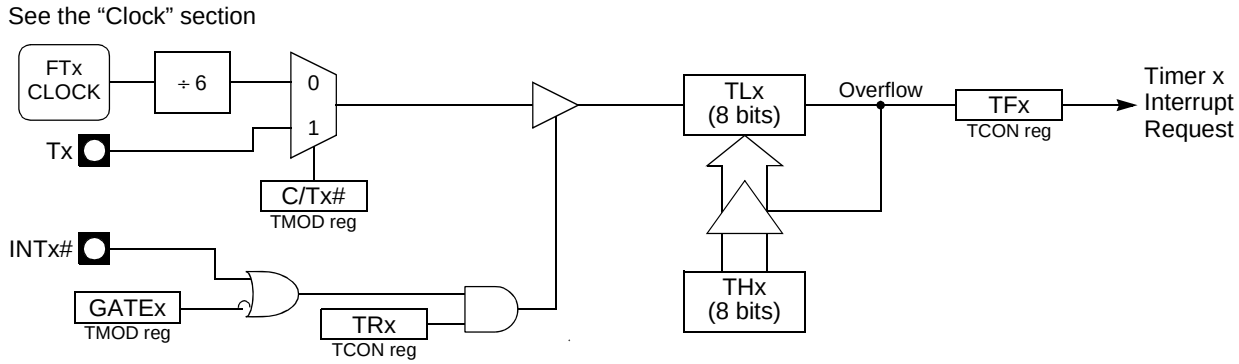
**Figure 36.** Timer/Counter x (x = 0 or 1) in Mode 1



**Mode 2 (8-bit Timer with Auto-Reload)**

Mode 2 configures Timer 0 as an 8-bit Timer (TL0 register) that automatically reloads from TH0 register (see Figure 37). TL0 overflow sets TF0 flag in TCON register and reloads TL0 with the contents of TH0, which is preset by software. When the interrupt request is serviced, hardware clears TF0. The reload leaves TH0 unchanged. The next reload value may be changed at any time by writing it to TH0 register.

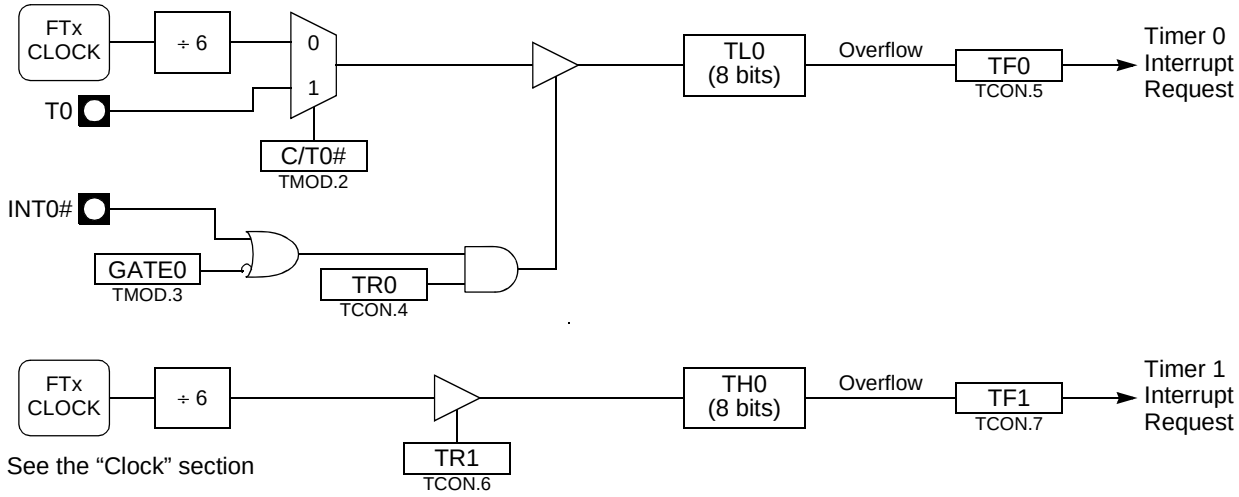
**Figure 37.** Timer/Counter x (x = 0 or 1) in Mode 2




**Mode 3 (Two 8-bit Timers)**

Mode 3 configures Timer 0 such that registers TL0 and TH0 operate as separate 8-bit Timers (see Figure 38). This mode is provided for applications requiring an additional 8-bit Timer or Counter. TL0 uses the Timer 0 control bits C/T0# and GATE0 in TMOD register, and TR0 and TF0 in TCON register in the normal manner. TH0 is locked into a Timer function (counting $F_{PER}$ /6) and takes over use of the Timer 1 interrupt (TF1) and run control (TR1) bits. Thus, operation of Timer 1 is restricted when Timer 0 is in mode 3.

**Figure 38.** Timer/Counter 0 in Mode 3: Two 8-bit Counters



## Timer 1

Timer 1 is identical to Timer 0 excepted for Mode 3 which is a hold-count mode. The following comments help to understand the differences:

- Timer 1 functions as either a Timer or event Counter in three modes of operation. Figure 35 to Figure 37 show the logical configuration for modes 0, 1, and 2. Timer 1's mode 3 is a hold-count mode.
- Timer 1 is controlled by the four high-order bits of TMOD register (see Figure 31) and bits 2, 3, 6 and 7 of TCON register (see Figure 30). TMOD register selects the method of Timer gating (GATE1), Timer or Counter operation (C/T1#) and mode of operation (M11 and M01). TCON register provides Timer 1 control functions: overflow flag (TF1), run control bit (TR1), interrupt flag (IE1) and interrupt type control bit (IT1).
- Timer 1 can serve as the Baud Rate Generator for the Serial Port. Mode 2 is best suited for this purpose.
- For normal Timer operation (GATE1 = 0), setting TR1 allows TL1 to be incremented by the selected input. Setting GATE1 and TR1 allows external pin INT1# to control Timer operation.
- Timer 1 overflow (count rolls over from all 1s to all 0s) sets the TF1 flag generating an interrupt request.
- When Timer 0 is in mode 3, it uses Timer 1's overflow flag (TF1) and run control bit (TR1). For this situation, use Timer 1 only for applications that do not require an interrupt (such as a Baud Rate Generator for the Serial Port) and switch Timer 1 in and out of mode 3 to turn it off and on.
- It is important to stop Timer/Counter before changing mode.

**Mode 0 (13-bit Timer)**

Mode 0 configures Timer 1 as a 13-bit Timer, which is set up as an 8-bit Timer (TH1 register) with a modulo-32 prescaler implemented with the lower 5 bits of the TL1 register (see Figure 35). The upper 3 bits of TL1 register are ignored. Prescaler overflow increments TH1 register.

**Mode 1 (16-bit Timer)**

Mode 1 configures Timer 1 as a 16-bit Timer with TH1 and TL1 registers connected in cascade (see Figure 36). The selected input increments TL1 register.

**Mode 2 (8-bit Timer with Auto-Reload)**

Mode 2 configures Timer 1 as an 8-bit Timer (TL1 register) with automatic reload from TH1 register on overflow (see Figure 37). TL1 overflow sets TF1 flag in TCON register and reloads TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.

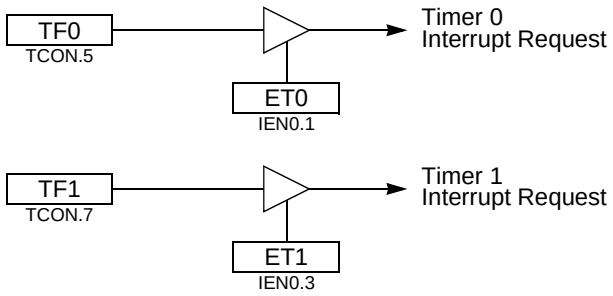
**Mode 3 (Halt)**

Placing Timer 1 in mode 3 causes it to halt and hold its count. This can be used to halt Timer 1 when TR1 run control bit is not available i.e. when Timer 0 is in mode 3.

## Interrupt

Each Timer handles one interrupt source that is the timer overflow flag TF0 or TF1. This flag is set every time an overflow occurs. Flags are cleared when vectoring to the Timer interrupt routine. Interrupts are enabled by setting ETx bit in IEN0 register. This assumes interrupts are globally enabled by setting EA bit in IEN0 register.

**Figure 39.** Timer Interrupt System




## Registers

**Table 30.** TCON Register
TCON (S:88h)
Timer/Counter Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **TF1** | **TR1** | **TF0** | **TR0** | **IE1** | **IT1** | **IE0** | **IT0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | TF1 | **Timer 1 Overflow Flag**<br>Cleared by hardware when processor vectors to interrupt routine.<br>Set by hardware on Timer/Counter overflow, when Timer 1 register overflows. |
| 6 | TR1 | **Timer 1 Run Control Bit**<br>Clear to turn off Timer/Counter 1.<br>Set to turn on Timer/Counter 1. |
| 5 | TF0 | **Timer 0 Overflow Flag**<br>Cleared by hardware when processor vectors to interrupt routine.<br>Set by hardware on Timer/Counter overflow, when Timer 0 register overflows. |
| 4 | TR0 | **Timer 0 Run Control Bit**<br>Clear to turn off Timer/Counter 0.<br>Set to turn on Timer/Counter 0. |
| 3 | IE1 | **Interrupt 1 Edge Flag**<br>Cleared by hardware when interrupt is processed if edge-triggered (see IT1).<br>Set by hardware when external interrupt is detected on INT1# pin. |
| 2 | IT1 | **Interrupt 1 Type Control Bit**<br>Clear to select low level active (level triggered) for external interrupt 1 (INT1#).<br>Set to select falling edge active (edge triggered) for external interrupt 1. |
| 1 | IE0 | **Interrupt 0 Edge Flag**<br>Cleared by hardware when interrupt is processed if edge-triggered (see IT0).<br>Set by hardware when external interrupt is detected on INT0# pin. |
| 0 | IT0 | **Interrupt 0 Type Control Bit**<br>Clear to select low level active (level triggered) for external interrupt 0 (INT0#).<br>Set to select falling edge active (edge triggered) for external interrupt 0. |

Reset Value = 0000 0000b

**Table 31.** TMOD Register

TMOD (S:89h)
Timer/Counter Mode Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **GATE1** | **C/T1#** | **M11** | **M01** | **GATE0** | **C/T0#** | **M10** | **M00** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | GATE1 | **Timer 1 Gating Control Bit**<br>Clear to enable Timer 1 whenever TR1 bit is set.<br>Set to enable Timer 1 only while INT1# pin is high and TR1 bit is set. |
| 6 | C/T1# | **Timer 1 Counter/Timer Select Bit**<br>Clear for Timer operation: Timer 1 counts the divided-down system clock.<br>Set for Counter operation: Timer 1 counts negative transitions on external pin T1. |
| 5 | M11 | **Timer 1 Mode Select Bits**<br>M11 M01 Operating mode<br>0 0 Mode 0: 8-bit Timer/Counter (TH1) with 5-bit prescaler (TL1).<br>0 1 Mode 1: 16-bit Timer/Counter.<br>1 0 Mode 2: 8-bit auto-reload Timer/Counter (TL1) (1)<br>1 1 Mode 3: Timer 1 halted. Retains count |
| 4 | M01 | |
| 3 | GATE0 | **Timer 0 Gating Control Bit**<br>Clear to enable Timer 0 whenever TR0 bit is set.<br>Set to enable Timer/Counter 0 only while INT0# pin is high and TR0 bit is set. |
| 2 | C/T0# | **Timer 0 Counter/Timer Select Bit**<br>Clear for Timer operation: Timer 0 counts the divided-down system clock.<br>Set for Counter operation: Timer 0 counts negative transitions on external pin T0. |
| 1 | M10 | **Timer 0 Mode Select Bit**<br>M10 M00 Operating mode<br>0 0 Mode 0: 8-bit Timer/Counter (TH0) with 5-bit prescaler (TL0).<br>0 1 Mode 1: 16-bit Timer/Counter.<br>1 0 Mode 2: 8-bit auto-reload Timer/Counter (TL0) (2)<br>1 1 Mode 3: TL0 is an 8-bit Timer/Counter<br>TH0 is an 8-bit Timer using Timer 1's TR0 and TF0 bits. |
| 0 | M00 | |

1. Reloaded from TH1 at overflow.
2. Reloaded from TH0 at overflow.

Reset Value = 0000 0000b

**Table 32.** TH0 Register

TH0 (S:8Ch)
Timer 0 High Byte Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:0 | | **High Byte of Timer 0.** |

Reset Value = 0000 0000b

**Table 33.** TL0 Register

TL0 (S:8Ah)
Timer 0 Low Byte Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:0 | | **Low Byte of Timer 0.** |

Reset Value = 0000 0000b

**Table 34.** TH1 Register

TH1 (S:8Dh)
Timer 1 High Byte Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:0 | | **High Byte of Timer 1.** |

Reset Value = 0000 0000b

**Table 35.** TL1 Register

TL1 (S:8Bh)
Timer 1 Low Byte Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:0 | | **Low Byte of Timer 1.** |

Reset Value = 0000 0000b

## Timer 2

The AT89C51CC03 timer 2 is compatible with timer 2 in the 80C52.

It is a 16-bit timer/counter: the count is maintained by two eight-bit timer registers, TH2 and TL2 that are cascade- connected. It is controlled by T2CON register (See Table ) and T2MOD register (See Table 38). Timer 2 operation is similar to Timer 0 and Timer 1. C/$\overline{T2}$ selects $F_{T2\ clock}$/6 (timer operation) or external pin T2 (counter operation) as timer clock. Setting TR2 allows TL2 to be incremented by the selected input.

Timer 2 includes the following enhancements:

- Auto-reload mode (up or down counter)
- Programmable clock-output

### Auto-Reload Mode

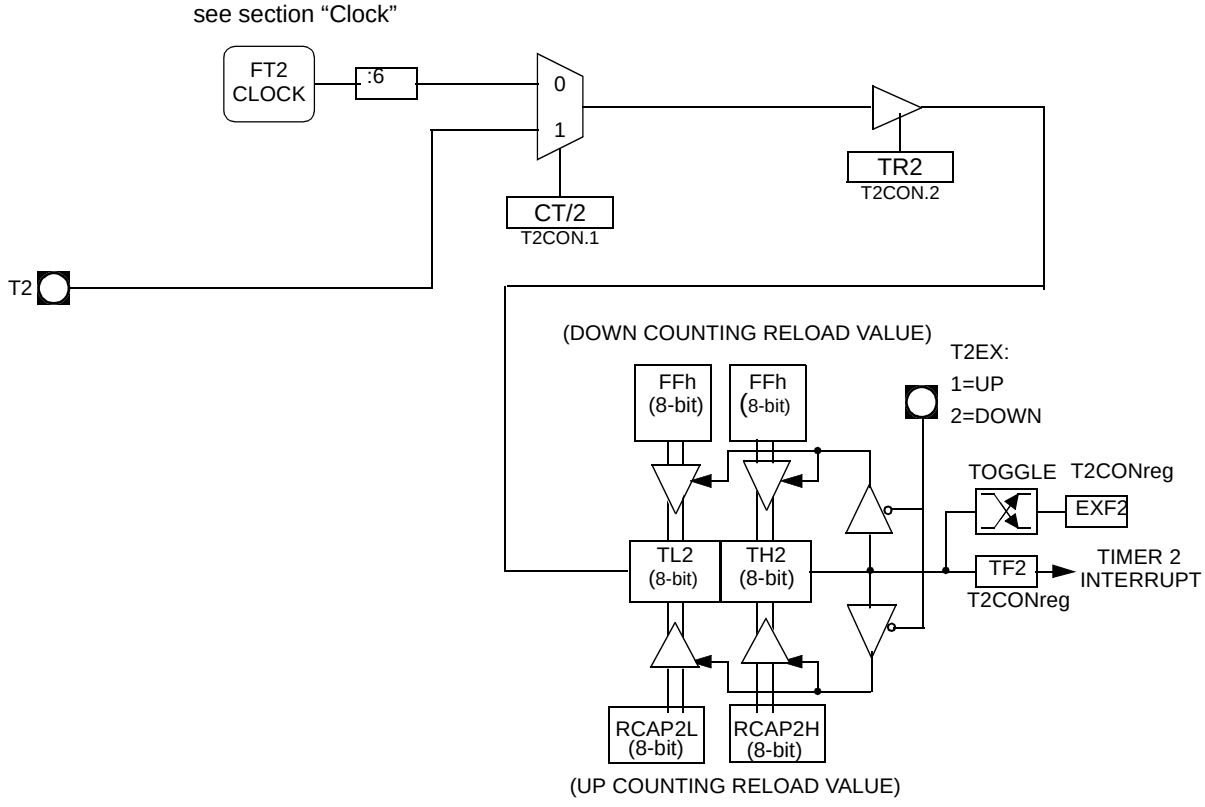
The auto-reload mode configures timer 2 as a 16-bit timer or event counter with automatic reload. This feature is controlled by the DCEN bit in T2MOD register (See Table 38). Setting the DCEN bit enables timer 2 to count up or down as shown in Figure 40. In this mode the T2EX pin controls the counting direction.

When T2EX is high, timer 2 counts up. Timer overflow occurs at FFFFh which sets the TF2 flag and generates an interrupt request. The overflow also causes the 16-bit value in RCAP2H and RCAP2L registers to be loaded into the timer registers TH2 and TL2.

When T2EX is low, timer 2 counts down. Timer underflow occurs when the count in the timer registers TH2 and TL2 equals the value stored in RCAP2H and RCAP2L registers. The underflow sets TF2 flag and reloads FFFFh into the timer registers.

The EXF2 bit toggles when timer 2 overflow or underflow, depending on the direction of the count. EXF2 does not generate an interrupt. This bit can be used to provide 17-bit resolution.

**Figure 40.** Auto-Reload Mode Up/Down Counter



## Programmable Clock-Output

In clock-out mode, timer 2 operates as a 50%-duty-cycle, programmable clock generator (See Figure 41). The input clock increments TL2 at frequency $F_{OSC}/2$. The timer repeatedly counts to overflow from a loaded value. At overflow, the contents of RCAP2H and RCAP2L registers are loaded into TH2 and TL2. In this mode, timer 2 overflows do not generate interrupts. The formula gives the clock-out frequency depending on the system oscillator frequency and the value in the RCAP2H and RCAP2L registers:

$$Clock-OutFrequency = \frac{FT2clock}{4 \times (65536 - RCAP2H/RCAP2L)}$$

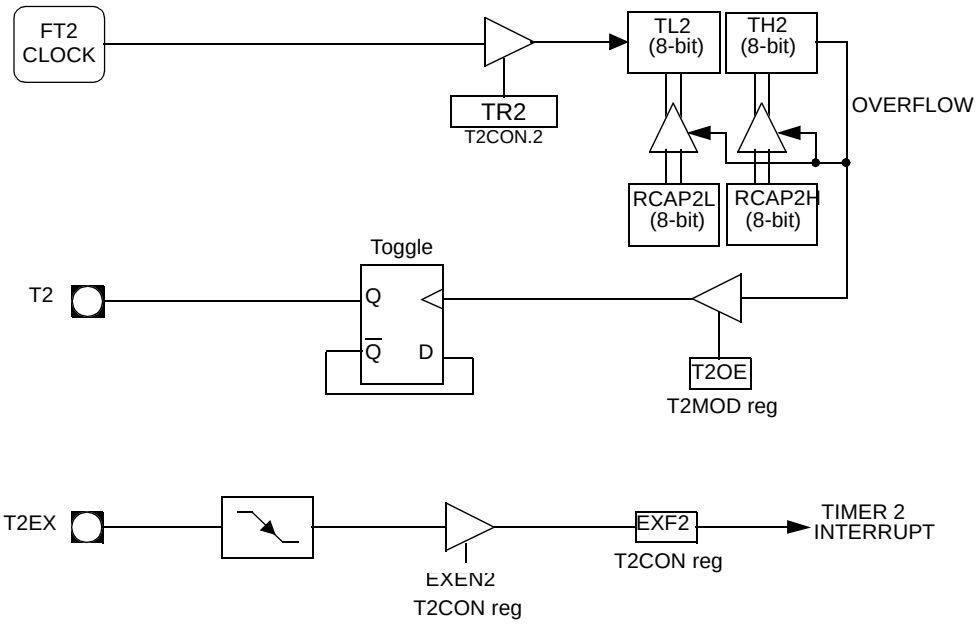
For a 16 MHz system clock in x1 mode, timer 2 has a programmable frequency range of 61 Hz ($F_{OSC}/2^{16}$) to 4 MHz ($F_{OSC}/4$). The generated clock signal is brought out to T2 pin (P1.0).

Timer 2 is programmed for the clock-out mode as follows:

- Set T2OE bit in T2MOD register.
- Clear C/$\overline{T2}$ bit in T2CON register.
- Determine the 16-bit reload value from the formula and enter it in RCAP2H/RCAP2L registers.
- Enter a 16-bit initial value in timer registers TH2/TL2. It can be the same as the reload value or different depending on the application.
- To start the timer, set TR2 run control bit in T2CON register.

It is possible to use timer 2 as a baud rate generator and a clock generator simultaneously. For this configuration, the baud rates and clock frequencies are not independent since both functions use the values in the RCAP2H and RCAP2L registers.

**Figure 41.** Clock-Out Mode



## Registers

**Table 36.** T2CON Register

T2CON (S:C8h)
Timer 2 Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/T2# | CP/RL2# |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | TF2 | **Timer 2 Overflow Flag**<br>TF2 is not set if RCLK=1 or TCLK = 1.<br>Must be cleared by software.<br>Set by hardware on timer 2 overflow. |
| 6 | EXF2 | **Timer 2 External Flag**<br>Set when a capture or a reload is caused by a negative transition on T2EX pin if EXEN2=1.<br>Set to cause the CPU to vector to timer 2 interrupt routine when timer 2 interrupt is enabled.<br>Must be cleared by software. |
| 5 | RCLK | **Receive Clock bit**<br>Clear to use timer 1 overflow as receive clock for serial port in mode 1 or 3.<br>Set to use timer 2 overflow as receive clock for serial port in mode 1 or 3. |
| 4 | TCLK | **Transmit Clock bit**<br>Clear to use timer 1 overflow as transmit clock for serial port in mode 1 or 3.<br>Set to use timer 2 overflow as transmit clock for serial port in mode 1 or 3. |
| 3 | EXEN2 | **Timer 2 External Enable bit**<br>Clear to ignore events on T2EX pin for timer 2 operation.<br>Set to cause a capture or reload when a negative transition on T2EX pin is detected, if timer 2 is not used to clock the serial port. |
| 2 | TR2 | **Timer 2 Run Control bit**<br>Clear to turn off timer 2.<br>Set to turn on timer 2. |
| 1 | C/T2# | **Timer/Counter 2 Select bit**<br>Clear for timer operation (input from internal clock system: $F_{OSC}$).<br>Set for counter operation (input from T2 input pin). |
| 0 | CP/RL2# | **Timer 2 Capture/Reload bit**<br>If RCLK=1 or TCLK=1, CP/RL2# is ignored and timer is forced to auto-reload on timer 2 overflow.<br>Clear to auto-reload on timer 2 overflows or negative transitions on T2EX pin if EXEN2=1.<br>Set to capture on negative transitions on T2EX pin if EXEN2=1. |

Reset Value = 0000 0000b
Bit addressable

**Table 37.** T2MOD Register

T2MOD (S:C9h)
Timer 2 Mode Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | T2OE | DCEN |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 2 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 1 | T2OE | **Timer 2 Output Enable bit**<br>Clear to program P1.0/T2 as clock input or I/O port.<br>Set to program P1.0/T2 as clock output. |
| 0 | DCEN | **Down Counter Enable bit**<br>Clear to disable timer 2 as up/down counter.<br>Set to enable timer 2 as up/down counter. |

Reset Value = XXXX XX00b
Not bit addressable

**Table 38.** TH2 Register

TH2 (S:CDh)
Timer 2 High Byte Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | | High Byte of Timer 2. |

Reset Value = 0000 0000b
Not bit addressable

**Table 39.** TL2 Register

TL2 (S:CCh)
Timer 2 Low Byte Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | | Low Byte of Timer 2. |

Reset Value = 0000 0000b
Not bit addressable

**Table 40.** RCAP2H Register

RCAP2H (S:CBh)
Timer 2 Reload/Capture High Byte Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | | High Byte of Timer 2 Reload/Capture. |

Reset Value = 0000 0000b
Not bit addressable

**Table 41.** RCAP2L Register

RCAP2L (S:CA$_H$)
TIMER 2 RELOAD/CAPTURE Low Byte Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | | Low Byte of Timer 2 Reload/Capture. |

Reset Value = 0000 0000b
Not bit addressable

## Watchdog Timer

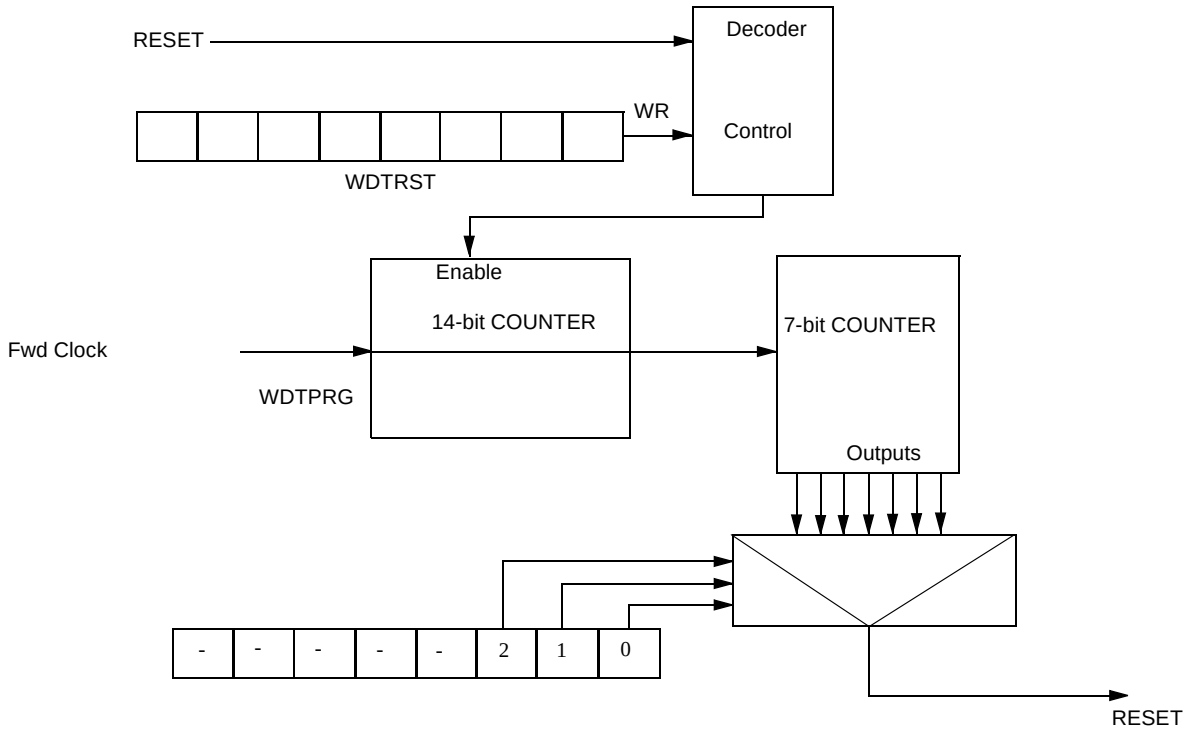
AT89C51CC03 contains a powerful programmable hardware Watchdog Timer (WDT) that automatically resets the chip if it software fails to reset the WDT before the selected time interval has elapsed. It permits large Time-Out ranking from 16ms to 2s @Fosc = 12MHz in X1 mode.

This WDT consists of a 14-bit counter plus a 7-bit programmable counter, a Watchdog Timer reset register (WDTRST) and a Watchdog Timer programming (WDTPRG) register. When exiting reset, the WDT is -by default- disable.

To enable the WDT, the user has to write the sequence 1EH and E1H into WDTRST register no instruction in between. When the Watchdog Timer is enabled, it will increment every machine cycle while the oscillator is running and there is no way to disable the WDT except through reset (either hardware reset or WDT overflow reset). When WDT overflows, it will generate an output RESET pulse at the RST pin. The RESET pulse duration is $96xT_{OSC}$, where $T_{OSC}=1/F_{OSC}$. To make the best use of the WDT, it should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset

Note: When the Watchdog is enable it is impossible to change its period.

**Figure 42.** Watchdog Timer



## Watchdog Programming

The three lower bits (S0, S1, S2) located into WDTPRG register permit to program the WDT duration.

**Table 42.** Machine Cycle Count

| S2 | S1 | S0 | Machine Cycle Count |
|---|---|---|---|
| 0 | 0 | 0 | $2^{14}$ |
| 0 | 0 | 1 | $2^{15}$ |
| 0 | 1 | 0 | $2^{16}$ |
| 0 | 1 | 1 | $2^{17}$ |
| 1 | 0 | 0 | $2^{18}$ |
| 1 | 0 | 1 | $2^{19}$ |
| 1 | 1 | 0 | $2^{20}$ |
| 1 | 1 | 1 | $2^{21}$ |

To compute WD Time-Out, the following formula is applied:

$$FTime-Out = \frac{F_{osc}}{6 \times 2^{WDX2 \wedge X2}(2^{14} \times 2^{Svalue})}$$

Note: Svalue represents the decimal value of (S2 S1 S0)

The following table outlines the time-out value for $Fosc_{XTAL}$ = 12 MHz in X1 mode

**Table 43.** Time-Out Computation

| S2 | S1 | S0 | Fosc = 12 MHz | Fosc = 16 MHz | Fosc = 20 MHz |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 16.38 ms | 12.28 ms | 9.82 ms |
| 0 | 0 | 1 | 32.77 ms | 24.57 ms | 19.66 ms |
| 0 | 1 | 0 | 65.54 ms | 49.14 ms | 39.32 ms |
| 0 | 1 | 1 | 131.07 ms | 98.28 ms | 78.64 ms |
| 1 | 0 | 0 | 262.14 ms | 196.56 ms | 157.28 ms |
| 1 | 0 | 1 | 524.29 ms | 393.12 ms | 314.56 ms |
| 1 | 1 | 0 | 1.05 s | 786.24 ms | 629.12 ms |
| 1 | 1 | 1 | 2.10 s | 1.57 s | 1.25 s |

## Watchdog Timer During Power-down Mode and Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode, the user does not need to service the WDT. There are 2 methods of exiting Power-down mode: by a hardware reset or via a level activated external interrupt which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, the Watchdog is disabled. Exiting Power-down with an interrupt is significantly different. The interrupt shall be held low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service for the interrupt used to exit Power-down.

To ensure that the WDT does not overflow within a few states of exiting powerdown, it is best to reset the WDT just before entering powerdown.

In the Idle mode, the oscillator continues to run. To prevent the WDT from resetting AT89C51CC03 while in Idle mode, the user should always set up a timer that will periodically exit Idle, service the WDT, and re-enter Idle mode.

### Register

**Table 44.** WDTPRG Register

WDTPRG (S:A7h)
Watchdog Timer Duration Programming Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | S2 | S1 | S0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 2 | S2 | **Watchdog Timer Duration selection bit 2**<br>Work in conjunction with bit 1 and bit 0. |
| 1 | S1 | **Watchdog Timer Duration selection bit 1**<br>Work in conjunction with bit 2 and bit 0. |
| 0 | S0 | **Watchdog Timer Duration selection bit 0**<br>Work in conjunction with bit 1 and bit 2. |

Reset Value = XXXX X000b

**Table 45.** WDTRST Register

WDTRST (S:A6h Write only)
Watchdog Timer Enable Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | Watchdog Control Value |

Reset Value = 1111 1111b

Note: The WDRST register is used to reset/enable the WDT by writing 1EH then E1H in sequence without instruction between these two sequences.

## CAN Controller

The CAN Controller provides all the features required to implement the serial communication protocol CAN as defined by BOSCH GmbH. The CAN specification as referred to by ISO/11898 (2.0A and 2.0B) for high speed and ISO/11519-2 for low speed. The CAN Controller is able to handle all types of frames (Data, Remote, Error and Overload) and achieves a bitrate of 1-Mbit/sec at 8 MHz[1] Crystal frequency in X2 mode.

Note: 1. At BRP = 1 sampling point will be fixed.

## CAN Protocol

The CAN protocol is an international standard defined in the ISO 11898 for high speed and ISO 11519-2 for low speed.

### Principles

CAN is based on a broadcast communication mechanism. This broadcast communication is achieved by using a message oriented transmission protocol. These messages are identified by using a message identifier. Such a message identifier has to be unique within the whole network and it defines not only the content but also the priority of the message.

The priority at which a message is transmitted compared to another less urgent message is specified by the identifier of each message. The priorities are laid down during system design in the form of corresponding binary values and cannot be changed dynamically. The identifier with the lowest binary number has the highest priority.

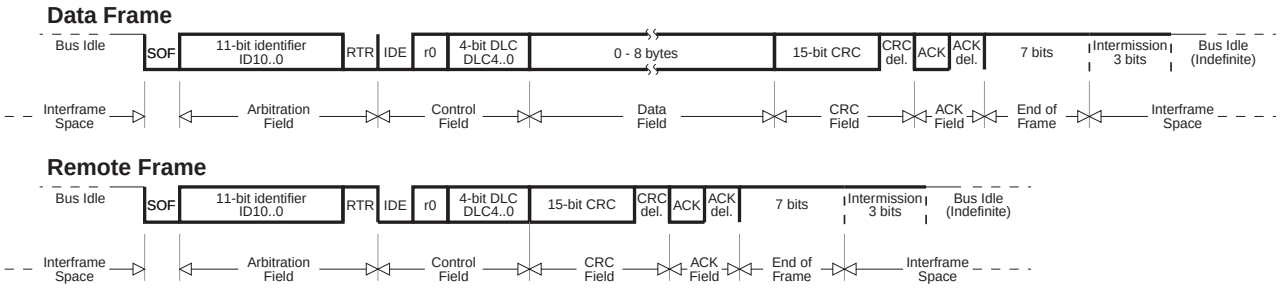
Bus access conflicts are resolved by bit-wise arbitration on the identifiers involved by each node observing the bus level bit for bit. This happens in accordance with the "wired and" mechanism, by which the dominant state overwrites the recessive state. The competition for bus allocation is lost by all nodes with recessive transmission and dominant observation. All the "losers" automatically become receivers of the message with the highest priority and do not re-attempt transmission until the bus is available again.

### Message Formats

The CAN protocol supports two message frame formats, the only essential difference being in the length of the identifier. The CAN standard frame, also known as CAN 2.0 A, supports a length of 11 bits for the identifier, and the CAN extended frame, also known as CAN 2.0 B, supports a length of 29 bits for the identifier.

*Can Standard Frame*

**Figure 43.** CAN Standard Frames

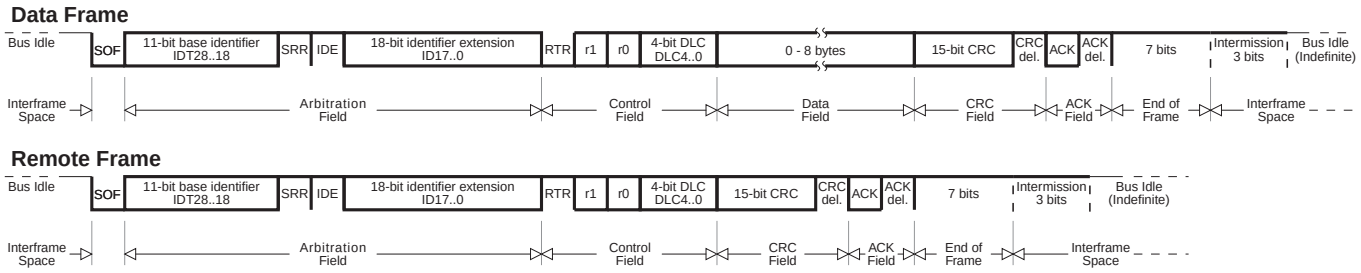



A message in the CAN standard frame format begins with the "Start Of Frame (SOF)", this is followed by the "Arbitration field" which consist of the identifier and the "Remote Transmission Request (RTR)" bit used to distinguish between the data frame and the data request frame called remote frame. The following "Control field" contains the "IDentifier Extension (IDE)" bit and the "Data Length Code (DLC)" used to indicate the

number of following data bytes in the "Data field". In a remote frame, the DLC contains the number of requested data bytes. The "Data field" that follows can hold up to 8 data bytes. The frame integrity is guaranteed by the following "Cyclic Redundant Check (CRC)" sum. The "ACKnowledge (ACK) field" compromises the ACK slot and the ACK delimiter. The bit in the ACK slot is sent as a recessive bit and is overwritten as a dominant bit by the receivers which have at this time received the data correctly. Correct messages are acknowledged by the receivers regardless of the result of the acceptance test. The end of the message is indicated by "End Of Frame (EOF)". The "Intermission Frame Space (IFS)" is the minimum number of bits separating consecutive messages. If there is no following bus access by any node, the bus remains idle.

*CAN Extended Frame*

**Figure 44.** CAN Extended Frames




A message in the CAN extended frame format is likely the same as a message in CAN standard frame format. The difference is the length of the identifier used. The identifier is made up of the existing 11-bit identifier (base identifier) and an 18-bit extension (identifier extension). The distinction between CAN standard frame format and CAN extended frame format is made by using the IDE bit which is transmitted as dominant in case of a frame in CAN standard frame format, and transmitted as recessive in the other case.

Format Co-existence

As the two formats have to co-exist on one bus, it is laid down which message has higher priority on the bus in the case of bus access collision with different formats and the same identifier / base identifier: The message in CAN standard frame format always has priority over the message in extended format.

There are three different types of CAN modules available:

- 2.0A - Considers 29 bit ID as an error
- 2.0B Passive - Ignores 29 bit ID messages
- 2.0B Active - Handles both 11 and 29 bit ID Messages

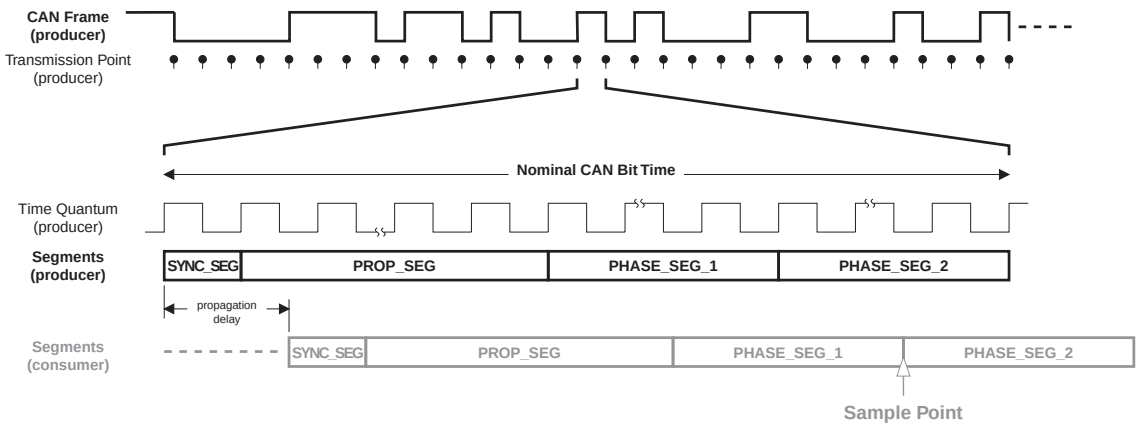
**Bit Timing**

To ensure correct sampling up to the last bit, a CAN node needs to re-synchronize throughout the entire frame. This is done at the beginning of each message with the falling edge SOF and on each recessive to dominant edge.

Bit Construction

One CAN bit time is specified as four non-overlapping time segments. Each segment is constructed from an integer multiple of the Time Quantum. The Time Quantum or TQ is the smallest discrete timing resolution used by a CAN node.

**Figure 45.** CAN Bit Construction




Synchronization Segment

The first segment is used to synchronize the various bus nodes.

On transmission, at the start of this segment, the current bit level is output. If there is a bit state change between the previous bit and the current bit, then the bus state change is expected to occur within this segment by the receiving nodes.

Propagation Time Segment

This segment is used to compensate for signal delays across the network.

This is necessary to compensate for signal propagation delays on the bus line and through the transceivers of the bus nodes.

Phase Segment 1

Phase Segment 1 is used to compensate for edge phase errors.

This segment may be lengthened during resynchronization.

Sample Point

The sample point is the point of time at which the bus level is read and interpreted as the value of the respective bit. Its location is at the end of Phase Segment 1 (between the two Phase Segments).

Phase Segment 2

This segment is also used to compensate for edge phase errors.

This segment may be shortened during resynchronization, but the length has to be at least as long as the information processing time and may not be more than the length of Phase Segment 1.

Information Processing Time

It is the time required for the logic to determine the bit level of a sampled bit.

The Information processing Time begins at the sample point, is measured in TQ and is fixed at 2 TQ for the Atmel CAN. Since Phase Segment 2 also begins at the sample point and is the last segment in the bit time, Phase Segment 2 minimum shall not be less than the Information processing Time.

Bit Lengthening

As a result of resynchronization, Phase Segment 1 may be lengthened or Phase Segment 2 may be shortened to compensate for oscillator tolerances. If, for example, the transmitter oscillator is slower than the receiver oscillator, the next falling edge used for resynchronization may be delayed. So Phase Segment 1 is lengthened in order to adjust the sample point and the end of the bit time.

Bit Shortening

If, on the other hand, the transmitter oscillator is faster than the receiver one, the next falling edge used for resynchronization may be too early. So Phase Segment 2 in bit N is shortened in order to adjust the sample point for bit N+1 and the end of the bit time

Synchronization Jump Width

The limit to the amount of lengthening or shortening of the Phase Segments is set by the Resynchronization Jump Width.

This segment may not be longer than Phase Segment 2.

Programming the Sample Point

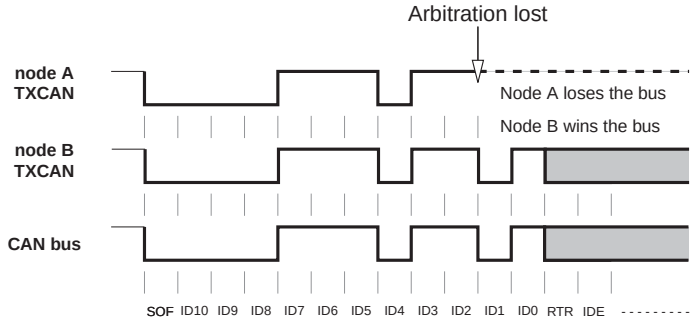
Programming of the sample point allows "tuning" of the characteristics to suit the bus.

Early sampling allows more Time Quanta in the Phase Segment 2 so the Synchronization Jump Width can be programmed to its maximum. This maximum capacity to shorten or lengthen the bit time decreases the sensitivity to node oscillator tolerances, so that lower cost oscillators such as ceramic resonators may be used.

Late sampling allows more Time Quanta in the Propagation Time Segment which allows a poorer bus topology and maximum bus length.

**Arbitration**

**Figure 46.** Bus Arbitration




The CAN protocol handles bus accesses according to the concept called “Carrier Sense Multiple Access with Arbitration on Message Priority”.

During transmission, arbitration on the CAN bus can be lost to a competing device with a higher priority CAN Identifier. This arbitration concept avoids collisions of messages whose transmission was started by more than one node simultaneously and makes sure the most important message is sent first without time loss.

The bus access conflict is resolved during the arbitration field mostly over the identifier value. If a data frame and a remote frame with the same identifier are initiated at the same time, the data frame prevails over the remote frame (c.f. RTR bit).

**Errors**

The CAN protocol signals any errors immediately as they occur. Three error detection mechanisms are implemented at the message level and two at the bit level:

Error at Message Level

- Cyclic Redundancy Check (CRC)
  The CRC safeguards the information in the frame by adding redundant check bits at the transmission end. At the receiver these bits are re-computed and tested against the received bits. If they do not agree there has been a CRC error.
- Frame Check
  This mechanism verifies the structure of the transmitted frame by checking the bit

fields against the fixed format and the frame size. Errors detected by frame checks are designated "format errors".

- ACK Errors
  As already mentioned frames received are acknowledged by all receivers through positive acknowledgement. If no acknowledgement is received by the transmitter of the message an ACK error is indicated.

Error at Bit Level

- Monitoring
  The ability of the transmitter to detect errors is based on the monitoring of bus signals. Each node which transmits also observes the bus level and thus detects differences between the bit sent and the bit received. This permits reliable detection of global errors and errors local to the transmitter.
- Bit Stuffing
  The coding of the individual bits is tested at bit level. The bit representation used by CAN is "Non Return to Zero (NRZ)" coding, which guarantees maximum efficiency in bit coding. The synchronization edges are generated by means of bit stuffing.

Error Signalling

If one or more errors are discovered by at least one node using the above mechanisms, the current transmission is aborted by sending an "error flag". This prevents other nodes accepting the message and thus ensures the consistency of data throughout the network. After transmission of an erroneous message that has been aborted, the sender automatically re-attempts transmission.

## CAN Controller Description

The CAN Controller accesses are made through SFR.
Several operations are possible by SFR:

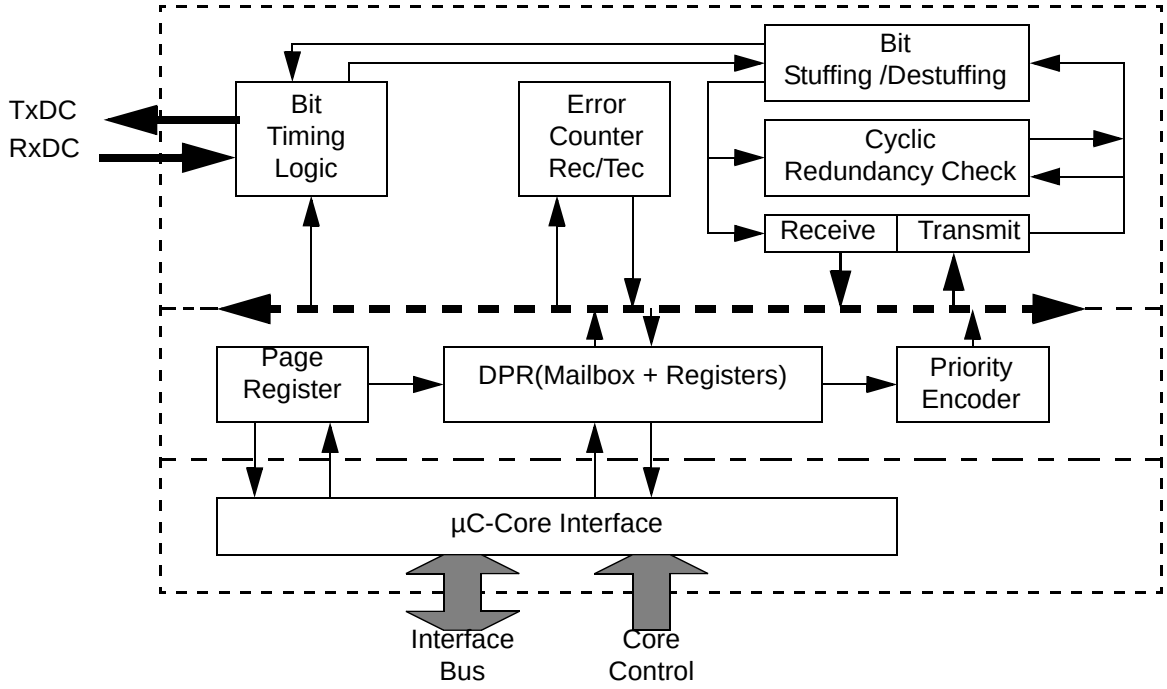
- arithmetic and logic operations, transfers and program control (SFR is accessible by direct addressing).
- 15 independent message objects are implemented, a pagination system manages their accesses.

Any message object can be programmed in a reception buffer block (even non-consecutive buffers). For the reception of defined messages one or several receiver message objects can be masked without participating in the buffer feature. An IT is generated when the buffer is full. The frames following the buffer-full interrupt will not be taken into account until at least one of the buffer message objects is re-enabled in reception. Higher priority of a message object for reception or transmission is given to the lower message object number.

The programmable 16-bit Timer (CANTIMER) is used to stamp each received and sent message in the CANSTMP register. This timer starts counting as soon as the CAN controller is enabled by the ENA bit in the CANGCON register.

The Time Trigger Communication (TTC) protocol is supported by the AT89C51CC03.

**Figure 47.** CAN Controller Block Diagram

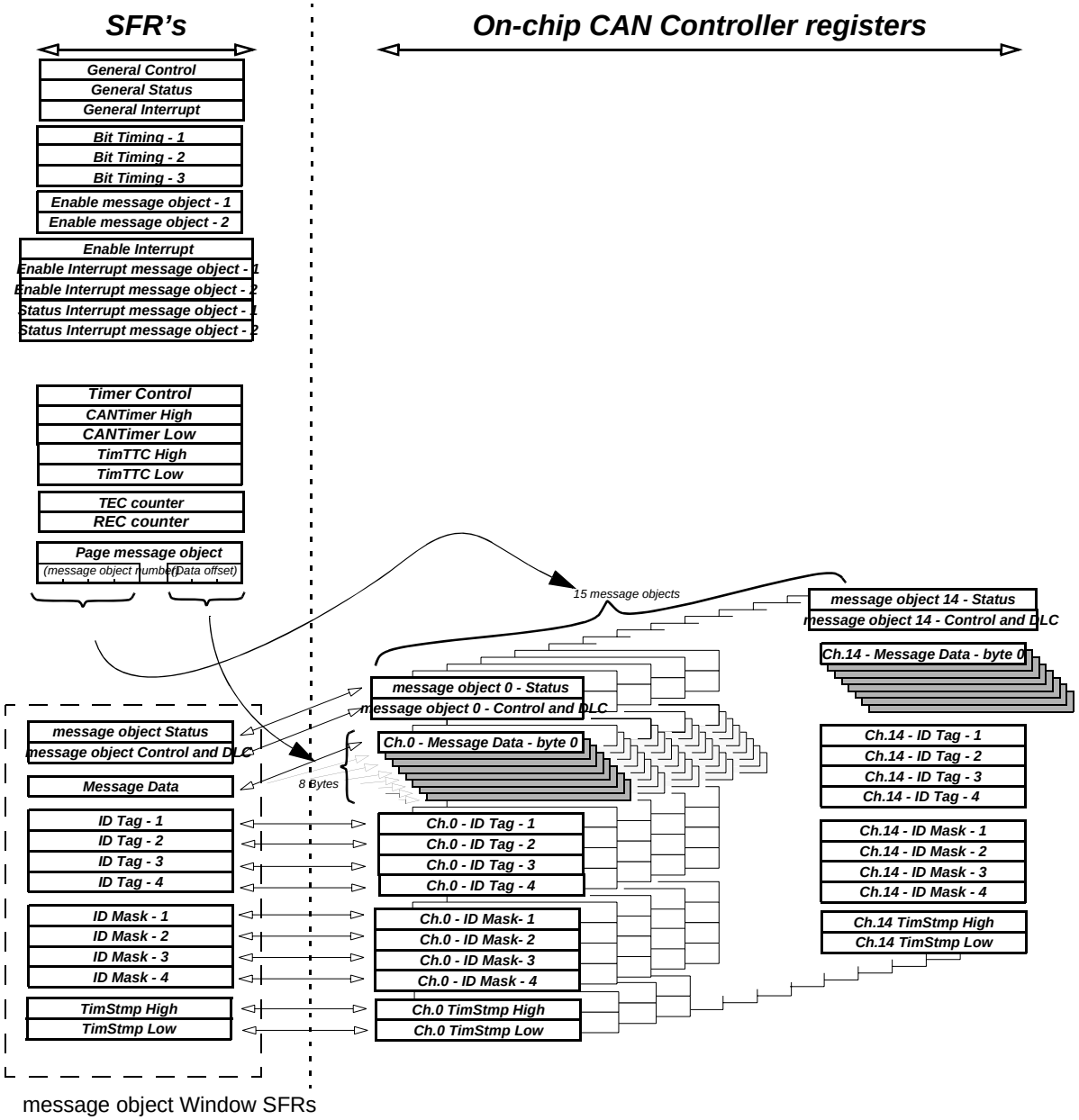



## CAN Controller Mailbox and Registers Organization

The pagination allows management of the 321 registers including 300(15x20) Bytes of mailbox via 34 SFR's.

All actions on the message object window SFRs apply to the corresponding message object registers pointed by the message object number find in the Page message object register (CANPAGE) as illustrate in Figure 48.

**Figure 48.** CAN Controller Memory Organization



### Working on Message Objects

The Page message object register (CANPAGE) is used to select one of the 15 message objects. Then, message object Control (CANCONCH) and message object Status (CANSTCH) are available for this selected message object number in the corresponding SFRs. A single register (CANMSG) is used for the message. The mailbox pointer is managed by the Page message object register with an auto-incrementation at the end of each access. The range of this counter is 8.
Note that the maibox is a pure RAM, dedicated to one message object, without overlap. In most cases, it is not necessary to transfer the received message into the standard memory. The message to be transmitted can be built directly in the maibox. Most calculations or tests can be executed in the mailbox area which provide quicker access.

## CAN Controller Management

In order to enable the CAN Controller correctly the following registers have to be initialized:

- General Control (CANGCON),
- Bit Timing (CANBT 1, 2 and 3),
- And for each page of 15 message objects
  - message object Control (CANCONCH),
  - message object Status (CANSTCH).

During operation, the CAN Enable message object registers 1 and 2 (CANEN 1 and 2) gives a fast overview of the message objects availability.

The CAN messages can be handled by interrupt or polling modes.

A message object can be configured as follows:

- Transmit message object,
- Receive message object,
- Receive buffer message object.
- Disable

This configuration is made in the CONCH1:2 field of the CANCONCH register (see Table 46).

When a message object is configured, the corresponding ENCH bit of CANEN 1 and 2 register is set.

**Table 46.** Configuration for CONCH1:2

| CONCH 1 | CONCH 2 | Type of Message Object |
|---|---|---|
| 0 | 0 | Disable |
| 0 | 1 | Transmitter |
| 1 | 0 | Receiver |
| 1 | 1 | Receiver buffer |

When a Transmitter or Receiver action of a message object is completed, the corresponding ENCH bit of the CANEN 1 and 2 register is cleared. In order to re-enable the message object, it is necessary to re-write the configuration in CANCONCH register.

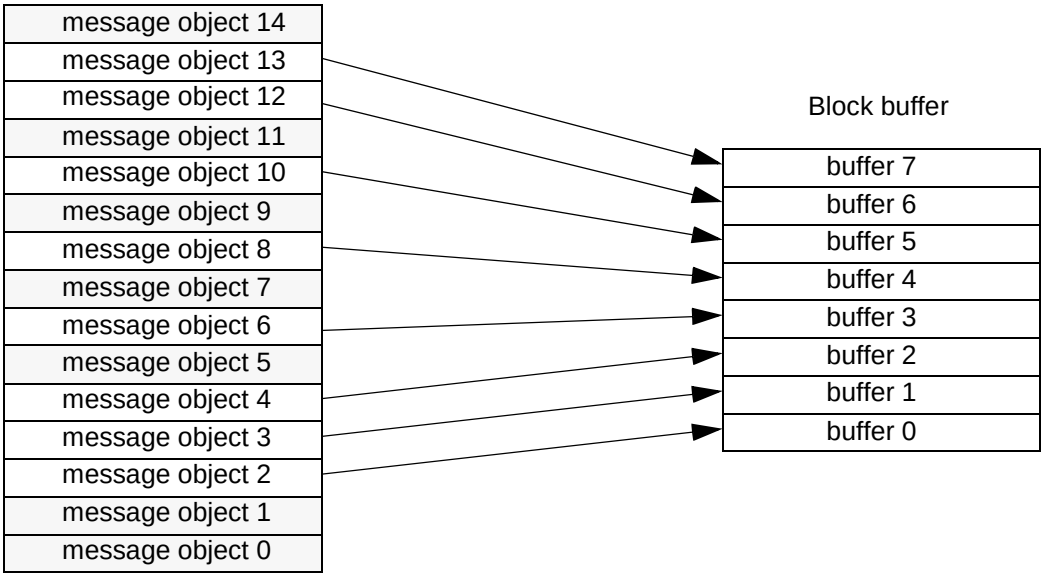
Non-consecutive message objects can be used for all three types of message objects (Transmitter, Receiver and Receiver buffer),

**Buffer Mode**

Any message object can be used to define one buffer, including non-consecutive message objects, and with no limitation in number of message objects used up to 15.

Each message object of the buffer must be initialized CONCH2 = 1 and CONCH1 = 1;

**Figure 49.** Buffer mode




The same acceptance filter must be defined for each message objects of the buffer. When there is no mask on the identifier or the IDE, all messages are accepted.

A received frame will always be stored in the lowest free message object.

When the flag Rxok is set on one of the buffer message objects, this message object can then be read by the application. This flag must then be cleared by the software and the message object re-enabled in buffer reception in order to free the message object.

The OVRBUF flag in the CANGIT register is set when the buffer is full. This flag can generate an interrupt.

The frames following the buffer-full interrupt will not stored and no status will be overwritten in the CANSTCH registers involved in the buffer until at least one of the buffer message objects is re-enabled in reception.

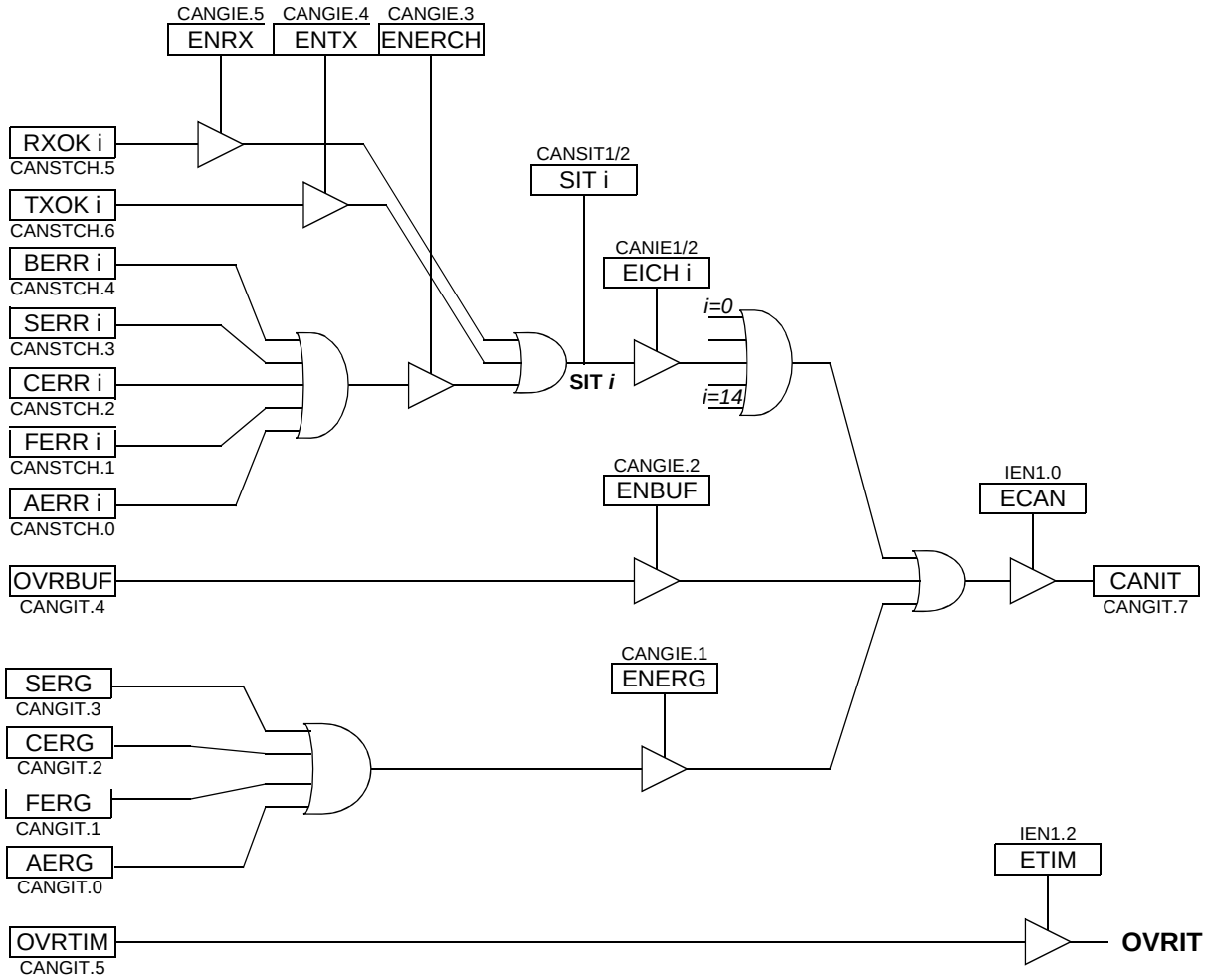
This flag must be cleared by the software in order to acknowledge the interrupt.

## IT CAN Management

The different interrupts are:

- Transmission interrupt,
- Reception interrupt,
- Interrupt on error (bit error, stuff error, crc error, form error, acknowledge error),
- Interrupt when Buffer receive is full,
- Interrupt on overrun of CAN Timer.

**Figure 50.** CAN Controller Interrupt Structure




To enable a transmission interrupt:

- Enable General CAN IT in the interrupt system register,
- Enable interrupt by message object, EICHi,
- Enable transmission interrupt, ENTX.

To enable a reception interrupt:

- Enable General CAN IT in the interrupt system register,
- Enable interrupt by message object, EICHi,
- Enable reception interrupt, ENRX.

To enable an interrupt on message object error:

- Enable General CAN IT in the interrupt system register,
- Enable interrupt by message object, EICHi,
- Enable interrupt on error, ENERCH.

To enable an interrupt on general error:

- Enable General CAN IT in the interrupt system register,
- Enable interrupt on error, ENERG.

To enable an interrupt on Buffer-full condition:

- Enable General CAN IT in the interrupt system register,
- Enable interrupt on Buffer full, ENBUF.

To enable an interrupt when Timer overruns:

- Enable Overrun IT in the interrupt system register.

When an interrupt occurs, the corresponding message object bit is set in the SIT register.

To acknowledge an interrupt, the corresponding CANSTCH bits (RXOK, TXOK,...) or CANGIT bits (OVRTIM, OVRBUF,...), must be cleared by the software application.

When the CAN node is in transmission and detects a Form Error in its frame, a bit Error will also be raised. Consequently, two consecutive interrupts can occur, both due to the same error.

When a message object error occurs and is set in CANSTCH register, no general error are set in CANGIE register.

## Bit Timing and Baud Rate

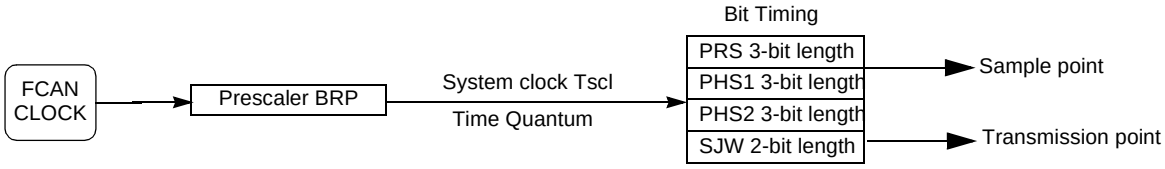
FSM's (Finite State Machine) of the CAN channel need to be synchronous to the time quantum. So, the input clock for bit timing is the clock used into CAN channel FSM's.

Field and segment abbreviations:

- BRP: Baud Rate Prescaler.
- TQ: Time Quantum (output of Baud Rate Prescaler).
- SYNS: SYNchronization Segment is 1 TQ long.
- PRS: PRopagation time Segment is programmable to be 1, 2, ..., 8 TQ long.
- PHS1: PHase Segment 1 is programmable to be 1, 2, ..., 8 TQ long.
- PHS2: PHase Segment 2 is programmable to be superior or equal to the INFORMATION PROCESSING TIME and inferior or equal to TPSH1.
- INFORMATION PROCESSING TIME is 2 TQ.
- SJW: (Re) Synchronization Jump Width is programmable to be minimum of PHS1 and 4.

The total number of TQ in a bit time has to be programmed at least from 8 to 25.

**Figure 51.** Sample And Transmission Point



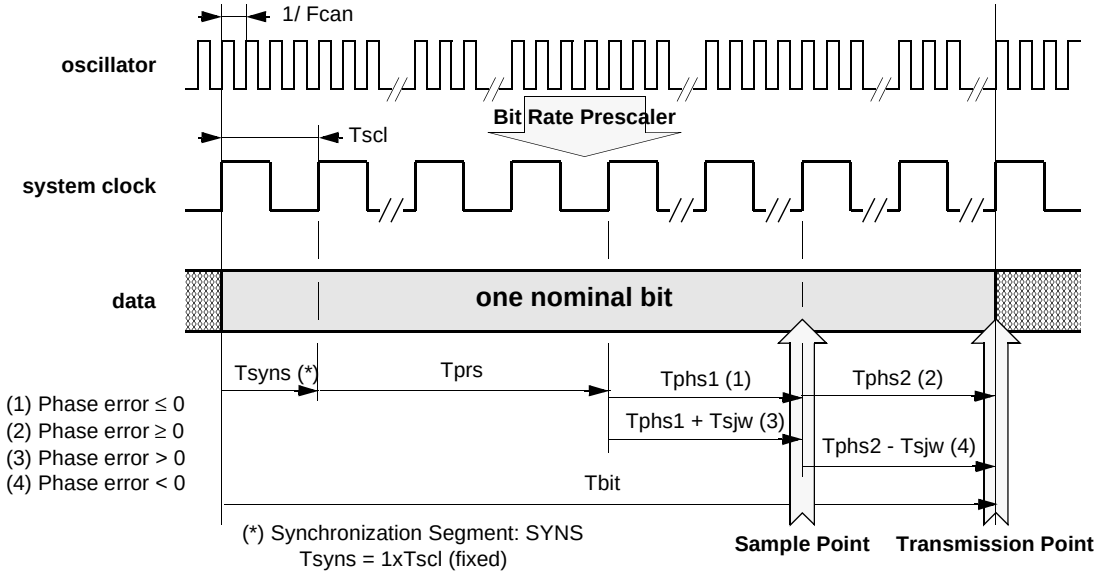

The baud rate selection is made by Tbit calculation:

Tbit = Tsyns + Tprs + Tphs1 + Tphs2

1. Tsyns = Tscl = (BRP[5..0]+ 1)/Fcan = 1TQ.
2. Tprs = (1 to 8) * Tscl = (PRS[2..0]+ 1) * Tscl
3. Tphs1 = (1 to 8) * Tscl = (PHS1[2..0]+ 1) * Tscl
4. Tphs2 = (1 to 8) * Tscl = (PHS2[2..0]+ 1) * Tscl
   **Tphs2 = Max of (Tphs1 and 2TQ)**
5. Tsjw = (1 to 4) * Tscl = (SJW[1..0]+ 1) * Tscl

The total number of Tscl (Time Quanta) in a bit time must be comprised between **8 to 25.**

**Figure 52.** General Structure of a Bit Period




Tbit calculation: $Tbit = Tsyns + Tprs + Tphs1 + Tphs2$

**example of bit timing determination for CAN baudrate of 500kbit/s:**
Fosc = 12 MHz in X1 mode => FCAN = 6 MHz

Verify that the CAN baud rate you want is an integer division of FCAN clock.
FCAN/CAN baudrate = 6 MHz/500 kHz = 12
The time quanta TQ must be comprised between 8 and 25: TQ = 12 and **BRP = 0**

Define the various timing parameters: Tbit = Tsyns + Tprs + Tphs1 + Tphs2 = 12TQ
Tsyns = 1TQ and Tsjw =1TQ => **SJW = 0**
If we chose a sample point at 66.6% => Tphs2 = 4TQ => **PHS2 = 3**
Tbit = 12 = 4 + 1 + Tphs1 + Tprs, let us choose Tprs = 3 Tphs1 = 4
**PHS1 = 3** and **PRS = 2**

BRP = 0 so CANBT1 = 00h
SJW = 0 and PRS = 2 so CANBT2 = 04h
PHS2 = 3 and PHS1 = 3 so CANBT3 = 36h

## Fault Confinement

With respect to fault confinement, a unit may be in one of the three following status:

- error active
- error passive
- bus off

An error active unit takes part in bus communication and can send an active error frame when the CAN macro detects an error.

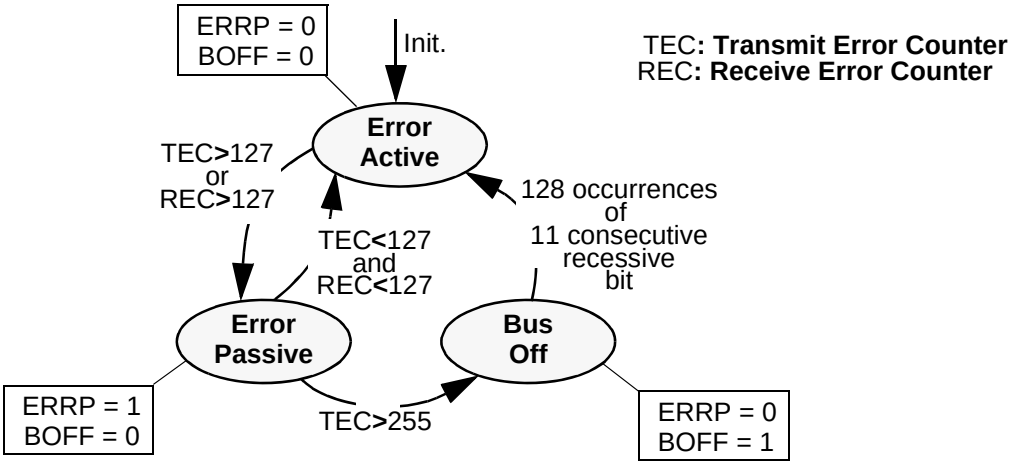
An error passive unit cannot send an active error frame. It takes part in bus communication, but when an error is detected, a passive error frame is sent. Also, after a transmission, an error passive unit will wait before initiating further transmission.

A bus off unit is not allowed to have any influence on the bus.

For fault confinement, two error counters (TEC and REC) are implemented.

See CAN Specification for details on Fault confinement.

**Figure 53.** Line Error Mode



## Acceptance Filter

Upon a reception hit (i.e., a good comparison between the ID+RTR+RB+IDE received and an ID+RTR+RB+IDE specified while taking the comparison mask into account) the ID+RTR+RB+IDE received are written over the ID TAG Registers.

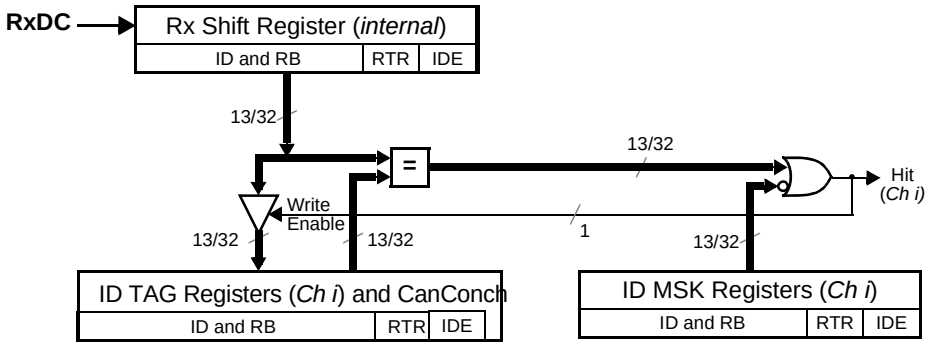
ID => IDT0-29

RTR => RTRTAG

RB => RB0-1TAG

IDE => IDE in CANCONCH register

**Figure 54.** Acceptance filter block diagram



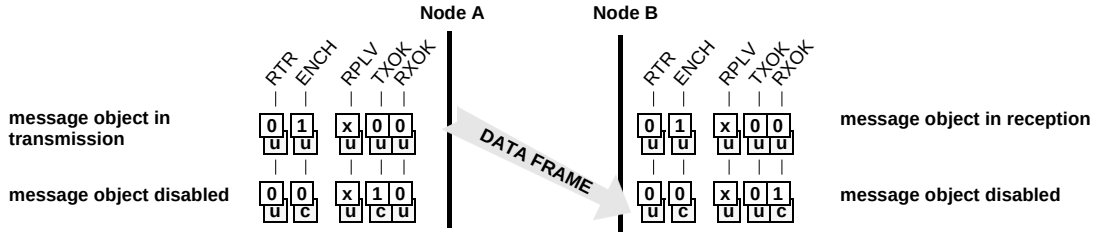

example:
To accept only ID = 318h in part A.
ID MSK = 111 1111 1111 b
ID TAG = 011 0001 1000 b

CAN SFRs

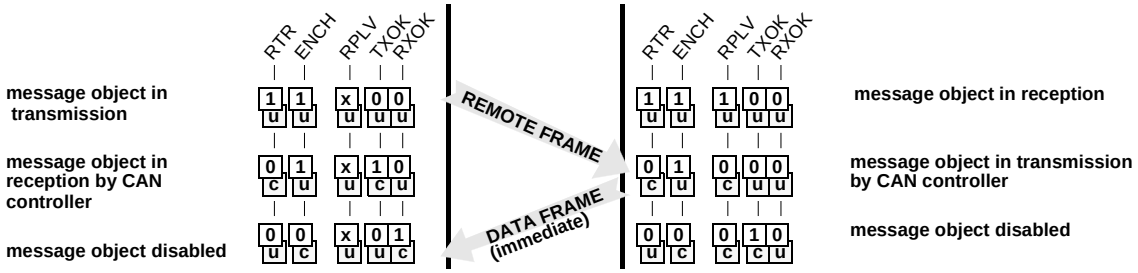
## Data and Remote Frame

Description of the different steps for:

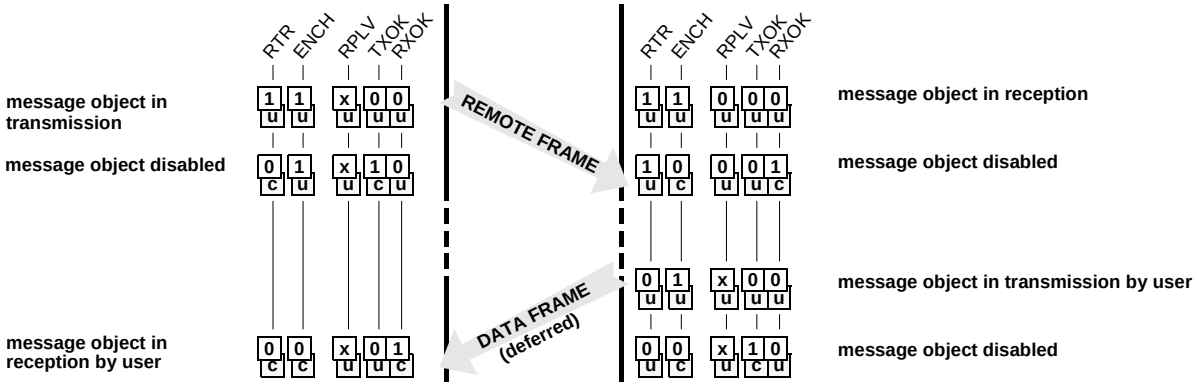
- Data Frame




- Remote Frame, With Automatic Reply,




- Remote Frame




i u : modified by user  i c : modified by CAN

## Time Trigger Communication (TTC) and Message Stamping

The AT89C51CC03 has a programmable 16-bit Timer (CANTIMH and CANTIML) for message stamp and TTC.

This CAN Timer starts after the CAN controller is enabled by the ENA bit in the CANGCON register.

Two modes in the timer are implemented:

- Time Trigger Communication:
  - Capture of this timer value in the CANTTCH and CANTTCL registers on Start Of Frame (SOF) or End Of Frame (EOF), depending on the SYNCTTC bit in the CANGCON register, when the network is configured in TTC by the TTC bit in the CANGCON register.

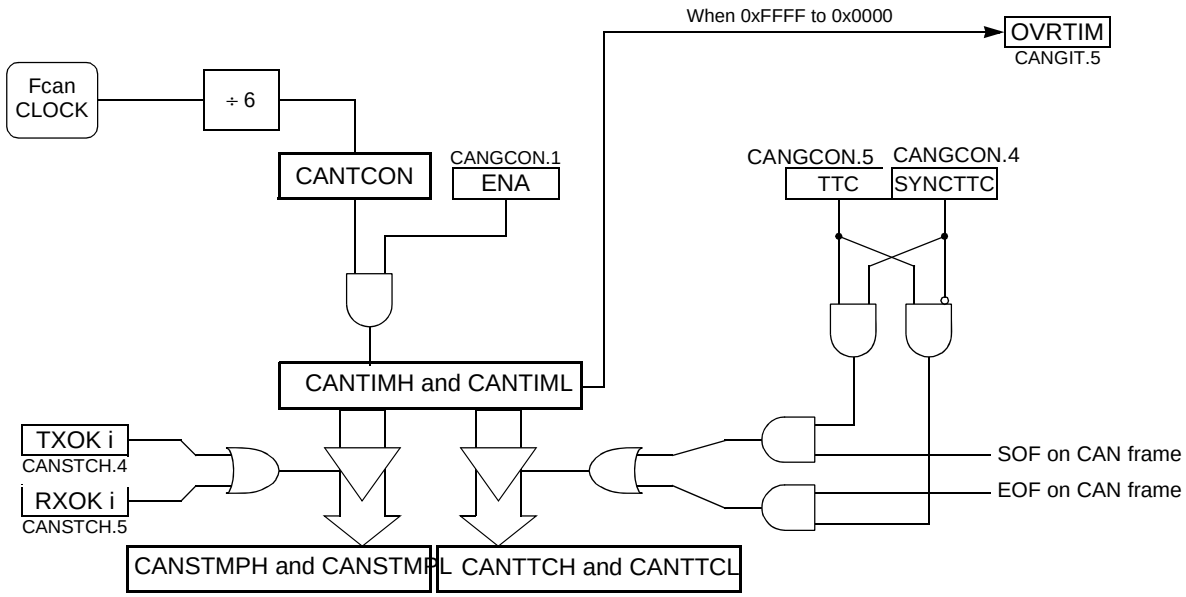
Note: In this mode, CAN ***only sends the frame once, even if an error occurs***.

- Message Stamping
  - Capture of this timer value in the CANSTMPH and CANSTMPL registers of the message object which received or sent the frame.
  - All messages can be stamps.
  - The stamping of a received frame occurs when the RxOk flag is set.
  - The stamping of a sent frame occurs when the TxOk flag is set.

The CAN Timer works in a roll-over from FFFFh to 0000h which serves as a time base.

When the timer roll-over from FFFFh to 0000h, an interrupt is generated if the ETIM bit in the interrupt enable register IEN1 is set.

**Figure 55.** Block Diagram of CAN Timer



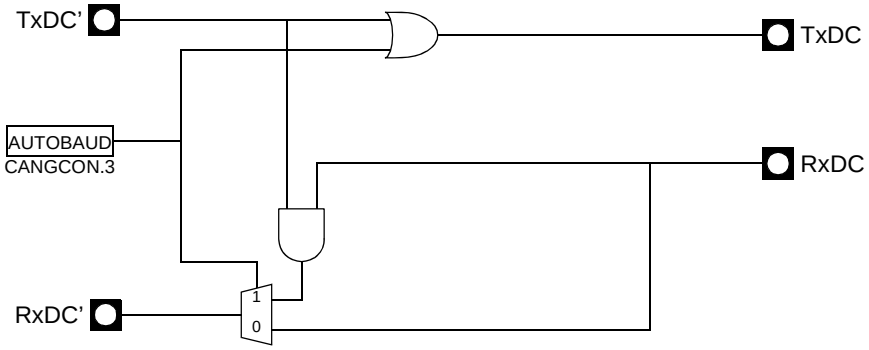
## CAN Autobaud and Listening Mode

To activate the Autobaud feature, the AUTOBAUD bit in the CANGCON register must be set. In this mode, the CAN controller is only listening to the line without acknowledging the received messages. It cannot send any message. The error flags are updated. The bit timing can be adjusted until no error occurs (good configuration find).

In this mode, the error counters are frozen.

To go back to the standard mode, the AUTOBAUD bit must be cleared.

**Figure 56.** Autobaud Mode




## Routines Examples

1. Init of CAN macro

```
// Reset the CAN macro
 CANGCON = 01h;
// Disable CAN interrupts
 ECAN   = 0;
 ETIM   = 0;
// Init the Mailbox
 for num_page =0; num_page <15; num_page++
{
   CANPAGE = num_channel << 4;
   CANCONCH = 00h
   CANSTCH = 00h;
   CANIDT1 = 00h;
   CANIDT2 = 00h;
   CANIDT3 = 00h;
   CANIDT4 = 00h;
   CANIDM1 = 00h;
   CANIDM2 = 00h;
   CANIDM3 = 00h;
   CANIDM4 = 00h;
   for num_data =0; num_data <8; num_data++)
    {
    CANMSG = 00h;
        }
}
// Configure the bit timing
 CANBT1 = xxh
 CANBT2 = xxh
 CANBT3 = xxh
```

```
// Enable the CAN macro
 CANGCON = 02h
```

2. Configure message object 3 in reception to receive only standard (11-bit identifier) message 100h

```
// Select the message object 3
 CANPAGE = 30h
// Enable the interrupt on this message object
 CANIE2 = 08h
// Clear the status and control register
 CANSTCH = 00h
 CANCONCH = 00h
// Init the acceptance filter to accept only message 100h in standard mode
 CANIDT1 = 20h
 CANIDT2 = 00h
 CANIDT3 = 00h
 CANIDT4 = 00h
 CANIDM1 = FFh
 CANIDM2 = FFh
 CANIDM3 = FFh
 CANIDM4 = FFh
// Enable channel in reception
 CANCONCH = 88h // enable reception
```

Note: To enable the CAN interrupt in reception:

```
EA = 1
ECAN = 1
CANGIE = 20h
```

3. Send a message on the message object 12

```
// Select the message object 12
 CANPAGE = C0h
// Enable the interrupt on this message object
 CANIE1 = 01h
// Clear the Status register
 CANSTCH = 00h;
// load the identifier to send (ex: 555h)
 CANIDT1 = AAh;
 CANIDT2 = A0h;
// load data to send
 CANMSG = 00h
 CANMSG = 01h
 CANMSG = 02h
 CANMSG = 03h
 CANMSG = 04h
 CANMSG = 05h
 CANMSG = 06h
 CANMSG = 07h
// configure the control register
 CANCONCH = 18h
```

4. Interrupt routine

```
// Save the current CANPAGE

// Find the first message object which generate an interrupt in CANSIT1 and CANSIT2

// Select the corresponding message object

// Analyse the CANSTCH register to identify which kind of interrupt is generated

// Manage the interrupt

// Clear the status register CANSTCH = 00h;

// if it is not a channel interrupt but a general interrupt
// Manage the general interrupt and clear CANGIT register

// restore the old CANPAGE
```

## CAN SFR's

**Table 47.** CAN SFR's With Reset Values

| | 0/8[1] | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F | |
|---|---|---|---|---|---|---|---|---|---|
| **F8h** | IPL1<br>xxxx x000 | CH<br>0000 0000 | CCAP0H<br>0000 0000 | CCAP1H<br>0000 0000 | CCAP2H<br>0000 0000 | CCAP3H<br>0000 0000 | CCAP4H<br>0000 0000 | | **FFh** |
| **F0h** | B<br>0000 0000 | | ADCLK<br>xx00 x000 | ADCON<br>0000 0000 | ADDL<br>xxxx xx00 | ADDH<br>0000 0000 | ADCF<br>0000 0000 | IPH1<br>xxxx x000 | **F7h** |
| **E8h** | IEN1<br>xxxx x000 | CL<br>0000 0000 | CCAP0L<br>0000 0000 | CCAP1L<br>0000 0000 | CCAP2L<br>0000 0000 | CCAP3L<br>0000 0000 | CCAP4L<br>0000 0000 | | **EFh** |
| **E0h** | ACC<br>0000 0000 | | | | | | | | **E7h** |
| **D8h** | CCON<br>00xx xx00 | CMOD<br>00xx x000 | CCAPM0<br>x000 0000 | CCAPM1<br>x000 0000 | CCAPM2<br>x000 0000 | CCAPM3<br>x000 0000 | CCAPM4<br>x000 0000 | | **DFh** |
| **D0h** | PSW<br>0000 0000 | FCON<br>0000 0000 | EECON<br>xxxx xx00 | FSTA<br>xxxx xx00 | SPCON<br>0001 0100 | SPSCR<br>0000 0000 | SPDAT<br>xxxx xxxx | | **D7h** |
| **C8h** | T2CON<br>0000 0000 | T2MOD<br>xxxx xx00 | RCAP2L<br>0000 0000 | RCAP2H<br>0000 0000 | TL2<br>0000 0000 | TH2<br>0000 0000 | CANEN1<br>xx00 0000 | CANEN2<br>0000 0000 | **CFh** |
| **C0h** | P4<br>xxxx xx11 | CANGIE<br>0000 0000 | CANIE1<br>xx00 0000 | CANIE2<br>0000 0000 | CANIDM1<br>xxxx xxxx | CANIDM2<br>xxxx xxxx | CANIDM3<br>xxxx xxxx | CANIDM4<br>xxxx xxxx | **C7h** |
| **B8h** | IPL0<br>x000 0000 | SADEN<br>0000 0000 | CANSIT1<br>0x00 0000 | CANSIT2<br>0000 0000 | CANIDT1<br>xxxx xxxx | CANIDT2<br>xxxx xxxx | CANIDT3<br>xxxx xxxx | CANIDT4<br>xxxx xxxx | **BFh** |
| **B0h** | P3<br>1111 1111 | CANPAGE<br>0000 0000 | CANSTCH<br>xxxx xxxx | CANCONCH<br>xxxx xxxx | CANBT1<br>xxxx xxxx | CANBT2<br>xxxx xxxx | CANBT3<br>xxxx xxxx | IPH0<br>x000 0000 | **B7h** |
| **A8h** | IEN0<br>0000 0000 | SADDR<br>0000 0000 | CANGSTA<br>0000 0000 | CANGCON<br>0000 x000 | CANTIML<br>0000 0000 | CANTIMH<br>0000 0000 | CANSTMPL<br>0000 0000 | CANSTMPH<br>0000 0000 | **AFh** |
| **A0h** | P2<br>1111 1111 | CANTCON<br>0000 0000 | AUXR1<br>xxxx 00x0 | CANMSG<br>xxxx xxxx | CANTTCL<br>0000 0000 | CANTTCH<br>0000 0000 | WDTRST<br>1111 1111 | WDTPRG<br>xxxx x000 | **A7h** |
| **98h** | SCON<br>0000 0000 | SBUF<br>0000 0000 | | CANGIT<br>0x00 0000 | CANTEC<br>0000 0000 | CANREC<br>0000 0000 | | CKCON1<br>xxxx xxx0 | **9Fh** |
| **90h** | P1<br>1111 1111 | | | | | | | | **97h** |
| **88h** | TCON<br>0000 0000 | TMOD<br>0000 0000 | TL0<br>0000 0000 | TL1<br>0000 0000 | TH0<br>0000 0000 | TH1<br>0000 0000 | AUXR<br>X001 0100 | CKCON<br>0000 0000 | **8Fh** |
| **80h** | P0<br>1111 1111 | SP<br>0000 0111 | DPL<br>0000 0000 | DPH<br>0000 0000 | | | | PCON<br>0000 0000 | **87h** |
| | **0/8[1]** | **1/9** | **2/A** | **3/B** | **4/C** | **5/D** | **6/E** | **7/F** | |

**Table 48.** CANGCON Register

CANGCON (S:ABh)
CAN General Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **ABRQ** | **OVRQ** | **TTC** | **SYNCTTC** | **AUTOBAUD** | **TEST** | **ENA** | **GRES** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | ABRQ | **Abort Request**<br>Not an auto-resetable bit. A reset of the ENCH bit (message object control and DLC register) is done for each message object. The pending transmission communications are immediately aborted but the on-going communication will be terminated normally, setting the appropriate status flags, TXOK or RXOK. |
| 6 | OVRQ | **Overload frame request (initiator)**<br>Auto-resetable bit.<br>Set to send an overload frame after the next received message.<br>Cleared by the hardware at the beginning of transmission of the overload frame. |
| 5 | TTC | **Network in Timer Trigger Communication**<br>set to select node in TTC.<br>clear to disable TTC features. |
| 4 | SYNCTTC | **Synchronization of TTC**<br>When this bit is set the TTC timer is caught on the last bit of the End Of Frame.<br>When this bit is clear the TTC timer is caught on the Start Of Frame.<br>This bit is only used in the TTC mode. |
| 3 | AUTOBAUD | **AUTOBAUD**<br>Set to activate listening mode.<br>Clear to disable listening mode |
| 2 | TEST | Test mode. The test mode is intended for factory testing and not for customer use. |
| 1 | ENA/$\overline{\text{STB}}$ | **Enable/Standby CAN Controller**<br>When this bit is set, it enables the CAN controller and its input clock.<br>When this bit is clear, the on-going communication is terminated normally and the CAN controller state of the machine is frozen (the ENCH bit of each message object does not change).<br>In the standby mode, the transmitter constantly provides a recessive level; the receiver is not activated and the input clock is stopped in the CAN controller. During the disable mode, the registers and the mailbox remain accessible.<br>Note that two clock periods are needed to start the CAN controller state of the machine. |
| 0 | GRES | **General Reset (software reset)**<br>Auto-resetable bit. This reset command is ‘ORed’ with the hardware reset in order to reset the controller. After a reset, the controller is disabled. |

Reset Value = 0000 0x00b

**Table 49.** CANGSTA Register

CANGSTA (S:AAh Read Only)
CAN General Status Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | OVFG | - | TBSY | RBSY | ENFG | BOFF | ERRP |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The values read from this bit is indeterminate. Do not set this bit. |
| 6 | OVFG | **Overload Frame Flag**<br>This status bit is set by the hardware as long as the produced overload frame is sent.<br>This flag does not generate an interrupt |
| 5 | - | **Reserved**<br>The values read from this bit is indeterminate. Do not set this bit. |
| 4 | TBSY | **Transmitter Busy**<br>This status bit is set by the hardware as long as the CAN transmitter generates a frame (remote, data, overload or error frame) or an ack field. This bit is also active during an InterFrame Spacing if a frame must be sent.<br>This flag does not generate an interrupt. |
| 3 | RBSY | **Receiver Busy**<br>This status bit is set by the hardware as long as the CAN receiver acquires or monitors a frame.<br>This flag does not generate an interrupt. |
| 2 | ENFG | **Enable On-chip CAN Controller Flag**<br>Because an enable/disable command is not effective immediately, this status bit gives the true state of a chosen mode.<br>This flag does not generate an interrupt. |
| 1 | BOFF | **Bus Off Mode**<br>**see Figure 53** |
| 0 | ERRP | **Error Passive Mode**<br>**see Figure 53** |

Reset Value = x0x0 0000b

**Table 50.** CANGIT Register

CANGIT (S:9Bh)
CAN General Interrupt

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **CANIT** | **-** | **OVRTIM** | **OVRBUF** | **SERG** | **CERG** | **FERG** | **AERG** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | CANIT | **General Interrupt Flag**[1]<br>This status bit is the image of all the CAN controller interrupts sent to the interrupt controller.<br>It can be used in the case of the polling method. |
| 6 | - | **Reserved**<br>The values read from this bit is indeterminate. Do not set this bit. |
| 5 | OVRTIM | **Overrun CAN Timer**<br>This status bit is set when the CAN timer switches 0xFFFF to 0x0000.<br>If the bit ETIM in the IE1 register is set, an interrupt is generated.<br>Clear this bit in order to reset the interrupt. |
| 4 | OVRBUF | **Overrun BUFFER**<br>0 - no interrupt.<br>1 - IT turned on<br>This bit is set when the buffer is full.<br>Bit resetable by user.<br>see Figure 50. |
| 3 | SERG | **Stuff Error General**<br>Detection of more than five consecutive bits with the same polarity.<br>This flag can generate an interrupt. resetable by user. |
| 2 | CERG | **CRC Error General**<br>The receiver performs a CRC check on each destuffed received message from the start of frame up to the data field.<br>If this checking does not match with the destuffed CRC field, a CRC error is set.<br>This flag can generate an interrupt. resetable by user. |
| 1 | FERG | **Form Error General**<br>The form error results from one or more violations of the fixed form in the following bit fields:<br>CRC delimiter<br>acknowledgment delimiter<br>end_of_frame<br>This flag can generate an interrupt. resetable by user. |
| 0 | AERG | **Acknowledgment Error General**<br>No detection of the dominant bit in the acknowledge slot.<br>This flag can generate an interrupt. resetable by user. |

Note: 1. This field is Read Only.

Reset Value = 0x00 0000b

**Table 51.** CANTEC Register

CANTEC (S:9Ch Read Only)
CAN Transmit Error Counter

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **TEC7** | **TEC6** | **TEC5** | **TEC4** | **TEC3** | **TEC2** | **TEC1** | **TEC0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | TEC7:0 | **Transmit Error Counter see Figure 53** |

Reset Value = 00h

**Table 52.** CANREC Register

CANREC (S:9Dh Read Only)
CAN Reception Error Counter

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **REC7** | **REC6** | **REC5** | **REC4** | **REC3** | **REC2** | **REC1** | **REC0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | REC7:0 | **Reception Error Counter see Figure 53** |

Reset Value = 00h

**Table 53.** CANGIE Register

CANGIE (S:C1h)
CAN General Interrupt Enable

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | ENRX | ENTX | ENERCH | ENBUF | ENERG | - |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-6 | - | **Reserved**<br>The values read from these bits are indeterminate. Do not set these bits. |
| 5 | ENRX | **Enable Receive Interrupt**<br>0 - Disable<br>1 - Enable |
| 4 | ENTX | **Enable Transmit Interrupt**<br>0 - Disable<br>1 - Enable |
| 3 | ENERCH | **Enable Message Object Error Interrupt**<br>0 - Disable<br>1 - Enable |
| 2 | ENBUF | **Enable BUF Interrupt**<br>0 - Disable<br>1 - Enable |
| 1 | ENERG | **Enable General Error Interrupt**<br>0 - Disable<br>1 - Enable |
| 0 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |

Note: See Figure 50

Reset Value = xx00 000xb

**Table 54.** CANEN1 Register

CANEN1 (S:CEh Read Only)
CAN Enable Message Object Registers 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | ENCH14 | ENCH13 | ENCH12 | ENCH11 | ENCH10 | ENCH9 | ENCH8 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The values read from this bit is indeterminate. Do not set this bit. |
| 6-0 | ENCH14:8 | **Enable Message Object**<br>These bits provide the availability of the MOb.<br>It is set to one when the MOb is enabled.<br>Once TXOK or RXOK is set to one (TXOK for automatic reply), the corresponding ENMOB is reset. ENMOB is also set to zero configuring the MOb in disabled mode, applying abortion or standby mode.<br>0 - message object disabled: MOb available for a new transmission or reception.<br>1 - message object enabled: MOb in use.<br>This bit is resetable by re-writing the CANCONCH of the corresponding message object. |

Reset Value = x000 0000b

**Table 55.** CANEN2 Register

CANEN2 (S:CFh Read Only)
CAN Enable Message Object Registers 2

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ENCH7 | ENCH6 | ENCH5 | ENCH4 | ENCH3 | ENCH2 | ENCH1 | ENCH0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | ENCH7:0 | **Enable Message Object**<br>These bits provide the availability of the MOb.<br>It is set to one when the MOb is enabled.<br>Once TXOK or RXOK is set to one (TXOK for automatic reply), the corresponding ENMOB is reset. ENMOB is also set to zero configuring the MOb in disabled mode, applying abortion or standby mode.<br>0 - message object disabled: MOb available for a new transmission or reception.<br>1 - message object enabled: MOb in use.<br>This bit is resetable by re-writing the CANCONCH of the corresponding message object. |

Reset Value = 0000 0000b

**Table 56.** CANSIT1 Register

CANSIT1 (S:BAh Read Only)
CAN Status Interrupt Message Object Registers 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | SIT14 | SIT13 | SIT12 | SIT11 | SIT10 | SIT9 | SIT8 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The values read from this bit is indeterminate. Do not set this bit. |
| 6-0 | SIT14:8 | **Status of Interrupt by Message Object**<br>0 - no interrupt.<br>1 - IT turned on. Reset when interrupt condition is cleared by user.<br>SIT14:8 = 0b 0000 1001 -> IT's on message objects 11 and 8.<br>see Figure 50. |

Reset Value = x000 0000b

**Table 57.** CANSIT2 Register

CANSIT2 (S:BBh Read Only)

CAN Status Interrupt Message Object Registers 2

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SIT7 | SIT6 | SIT5 | SIT4 | SIT3 | SIT2 | SIT1 | SIT0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | SIT7:0 | **Status of Interrupt by Message Object**<br>0 - no interrupt.<br>1 - IT turned on. Reset when interrupt condition is cleared by user.<br>SIT7:0 = 0b 0000 1001 -> IT's on message objects 3 and 0<br>see Figure 50. |

Reset Value = 0000 0000b

**Table 58.** CANIE1 Register

CANIE1 (S:C2h)
CAN Enable Interrupt Message Object Registers 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | IECH14 | IECH13 | IECH12 | IECH11 | IECH10 | IECH9 | IECH8 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The values read from this bit is indeterminate. Do not set this bit. |
| 6-0 | IECH14:8 | **Enable interrupt by Message Object**<br>0 - disable IT.<br>1 - enable IT.<br>IECH14:8 = 0b 0000 1100 -> Enable IT's of message objects 11 and 10.<br>see Figure 50. |

Reset Value = x000 0000b

**Table 59.** CANIE2 Register

CANIE2 (S:C3h)
CAN Enable Interrupt Message Object Registers 2

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IECH 7 | IECH 6 | IECH 5 | IECH 4 | IECH 3 | IECH 2 | IECH 1 | IECH 0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | IECH7:0 | **Enable interrupt by Message Object**<br>0 - disable IT.<br>1 - enable IT.<br>IECH7:0 = 0b 0000 1100 -> Enable IT's of message objects 3 and 2. |

Reset Value = 0000 0000b

**Table 60.** CANBT1 Register

CANBT1 (S:B4h)
CAN Bit Timing Registers 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | BRP 5 | BRP 4 | BRP 3 | BRP 2 | BRP 1 | BRP 0 | - |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6-1 | BRP5:0 | **Baud rate prescaler**<br>The period of the CAN controller system clock Tscl is programmable and determines the individual bit timing.<br>$Tscl = \frac{BRP[5..0] + 1}{Fcan}$ |
| 0 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |

Note: The CAN controller bit timing registers must be accessed only if the CAN controller is disabled with the ENA bit of the CANGCON register set to 0.
See Figure 52.

No default value after reset.

**Table 61.** CANBT2 Register

CANBT2 (S:B5h)
CAN Bit Timing Registers 2

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | SJW 1 | SJW 0 | - | PRS 2 | PRS 1 | PRS 0 | - |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6-5 | SJW1:0 | **Re-synchronization Jump Width**<br>To compensate for phase shifts between clock oscillators of different bus controllers, the controller must re-synchronize on any relevant signal edge of the current transmission.<br>The synchronization jump width defines the maximum number of clock cycles. A bit period may be shortened or lengthened by a re-synchronization.<br>Tsjw = Tscl x (SJW [1..0] +1) |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3-1 | PRS2:0 | **Programming Time Segment**<br>This part of the bit time is used to compensate for the physical delay times within the network. It is twice the sum of the signal propagation time on the bus line, the input comparator delay and the output driver delay.<br>Tprs = Tscl x (PRS[2..0] + 1) |
| 0 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |

Note: The CAN controller bit timing registers must be accessed only if the CAN controller is disabled with the ENA bit of the CANGCON register set to 0.
See Figure 52.

No default value after reset.

**Table 62.** CANBT3 Register

CANBT3 (S:B6h)
CAN Bit Timing Registers 3

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | PHS2 2 | PHS2 1 | PHS2 0 | PHS1 2 | PHS1 1 | PHS1 0 | SMP |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6-4 | PHS2 2:0 | **Phase Segment 2**<br>This phase is used to compensate for phase edge errors. This segment can be shortened by the re-synchronization jump width.<br>Tphs2 = Tscl x (PHS2[2..0] + 1)<br>Phase segment 2 is the maximum of Phase segment 1 and the Information Processing Time (= 2TQ). |
| 3-1 | PHS1 2:0 | **Phase Segment 1**<br>This phase is used to compensate for phase edge errors. This segment can be lengthened by the re-synchronization jump width.<br>Tphs1 = Tscl x (PHS1[2..0] + 1) |
| 0 | SMP | **Sample Type**<br>0 - once, at the sample point.<br>1 - three times, the threefold sampling of the bus is the sample point and twice over a distance of a 1/2 period of the Tscl. The result corresponds to the majority decision of the three values. |

Note: The CAN controller bit timing registers must be accessed only if the CAN controller is disabled with the ENA bit of the CANGCON register set to 0.
See Figure 52.

No default value after reset.

**Table 63.** CANPAGE Register

CANPAGE (S:B1h)
CAN Message Object Page Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CHNB 3 | CHNB 2 | CHNB 1 | CHNB 0 | AINC | INDX2 | INDX1 | INDX0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-4 | CHNB3:0 | **Selection of Message Object Number**<br>The available numbers are: 0 to 14 (see Figure 48). |
| 3 | AINC | **Auto Increment of the Index (active low)**<br>0 - auto-increment of the index (default value).<br>1 - non-auto-increment of the index. |
| 2-0 | INDX2:0 | **Index**<br>Byte location of the data field for the defined message object (see Figure 48). |

Reset Value = 0000 0000b

**Table 64.** CANCONCH Register

CANCONCH (S:B3h)
CAN Message Object Control and DLC Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CONCH 1 | CONCH 0 | RPLV | IDE | DLC 3 | DLC 2 | DLC 1 | DLC 0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-6 | CONCH1:0 | **Configuration of Message Object**<br>**CONCH1 CONCH0**<br>0 0: disable<br>0 1: Launch transmission<br>1 0: Enable Reception<br>1 1: Enable Reception Buffer<br>Note: The user must re-write the configuration to enable the corresponding bit in the CANEN1:2 registers. |
| 5 | RPLV | **Reply Valid**<br>Used in the automatic reply mode after receiving a remote frame<br>0 - reply not ready.<br>1 - reply ready and valid. |
| 4 | IDE | **Identifier Extension**<br>0 - CAN standard rev 2.0 A (ident = 11 bits).<br>1 - CAN standard rev 2.0 B (ident = 29 bits). |
| 3-0 | DLC3:0 | **Data Length Code**<br>Number of Bytes in the data field of the message.<br>The range of DLC is from 0 up to 8.<br>This value is updated when a frame is received (data or remote frame).<br>If the expected DLC differs from the incoming DLC, a warning appears in the CANSTCH register. |

No default value after reset

**Table 65.** CANSTCH Register

CANSTCH (S:B2h)
CAN Message Object Status Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DLCW | TXOK | RXOK | BERR | SERR | CERR | FERR | AERR |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | DLCW | **Data Length Code Warning**<br>The incoming message does not have the DLC expected. Whatever the frame type, the DLC field of the CANCONCH register is updated by the received DLC. |
| 6 | TXOK | **Transmit OK**<br>The communication enabled by transmission is completed.<br>When the controller is ready to send a frame, if two or more message objects are enabled as producers, the lower index message object (0 to 13) is supplied first.<br>This flag can generate an interrupt. |
| 5 | RXOK | **Receive OK**<br>The communication enabled by reception is completed.<br>In the case of two or more message object reception hits, the lower index message object (0 to 13) is updated first.<br>This flag can generate an interrupt. |
| 4 | BERR | **Bit Error (Only in Transmission)**<br>The bit value monitored is different from the bit value sent.<br>Exceptions:<br>the monitored recessive bit sent as a dominant bit during the arbitration field and the acknowledge slot detecting a dominant bit during the sending of an error frame.<br>This flag can generate an interrupt. |
| 3 | SERR | **Stuff Error**<br>Detection of more than five consecutive bits with the same polarity.<br>This flag can generate an interrupt. |
| 2 | CERR | **CRC Error**<br>The receiver performs a CRC check on each destuffed received message from the start of frame up to the data field.<br>If this checking does not match with the destuffed CRC field, a CRC error is set.<br>This flag can generate an interrupt. |
| 1 | FERR | **Form Error**<br>The form error results from one or more violations of the fixed form in the following bit fields:<br>CRC delimiter<br>acknowledgment delimiter<br>end_of_frame<br>This flag can generate an interrupt. |
| 0 | AERR | **Acknowledgment Error**<br>No detection of the dominant bit in the acknowledge slot.<br>This flag can generate an interrupt. |

Note: See Figure 50.

No default value after reset.

**Table 66.** CANIDT1 Register for V2.0 part A

CANIDT1 for V2.0 part A (S:BCh)
CAN Identifier Tag Registers 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDT 10 | IDT 9 | IDT 8 | IDT 7 | IDT 6 | IDT 5 | IDT 4 | IDT 3 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | IDT10:3 | **IDentifier tag value**<br>See Figure 54. |

No default value after reset.

**Table 67.** CANIDT2 Register for V2.0 part A

CANIDT2 for V2.0 part A (S:BDh)
CAN Identifier Tag Registers 2

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDT 2 | IDT 1 | IDT 0 | - | - | - | - | - |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-5 | IDT2:0 | **IDentifier tag value**<br>See Figure 54. |
| 4-0 | - | **Reserved**<br>The values read from these bits are indeterminate. Do not set these bits. |

No default value after reset.

**Table 68.** CANIDT3 Register for V2.0 part A

CANIDT3 for V2.0 part A (S:BEh)
CAN Identifier Tag Registers 3

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | - | **Reserved**<br>The values read from these bits are indeterminate. Do not set these bits. |

No default value after reset.

**Table 69.** CANIDT4 Register for V2.0 part A

CANIDT4 for V2.0 part A (S:BFh)
CAN Identifier Tag Registers 4

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | RTRTAG | - | RB0TAG |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-3 | - | **Reserved**<br>The values read from these bits are indeterminate. Do not set these bits. |
| 2 | RTRTAG | **Remote Transmission Request Tag Value**. |
| 1 | - | **Reserved**<br>The values read from this bit are indeterminate. Do not set these bit. |
| 0 | RB0TAG | **Reserved Bit 0 Tag Value.** |

No default value after reset.

**Table 70.** CANIDT4 Register for V2.0 part A

CANIDT1 for V2.0 part B (S:BCh)
CAN Identifier Tag Registers 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDT 28 | IDT 27 | IDT 26 | IDT 25 | IDT 24 | IDT 23 | IDT 22 | IDT 21 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | IDT28:21 | **IDentifier Tag Value**<br>See Figure 54. |

No default value after reset.

**Table 71.** CANIDT2 Register for V2.0 part B

CANIDT2 for V2.0 part B (S:BDh)
CAN Identifier Tag Registers 2

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDT 20 | IDT 19 | IDT 18 | IDT 17 | IDT 16 | IDT 15 | IDT 14 | IDT 13 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | IDT20:13 | **IDentifier Tag Value**<br>See Figure 54. |

No default value after reset.

**Table 72.** CANIDT3 Register for V2.0 part B

CANIDT3 for V2.0 part B (S:BEh)
CAN Identifier Tag Registers 3

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDT 12 | IDT 11 | IDT 10 | IDT 9 | IDT 8 | IDT 7 | IDT 6 | IDT 5 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | IDT12:5 | **IDentifier Tag Value**<br>See Figure 54. |

No default value after reset.

**Table 73.** CANIDT4 Register for V2.0 part B

CANIDT4 for V2.0 part B (S:BFh)
CAN Identifier Tag Registers 4

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDT 4 | IDT 3 | IDT 2 | IDT 1 | IDT 0 | RTRTAG | RB1TAG | RB0TAG |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-3 | IDT4:0 | **IDentifier Tag Value**<br>See Figure 54. |
| 2 | RTRTAG | **Remote Transmission Request Tag Value** |
| 1 | RB1TAG | **Reserved bit 1 Tag Value** |
| 0 | RB0TAG | **Reserved bit 0 Tag Value** |

No default value after reset.

**Table 74.** CANIDM1 Register for V2.0 part A

CANIDM1 for V2.0 part A (S:C4h)
CAN Identifier Mask Registers 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDMSK 10 | IDMSK 9 | IDMSK 8 | IDMSK 7 | IDMSK 6 | IDMSK 5 | IDMSK 4 | IDMSK 3 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | IDTMSK10:3 | **IDentifier mask value**<br>0 - comparison true forced.<br>1 - bit comparison enabled.<br>See Figure 54. |

No default value after reset.

**Table 75.** CANIDM2 Register for V2.0 part A

CANIDM2 for V2.0 part A (S:C5h)
CAN Identifier Mask Registers 2

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **IDMSK 2** | **IDMSK 1** | **IDMSK 0** | **-** | **-** | **-** | **-** | **-** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-5 | IDTMSK2:0 | **IDentifier Mask Value**<br>0 - comparison true forced.<br>1 - bit comparison enabled.<br>See Figure 54. |
| 4-0 | - | **Reserved**<br>The values read from these bits are indeterminate. Do not set these bits. |

No default value after reset.

**Table 76.** CANIDM3 Register for V2.0 part A

CANIDM3 for V2.0 part A (S:C6h)
CAN Identifier Mask Registers 3

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | - | **Reserved**<br>The values read from these bits are indeterminate. |

No default value after reset.

**Table 77.** CANIDM4 Register for V2.0 part A

CANIDM4 for V2.0 part A (S:C7h)
CAN Identifier Mask Registers 4

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | RTRMSK | - | IDEMSK |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-3 | - | **Reserved**<br>The values read from these bits are indeterminate. Do not set these bits. |
| 2 | RTRMSK | **Remote Transmission Request Mask Value**<br>0 - comparison true forced.<br>1 - bit comparison enabled. |
| 1 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 0 | IDEMSK | **IDentifier Extension Mask Value**<br>0 - comparison true forced.<br>1 - bit comparison enabled. |

Note: The ID Mask is only used for reception.

No default value after reset.

**Table 78.** CANIDM1 Register for V2.0 part B

CANIDM1 for V2.0 part B (S:C4h)
CAN Identifier Mask Registers 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDMSK 28 | IDMSK 27 | IDMSK 26 | IDMSK 25 | IDMSK 24 | IDMSK 23 | IDMSK 22 | IDMSK 21 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | IDMSK28:21 | **IDentifier Mask Value**<br>0 - comparison true forced.<br>1 - bit comparison enabled.<br>See Figure 54. |

Note: The ID Mask is only used for reception.

No default value after reset.

**Table 79.** CANIDM2 Register for V2.0 part B

CANIDM2 for V2.0 part B (S:C5h)
CAN Identifier Mask Registers 2

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDMSK 20 | IDMSK 19 | IDMSK 18 | IDMSK 17 | IDMSK 16 | IDMSK 15 | IDMSK 14 | IDMSK 13 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | IDMSK20:13 | **IDentifier Mask Value**<br>0 - comparison true forced.<br>1 - bit comparison enabled.<br>See Figure 54. |

Note: The ID Mask is only used for reception.

No default value after reset.

**Table 80.** CANIDM3 Register for V2.0 part B

CANIDM3 for V2.0 part B (S:C6h)
CAN Identifier Mask Registers 3

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDMSK 12 | IDMSK 11 | IDMSK 10 | IDMSK 9 | IDMSK 8 | IDMSK 7 | IDMSK 6 | IDMSK 5 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | IDMSK12:5 | **IDentifier Mask Value**<br>0 - comparison true forced.<br>1 - bit comparison enabled.<br>See Figure 54. |

Note: The ID Mask is only used for reception.

No default value after reset.

**Table 81.** CANIDM4 Register for V2.0 part B

CANIDM4 for V2.0 part B (S:C7h)
CAN Identifier Mask Registers 4

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDMSK 4 | IDMSK 3 | IDMSK 2 | IDMSK 1 | IDMSK 0 | RTRMSK | - | IDEMSK |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-3 | IDMSK4:0 | **IDentifier Mask Value**<br>0 - comparison true forced.<br>1 - bit comparison enabled.<br>See Figure 54. |
| 2 | RTRMSK | **Remote Transmission Request Mask Value**<br>0 - comparison true forced.<br>1 - bit comparison enabled. |
| 1 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 0 | IDEMSK | **IDentifier Extension Mask Value**<br>0 - comparison true forced.<br>1 - bit comparison enabled. |

Note: The ID Mask is only used for reception.

No default value after reset.

**Table 82.** CANMSG Register

CANMSG (S:A3h)
CAN Message Data Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSG 7 | MSG 6 | MSG 5 | MSG 4 | MSG 3 | MSG 2 | MSG 1 | MSG 0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | MSG7:0 | **Message Data**<br>This register contains the mailbox data byte pointed at the page message object register.<br>After writing in the page message object register, this byte is equal to the specified message location (in the mailbox) of the pre-defined identifier + index. If auto-incrementation is used, at the end of the data register writing or reading cycle, the mailbox pointer is auto-incremented. The range of the counting is 8 with no end loop (0, 1,..., 7, 0,...) |

No default value after reset.

**Table 83.** CANTCON Register

CANTCON (S:A1h)
CAN Timer ClockControl

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **TPRESC 7** | **TPRESC 6** | **TPRESC 5** | **TPRESC 4** | **TPRESC 3** | **TPRESC 2** | **TPRESC 1** | **TPRESC 0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | TPRESC7:0 | **Timer Prescaler of CAN Timer**<br>This register is a prescaler for the main timer upper counter<br>range = 0 to 255.<br>See Figure 55. |

Reset Value = 00h

**Table 84.** CANTIMH Register

CANTIMH (S:ADh)
CAN Timer High

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **CANGTIM 15** | **CANGTIM 14** | **CANGTIM 13** | **CANGTIM 12** | **CANGTIM 11** | **CANGTIM 10** | **CANGTIM 9** | **CANGTIM 8** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | CANGTIM15:8 | **High byte of Message Timer**<br>See Figure 55. |

Reset Value = 0000 0000b

**Table 85.** CANTIML Register

CANTIML (S:ACh)
CAN Timer Low

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **CANGTIM 7** | **CANGTIM 6** | **CANGTIM 5** | **CANGTIM 4** | **CANGTIM 3** | **CANGTIM 2** | **CANGTIM 1** | **CANGTIM 0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | CANGTIM7:0 | **Low byte of Message Timer**<br>See Figure 55. |

Reset Value = 0000 0000b

**Table 86.** CANSTMPH Register

CANSTMPH (S:AFh Read Only)
CAN Stamp Timer High

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TIMSTMP 15 | TIMSTMP 14 | TIMSTMP 13 | TIMSTMP 12 | TIMSTMP 11 | TIMSTMP 10 | TIMSTMP 9 | TIMSTMP 8 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | TIMSTMP15:8 | **High byte of Time Stamp**<br>See Figure 55. |

No default value after reset

**Table 87.** CANSTMPL Register

CANSTMPL (S:AEh Read Only)
CAN Stamp Timer Low

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TIMSTMP 7 | TIMSTMP 6 | TIMSTMP 5 | TIMSTMP 4 | TIMSTMP 3 | TIMSTMP 2 | TIMSTMP 1 | TIMSTMP 0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | TIMSTMP7:0 | **Low byte of Time Stamp**<br>See Figure 55. |

No default value after reset

**Table 88.** CANTTCH Register

CANTTCH (S:A5h Read Only)
CAN TTC Timer High

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TIMTTC 15 | TIMTTC 14 | TIMTTC 13 | TIMTTC 12 | TIMTTC 11 | TIMTTC 10 | TIMTTC 9 | TIMTTC 8 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | TIMTTC15:8 | **High byte of TTC Timer**<br>See Figure 55. |

Reset Value = 0000 0000b

**Table 89.** CANTTCL Register

CANTTCL (S:A4h Read Only)
CAN TTC Timer Low

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **TIMTTC 7** | **TIMTTC 6** | **TIMTTC 5** | **TIMTTC 4** | **TIMTTC 3** | **TIMTTC 2** | **TIMTTC 1** | **TIMTTC 0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | TIMTTC7:0 | **Low byte of TTC Timer**<br>See Figure 55. |

Reset Value = 0000 0000b

## Serial Port Interface (SPI)

The Serial Peripheral Interface Module (SPI) allows full-duplex, synchronous, serial communication between the MCU and peripheral devices, including other MCUs.

### Features

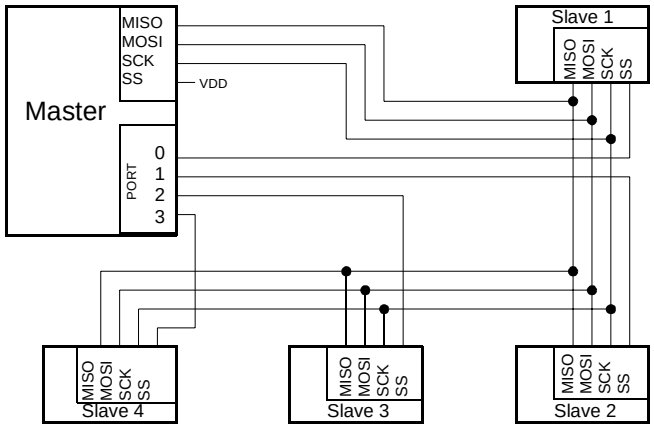
Features of the SPI Module include the following:

- Full-duplex, three-wire synchronous transfers
- Master or Slave operation
- Six programmable Master clock rates in master mode
- Serial clock with programmable polarity and phase
- Master Mode fault error flag with MCU interrupt capability

### Signal Description

Figure 57 shows a typical SPI bus configuration using one Master controller and many Slave peripherals. The bus is made of three wires connecting all the devices.

**Figure 57.** SPI Master/Slaves Interconnection




The Master device selects the individual Slave devices by using four pins of a parallel port to control the four $\overline{SS}$ pins of the Slave devices.

**Master Output Slave Input (MOSI)**

This 1-bit signal is directly connected between the Master Device and a Slave Device. The MOSI line is used to transfer data in series from the Master to the Slave. Therefore, it is an output signal from the Master, and an input signal to a Slave. A Byte (8-bit word) is transmitted most significant bit (MSB) first, least significant bit (LSB) last.

**Master Input Slave Output (MISO)**

This 1-bit signal is directly connected between the Slave Device and a Master Device. The MISO line is used to transfer data in series from the Slave to the Master. Therefore, it is an output signal from the Slave, and an input signal to the Master. A Byte (8-bit word) is transmitted most significant bit (MSB) first, least significant bit (LSB) last.

**SPI Serial Clock (SCK)**

This signal is used to synchronize the data transmission both in and out of the devices through their MOSI and MISO lines. It is driven by the Master for eight clock cycles which allows to exchange one Byte on the serial lines.

**Slave Select ($\overline{SS}$)**

Each Slave peripheral is selected by one Slave Select pin ($\overline{SS}$). This signal must stay low for any message for a Slave. It is obvious that only one Master ($\overline{SS}$ high level) can drive the network. The Master may select each Slave device by software through port pins (Figure 58). To prevent bus conflicts on the MISO line, only one slave should be selected at a time by the Master for a transmission.

In a Master configuration, the $\overline{SS}$ line can be used in conjunction with the MODF flag in the SPI Status register (SPSCR) to prevent multiple masters from driving MOSI and SCK (see Error conditions).

A high level on the $\overline{SS}$ pin puts the MISO line of a Slave SPI in a high-impedance state.

The $\overline{SS}$ pin could be used as a general-purpose if the following conditions are met:

- The device is configured as a Master and the SSDIS control bit in SPCON is set. This kind of configuration can be found when only one Master is driving the network and there is no way that the $\overline{SS}$ pin could be pulled low. Therefore, the MODF flag in the SPSCR will never be set(1).
- The Device is configured as a Slave with CPHA and SSDIS control bits set(2). This kind of configuration can happen when the system includes one Master and one Slave only. Therefore, the device should always be selected and there is no reason that the Master uses the $\overline{SS}$ pin to select the communicating Slave device.

Note: 1. Clearing SSDIS control bit does not clear MODF.

2. Special care should be taken not to set SSDIS control bit when CPHA ='0' because in this mode, the $\overline{SS}$ is used to start the transmission.

**Baud Rate**

In Master mode, the baud rate can be selected from a baud rate generator which is controlled by three bits in the SPCON register: SPR2, SPR1 and SPR0.The Master clock is selected from one of seven clock rates resulting from the division of the internal clock by 4, 8, 16, 32, 64 or 128.

Table 90 gives the different clock rates selected by SPR2:SPR1:SPR0.

In Slave mode, the maximum baud rate allowed on the SCK input is limited to $F_{sys}/4$

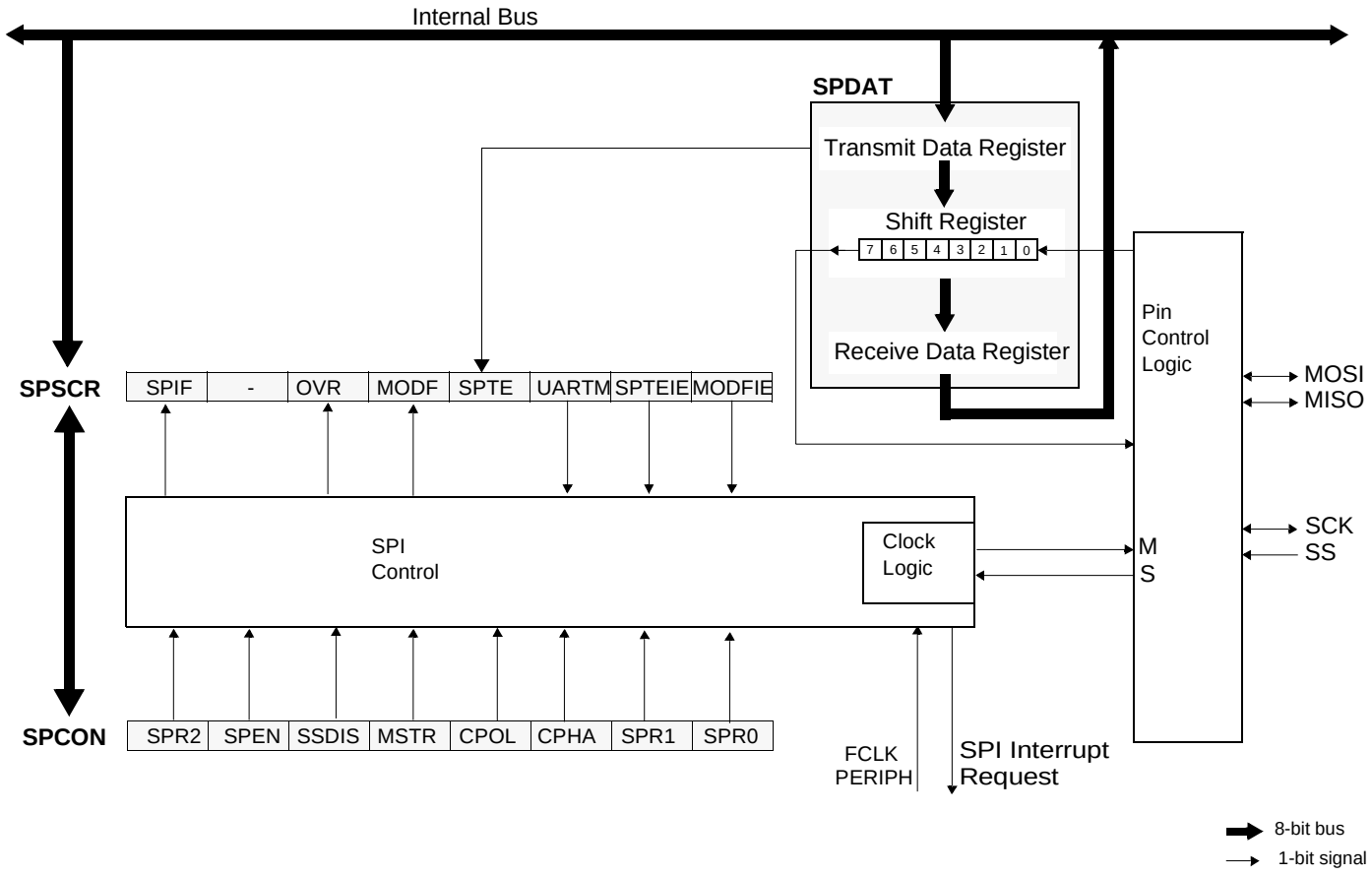
**Table 90.** SPI Master Baud Rate Selection

| SPR2 | SPR1 | SPR0 | Clock Rate | Baud Rate Divisor (BD) |
|---|---|---|---|---|
| 0 | 0 | 0 | Don't Use | No BRG |
| 0 | 0 | 1 | $F_{CLK\ PERIPH}$ /4 | 4 |
| 0 | 1 | 0 | $F_{CLK\ PERIPH}$/8 | 8 |
| 0 | 1 | 1 | $F_{CLK\ PERIPH}$ /16 | 16 |
| 1 | 0 | 0 | $F_{CLK\ PERIPH}$ /32 | 32 |
| 1 | 0 | 1 | $F_{CLK\ PERIPH}$ /64 | 64 |
| 1 | 1 | 0 | $F_{CLK\ PERIPH}$ /128 | 128 |
| 1 | 1 | 1 | Don't Use | No BRG |

## Functional Description

Figure 58 shows a detailed structure of the SPI Module.

**Figure 58.** SPI Module Block Diagram




### Operating Modes

The Serial Peripheral Interface can be configured in one of the two modes: Master mode or Slave mode. The configuration and initialization of the SPI Module is made through two registers:

- The Serial Peripheral Control register (SPCON)
- The Serial Peripheral Status and Control Register (SPSCR)

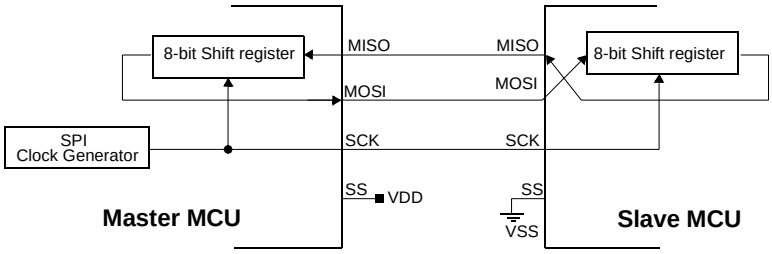
Once the SPI is configured, the data exchange is made using:

- The Serial Peripheral DATa register (SPDAT)

During an SPI transmission, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line (SCK) synchronizes shifting and sampling on the two serial data lines (MOSI and MISO). A Slave Select line ($\overline{SS}$) allows individual selection of a Slave SPI device; Slave devices that are not selected do not interfere with SPI bus activities.

When the Master device transmits data to the Slave device via the MOSI line, the Slave device responds by sending data to the Master device via the MISO line. This implies full-duplex transmission with both data out and data in synchronized with the same clock (Figure 59).

**Figure 59.** Full-Duplex Master-Slave Interconnection




*Master Mode*

The SPI operates in Master mode when the Master bit, MSTR (1), in the SPCON register is set. Only one Master SPI device can initiate transmissions. Software begins the transmission from a Master SPI Module by writing to the Serial Peripheral Data Register (SPDAT). If the shift register is empty, the Byte is immediately transferred to the shift register. The Byte begins shifting out on MOSI pin under the control of the serial clock, SCK. Simultaneously, another Byte shifts in from the Slave on the Master's MISO pin. The transmission ends when the Serial Peripheral transfer data flag, SPIF, in SPSCR becomes set. At the same time that SPIF becomes set, the received Byte from the Slave is transferred to the receive data register in SPDAT. Software clears SPIF by reading the Serial Peripheral Status register (SPSCR) with the SPIF bit set, and then reading the SPDAT.

*Slave Mode*

The SPI operates in Slave mode when the Master bit, MSTR (2), in the SPCON register is cleared. Before a data transmission occurs, the Slave Select pin, $\overline{SS}$, of the Slave device must be set to'0'. $\overline{SS}$ must remain low until the transmission is complete.

In a Slave SPI Module, data enters the shift register under the control of the SCK from the Master SPI Module. After a Byte enters the shift register, it is immediately transferred to the receive data register in SPDAT, and the SPIF bit is set. To prevent an overflow condition, Slave software must then read the SPDAT before another Byte enters the shift register (3). A Slave SPI must complete the write to the SPDAT (shift register) at least one bus cycle before the Master SPI starts a transmission. If the write to the data register is late, the SPI transmits the data already in the shift register from the previous transmission.

**Transmission Formats**

Software can select any of four combinations of serial clock (SCK) phase and polarity using two bits in the SPCON: the Clock Polarity (CPOL (4)) and the Clock Phase (CPHA4). CPOL defines the default SCK line level in idle state. It has no significant effect on the transmission format. CPHA defines the edges on which the input data are sampled and the edges on which the output data are shifted (Figure 60 and Figure 61). The clock phase and polarity should be identical for the Master SPI device and the communicating Slave device.

1. The SPI Module should be configured as a Master before it is enabled (SPEN set). Also, the Master SPI should be configured before the Slave SPI.
2. The SPI Module should be configured as a Slave before it is enabled (SPEN set).
3. The maximum frequency of the SCK for an SPI configured as a Slave is the bus clock speed.
4. Before writing to the CPOL and CPHA bits, the SPI should be disabled (SPEN ='0').

**Figure 60.** Data Transmission Format (CPHA = 0)

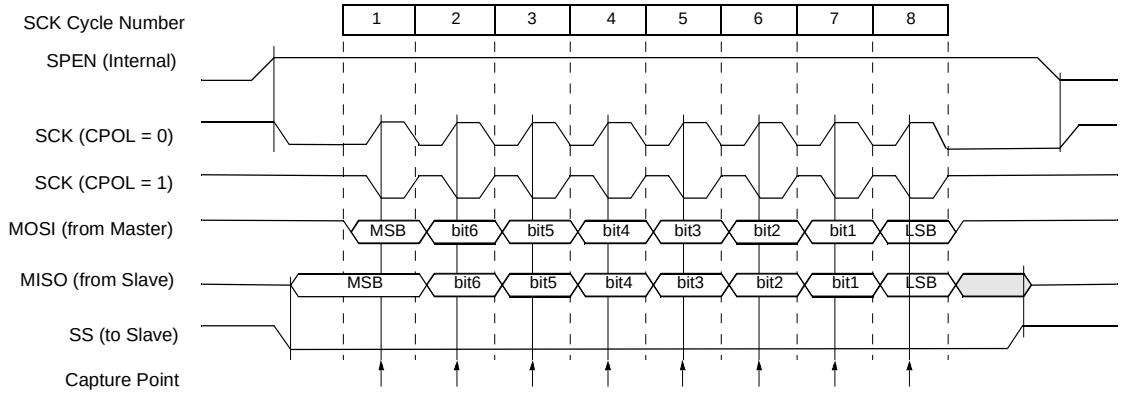



**Figure 61.** Data Transmission Format (CPHA = 1)

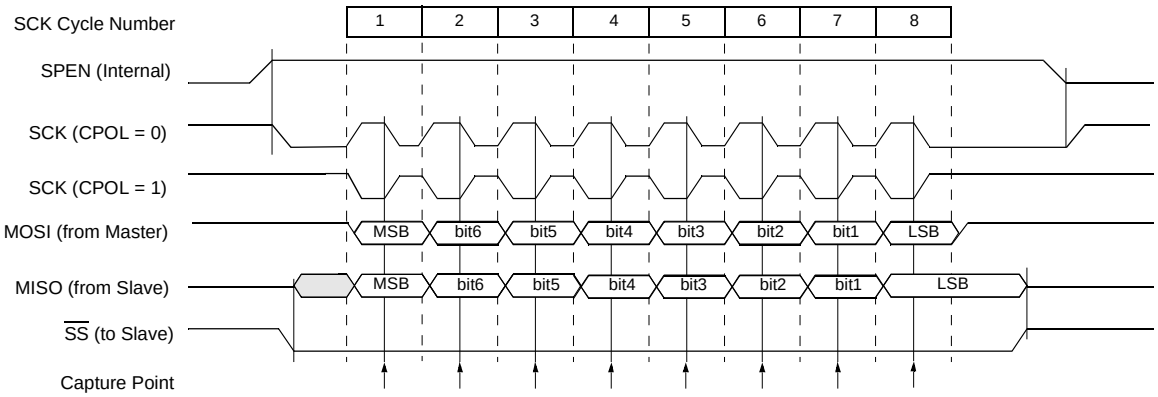
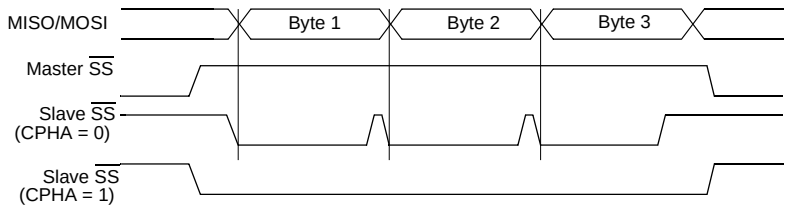



**Figure 62.** CPHA/SS Timing




As shown in Figure 60, the first SCK edge is the MSB capture strobe. Therefore, the Slave must begin driving its data before the first SCK edge, and a falling edge on the SS pin is used to start the transmission. The SS pin must be toggled high and then low between each Byte transmitted (Figure 62).

Figure 61 shows an SPI transmission in which CPHA is '1'. In this case, the Master begins driving its MOSI pin on the first SCK edge. Therefore, the Slave uses the first SCK edge as a start transmission signal. The SS pin can remain low between transmissions (Figure 62). This format may be preferred in systems having only one Master and only one Slave driving the MISO data line.

*Queuing transmission*

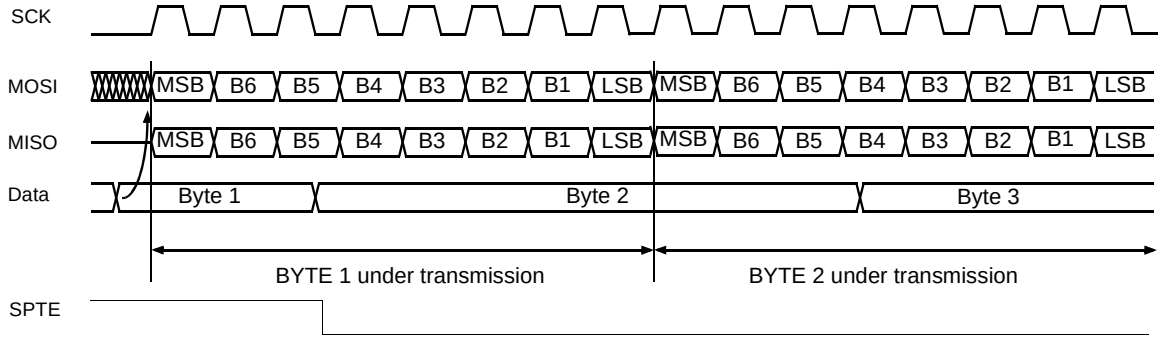
For an SPI configured in master or slave mode, a queued data byte must be transmitted/received immediately after the previous transmission has completed.

When a transmission is in progress a new data can be queued and sent as soon as transmission has been completed. So it is possible to transmit bytes without latency, useful in some applications.

The SPTE bit in SPSCR is set as long as the transmission buffer is free. It means that the user application can write SPDAT with the data to be transmitted until the SPTE becomes cleared.

Figure 63 shows a queuing transmission in master mode. Once the Byte 1 is ready, it is immediately sent on the bus. Meanwhile an other byte is prepared (and the SPTE is cleared), it will be sent at the end of the current transmission. The next data must be ready before the end of the current transmission.

**Figure 63.** Queuing Transmission In Master Mode




In slave mode it is almost the same except it is the external master that start the transmission.

Also, in slave mode, if no new data is ready, the last value received will be the next data byte transmitted.

**Error Conditions**

The following flags in the SPSCR register indicate the SPI error conditions:

*Mode Fault Error (MODF)*

Mode Fault error in Master mode SPI indicates that the level on the Slave Select ($\overline{SS}$) pin is inconsistent with the actual mode of the device.

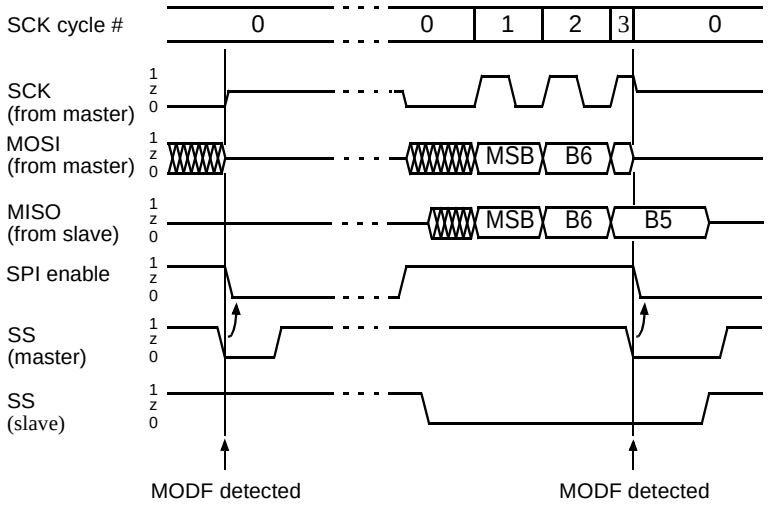
- Mode fault detection in Master mode:

MODF is set to warn that there may be a multi-master conflict for system control. In this case, the SPI system is affected in the following ways:

  – An SPI receiver/error CPU interrupt request is generated
  – The SPEN bit in SPCON is cleared. This disables the SPI
  – The MSTR bit in SPCON is cleared

Clearing the MODF bit is accomplished by a read of SPSCR register with MODF bit set, followed by a write to the SPCON register. SPEN Control bit may be restored to its original set state after the MODF bit has been cleared.

**Figure 64.** Mode Fault Conditions in Master Mode (Cpha ='1'/Cpol ='0')



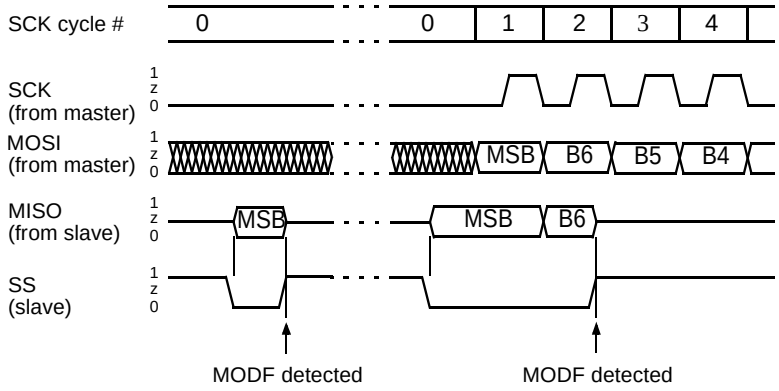

Note: When SS is discarded (SS disabled) it is not possible to detect a MODF error in master mode because the SPI is internally unselected and the SS pin is a general purpose I/O.

- Mode fault detection in Slave mode

In slave mode, the MODF error is detected when SS goes high during a transmission.
A transmission begins when SS goes low and ends once the incoming SCK goes back to its idle level following the shift of the eighteen data bit.
A MODF error occurs if a slave is selected (SS is low) and later unselected (SS is high) even if no SCK is sent to that slave.

At any time, a '1' on the SS pin of a slave SPI puts the MISO pin in a high impedance state and internal state counter is cleared. Also, the slave SPI ignores all incoming SCK clocks, even if it was already in the middle of a transmission. A new transmission will be performed as soon as SS pin returns low.

**Figure 65.** Mode Fault Conditions in Slave Mode




Note: when SS is discarded (SS disabled) it is not possible to detect a MODF error in slave mode because the SPI is internally selected. Also the SS pin becomes a general purpose I/O.

*OverRun Condition*

This error mean that the speed is not adapted for the running application:

An OverRun condition occurs when a byte has been received whereas the previous one has not been read by the application yet.
The last byte (which generate the overrun error) does not overwrite the unread data so that it can still be read. Therefore, an overrun error always indicates the loss of data.

**Interrupts**

Three SPI status flags can generate a CPU interrupt requests:

**Table 91.** SPI Interrupts

| Flag | Request |
|---|---|
| SPIF (SPI data transfer) | SPI Transmitter Interrupt Request |
| MODF (Mode Fault) | SPI mode-fault Interrupt Request |
| SPTE (Transmit register empty) | SPI transmit register empty Interrupt Request |

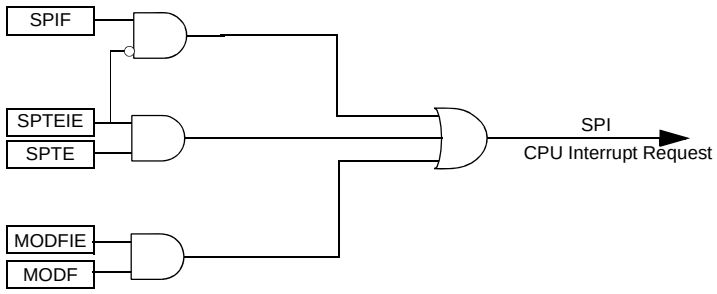
Serial Peripheral data transfer flag, SPIF: This bit is set by hardware when a transfer has been completed. SPIF bit generates transmitter CPU interrupt request only when SPTEIE is disabled.

Mode Fault flag, MODF: This bit is set to indicate that the level on the $\overline{SS}$ is inconsistent with the mode of the SPI (in both master and slave modes).

Serial Peripheral Transmit Register empty flag, SPTE: This bit is set when the transmit buffer is empty (other data can be loaded is SPDAT). SPTE bit generates transmitter CPU interrupt request only when SPTEIE is enabled.

Note: While using SPTE interruption for “burst mode” transfers (SPTEIE=’1’), the user software application should take care to clear SPTEIE, during the last but one data reception (to be able to generate an interrupt on SPIF flag at the end of the last data reception).

**Figure 66. SPI Interrupt Requests Generation**




## Registers

Three registers in the SPI module provide control, status and data storage functions. These registers are describe in the following paragraphs.

### *Serial Peripheral Control Register (SPCON)*

- The Serial Peripheral Control Register does the following:
- Selects one of the Master clock rates
- Configure the SPI Module as Master or Slave
- Selects serial clock polarity and phase
- Enables the SPI Module
- Frees the SS pin for a general-purpose

Table 92 describes this register and explains the use of each bit

**Table 92.** SPCON Register

SPCON - Serial Peripheral Control Register (0D4H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **SPR2** | **SPEN** | **SSDIS** | **MSTR** | **CPOL** | **CPHA** | **SPR1** | **SPR0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | SPR2 | **Serial Peripheral Rate 2**<br>Bit with SPR1 and SPR0 define the clock rate (See bits SPR1 and SPR0 for detail). |
| 6 | SPEN | **Serial Peripheral Enable**<br>Cleared to disable the SPI interface (internal reset of the SPI).<br>Set to enable the SPI interface. |
| 5 | SSDIS | **$\overline{SS}$ Disable**<br>Cleared to enable $\overline{SS}$ in both Master and Slave modes.<br>Set to disable $\overline{SS}$ in both Master and Slave modes. In Slave mode, this bit has no effect if CPHA ='0'. When SSDIS is set, no MODF interrupt request is generated. |
| 4 | MSTR | **Serial Peripheral Master**<br>Cleared to configure the SPI as a Slave.<br>Set to configure the SPI as a Master. |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 3 | CPOL | **Clock Polarity**<br>Cleared to have the SCK set to '0' in idle state.<br>Set to have the SCK set to '1' in idle state. |
| 2 | CPHA | **Clock Phase**<br>Cleared to have the data sampled when the SCK leaves the idle state (see CPOL).<br>Set to have the data sampled when the SCK returns to idle state (see CPOL). |
| 1 | SPR1 | **SPR2 SPR1 SPR0 Serial Peripheral Rate**<br>0 0 0 Invalid<br>0 0 1 $F_{CLK\ PERIPH}$/4<br>0 1 0 $F_{CLK\ PERIPH}$/8<br>0 1 1 $F_{CLK\ PERIPH}$/16<br>1 0 0 $F_{CLK\ PERIPH}$/32<br>1 0 1 $F_{CLK\ PERIPH}$/64<br>1 1 0 $F_{CLK\ PERIPH}$/128<br>1 1 1 Invalid |
| 0 | SPR0 | |

Reset Value = 0001 0100b

Not bit addressable

*Serial Peripheral Status Register and Control (SPSCR)*

The Serial Peripheral Status Register contains flags to signal the following conditions:

- Data transfer complete
- Write collision
- Inconsistent logic level on $\overline{SS}$ pin (mode fault error)

**Table 93.** SPSCR Register

SPSCR - Serial Peripheral Status and Control register (0D5H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **SPIF** | **-** | **OVR** | **MODF** | **SPTE** | **UARTM** | **SPTEIE** | **MODFIE** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | SPIF | **Serial Peripheral Data Transfer Flag**<br>Cleared by hardware to indicate data transfer is in progress or has been approved by a clearing sequence.<br>Set by hardware to indicate that the data transfer has been completed.<br>This bit is cleared when reading or writing SPDATA after reading SPSCR. |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | OVR | **Overrun Error Flag**<br>- Set by hardware when a byte is received whereas SPIF is set (the previous received data is not overwritten).<br>- Cleared by hardware when reading SPSCR |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 4 | MODF | **Mode Fault**<br>- Set by hardware to indicate that the $\overline{SS}$ pin is in inappropriate logic level (in both master and slave modes).<br>- Cleared by hardware when reading SPSCR<br>When MODF error occurred:<br>- In slave mode: SPI interface ignores all transmitted data while $\overline{SS}$ remains high. A new transmission is perform as soon as SS returns low.<br>- In master mode: SPI interface is disabled (SPEN=0, see description for SPEN bit in SPCON register). |
| 3 | SPTE | **Serial Peripheral Transmit register Empty**<br>- Set by hardware when transmit register is empty (if needed, SPDAT can be loaded with another data).<br>- Cleared by hardware when transmit register is full (no more data should be loaded in SPDAT). |
| 2 | UARTM | **Serial Peripheral UART mode**<br>Set and cleared by software:<br>- Clear: Normal mode, data are transmitted MSB first (default)<br>- Set: UART mode, data are transmitted LSB first. |
| 1 | SPTEIE | **Interrupt Enable for SPTE**<br>Set and cleared by software:<br>- Set to enable SPTE interrupt generation (when SPTE goes high, an interrupt is generated).<br>- Clear to disable SPTE interrupt generation<br>Caution: When SPTEIE is set no interrupt generation occurred when SPIF flag goes high. To enable SPIF interrupt again, SPTEIE should be cleared. |
| 0 | MODFIE | **Interrupt Enable for MODF**<br>Set and cleared by software:<br>- Set to enable MODF interrupt generation<br>- Clear to disable MODF interrupt generation |

Reset Value = 00X0 XXXXb

Not Bit addressable

*Serial Peripheral DATa Register (SPDAT)*

The Serial Peripheral Data Register (Table 94) is a read/write buffer for the receive data register. A write to SPDAT places data directly into the shift register. No transmit buffer is available in this model.

A Read of the SPDAT returns the value located in the receive buffer and not the content of the shift register.

**Table 94.** SPDAT Register

SPDAT - Serial Peripheral Data Register (0D6H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **R7** | **R6** | **R5** | **R4** | **R3** | **R2** | **R1** | **R0** |

Reset Value = Indeterminate

R7:R0: Receive data bits

SPCON, SPSTA and SPDAT registers may be read and written at any time while there is no on-going exchange. However, special care should be taken when writing to them while a transmission is on-going:

- Do not change SPR2, SPR1 and SPR0
- Do not change CPHA and CPOL
- Do not change MSTR
- Clearing SPEN would immediately disable the peripheral
- Writing to the SPDAT will cause an overflow.

## Programmable Counter Array (PCA)

The PCA provides more timing capabilities with less CPU intervention than the standard timer/counters. Its advantages include reduced software overhead and improved accuracy. The PCA consists of a dedicated timer/counter which serves as the time base for an array of five compare/capture modules. Its clock input can be programmed to count any of the following signals:

- PCA clock frequency/6 (see “clock” section)
- PCA clock frequency/2
- Timer 0 overflow
- External input on ECI (P1.2)

Each compare/capture modules can be programmed in any one of the following modes:

- rising and/or falling edge capture,
- software timer,
- high-speed output,
- pulse width modulator.

Module 4 can also be programmed as a WatchDog timer. see the "PCA WatchDog Timer" section.

When the compare/capture modules are programmed in capture mode, software timer, or high speed output mode, an interrupt can be generated when the module executes its function. All five modules plus the PCA timer overflow share one interrupt vector.

The PCA timer/counter and compare/capture modules share Port 1 for external I/Os. These pins are listed below. If the port is not used for the PCA, it can still be used for standard I/O.

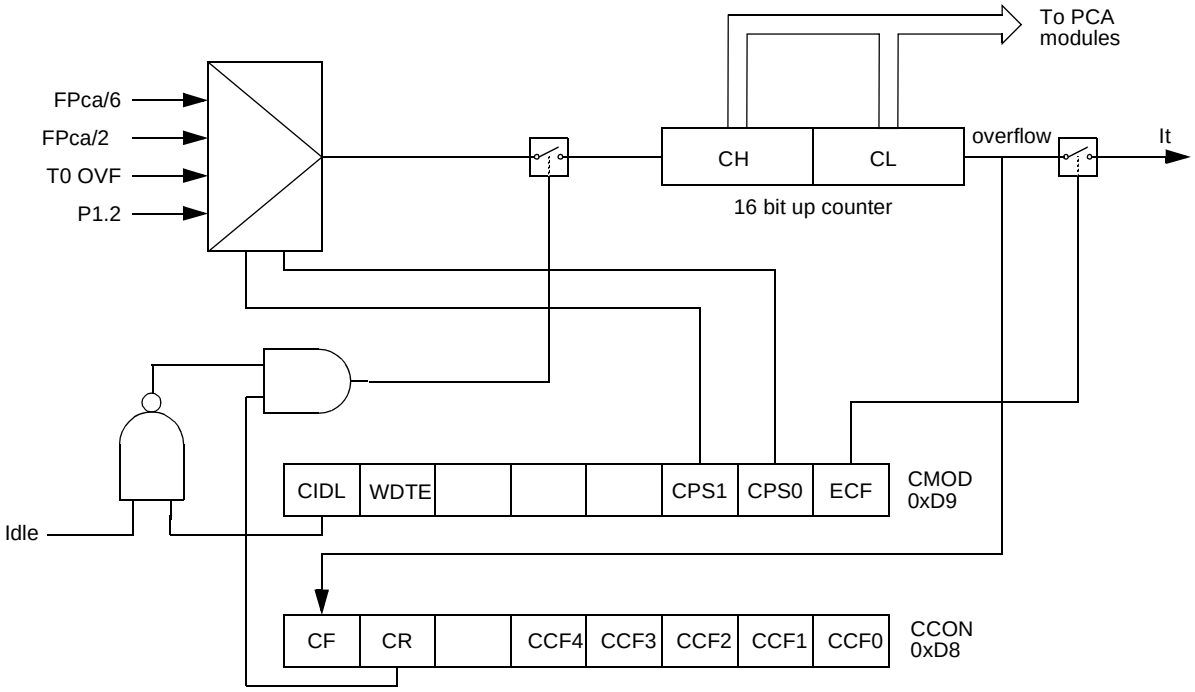
| PCA Component | External I/O Pin |
|---|---|
| 16-bit Counter | P1.2/ECI |
| 16-bit Module 0 | P1.3/CEX0 |
| 16-bit Module 1 | P1.4/CEX1 |
| 16-bit Module 2 | P1.5/CEX2 |
| 16-bit Module 3 | P1.6/CEX3 |
| 16-bit Module 4 | P1.7/CEX4 |

### PCA Timer

The PCA timer is a common time base for all five modules (see Figure 67). The timer count source is determined from the CPS1 and CPS0 bits in the CMOD SFR (see Table 8) and can be programmed to run at:

- 1/6 the PCA clock frequency.
- 1/2 the PCA clock frequency.
- the Timer 0 overflow.
- the input on the ECI pin (P1.2).

**Figure 67.** PCA Timer/Counter




The CMOD register includes three additional bits associated with the PCA.

- The CIDL bit which allows the PCA to stop during idle mode.
- The WDTE bit which enables or disables the WatchDog function on module 4.
- The ECF bit which when set causes an interrupt and the PCA overflow flag CF in CCON register to be set when the PCA timer overflows.

The CCON register contains the run control bit for the PCA and the flags for the PCA timer and each module.

- The CR bit must be set to run the PCA. The PCA is shut off by clearing this bit.
- The CF bit is set when the PCA counter overflows and an interrupt will be generated if the ECF bit in CMOD register is set. The CF bit can only be cleared by software.
- The CCF0:4 bits are the flags for the modules (CCF0 for module0...) and are set by hardware when either a match or a capture occurs. These flags also can be cleared by software.

## PCA Modules

Each one of the five compare/capture modules has six possible functions. It can perform:

- 16-bit Capture, positive-edge triggered
- 16-bit Capture, negative-edge triggered
- 16-bit Capture, both positive and negative-edge triggered
- 16-bit Software Timer
- 16-bit High Speed Output
- 8-bit Pulse Width Modulator.

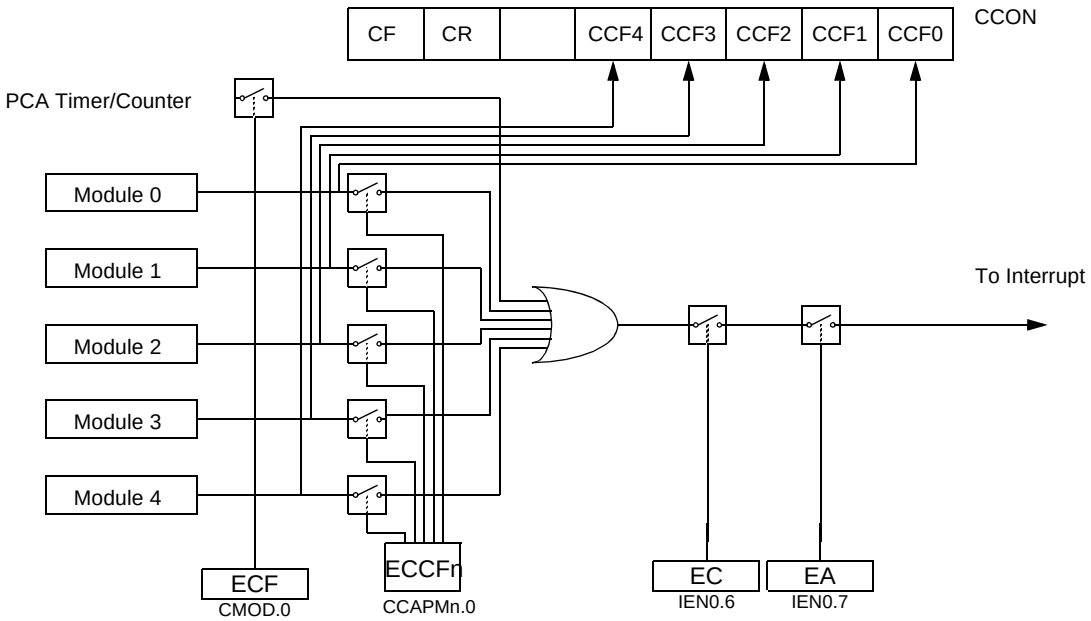
In addition module 4 can be used as a WatchDog Timer.

Each module in the PCA has a special function register associated with it (CCAPM0 for module 0 ...). The CCAPM0:4 registers contain the bits that control the mode that each module will operate in.

- The ECCF bit enables the CCF flag in the CCON register to generate an interrupt when a match or compare occurs in the associated module.
- The PWM bit enables the pulse width modulation mode.
- The TOG bit when set causes the CEX output associated with the module to toggle when there is a match between the PCA counter and the module's capture/compare register.
- The match bit MAT when set will cause the CCFn bit in the CCON register to be set when there is a match between the PCA counter and the module's capture/compare register.
- The two bits CAPN and CAPP in CCAPMn register determine the edge that a capture input will be active on. The CAPN bit enables the negative edge, and the CAPP bit enables the positive edge. If both bits are set both edges will be enabled.
- The bit ECOM in CCAPM register when set enables the comparator function.

## PCA Interrupt

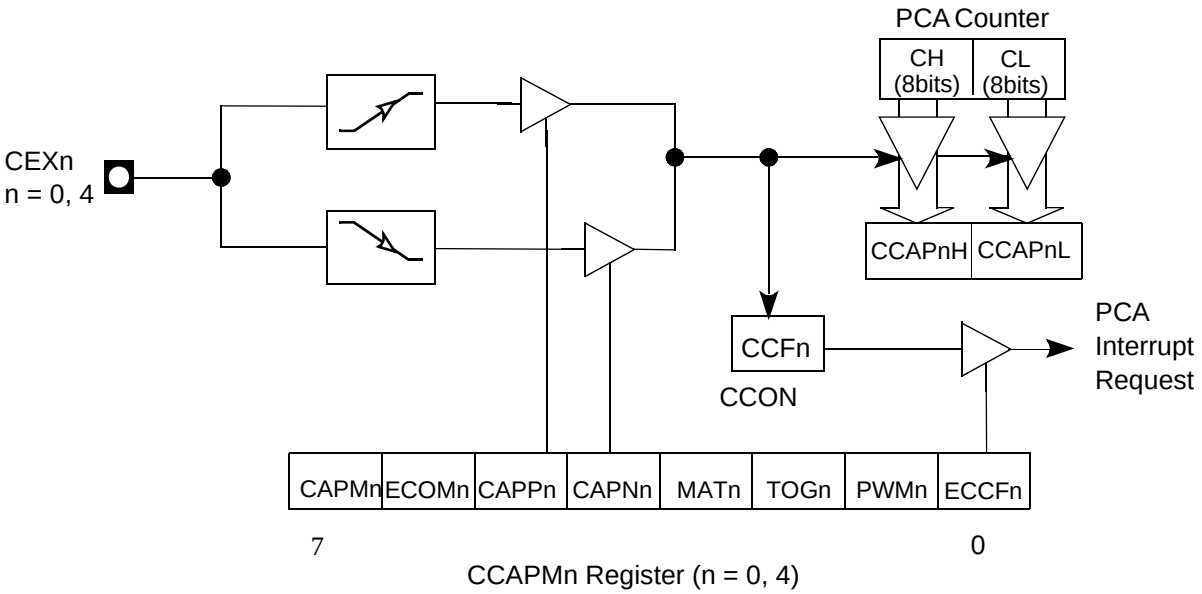
**Figure 68.** PCA Interrupt System




## PCA Capture Mode

To use one of the PCA modules in capture mode either one or both of the CCAPM bits CAPN and CAPP for that module must be set. The external CEX input for the module (on port 1) is sampled for a transition. When a valid transition occurs the PCA hardware loads the value of the PCA counter registers (CH and CL) into the module's capture registers (CCAPnL and CCAPnH). If the CCFn bit for the module in the CCON SFR and the ECCFn bit in the CCAPMn SFR are set then an interrupt will be generated.

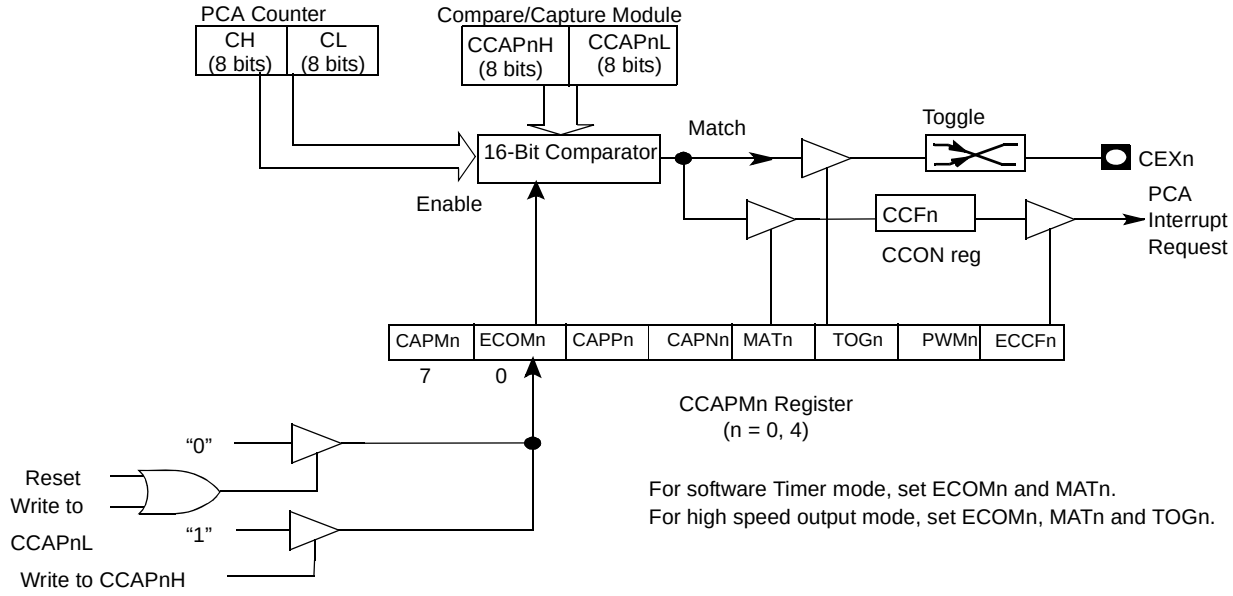
**Figure 69.** PCA Capture Mode




## 16-bit Software Timer Mode

The PCA modules can be used as software timers by setting both the ECOM and MAT bits in the modules CCAPMn register. The PCA timer will be compared to the module's capture registers and when a match occurs an interrupt will occur if the CCFn (CCON SFR) and the ECCFn (CCAPMn SFR) bits for the module are both set.

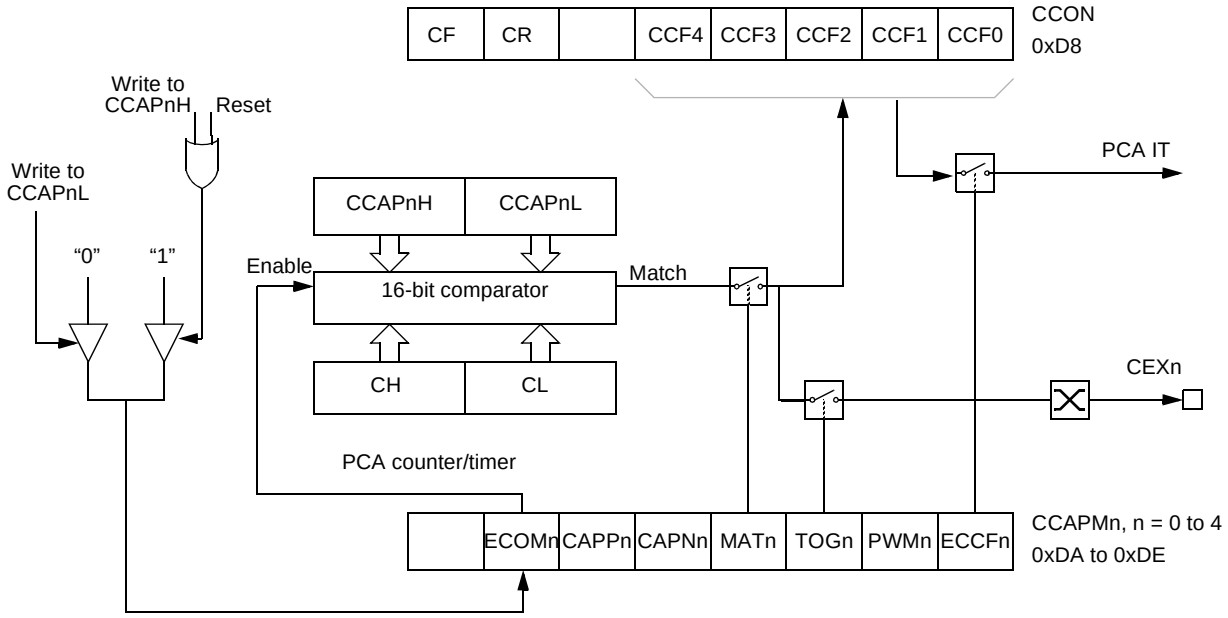
**Figure 70.** PCA 16-bit Software Timer and High Speed Output Mode



## High Speed Output Mode

In this mode the CEX output (on port 1) associated with the PCA module will toggle each time a match occurs between the PCA counter and the module's capture registers. To activate this mode the TOG, MAT, and ECOM bits in the module's CCAPMn SFR must be set.

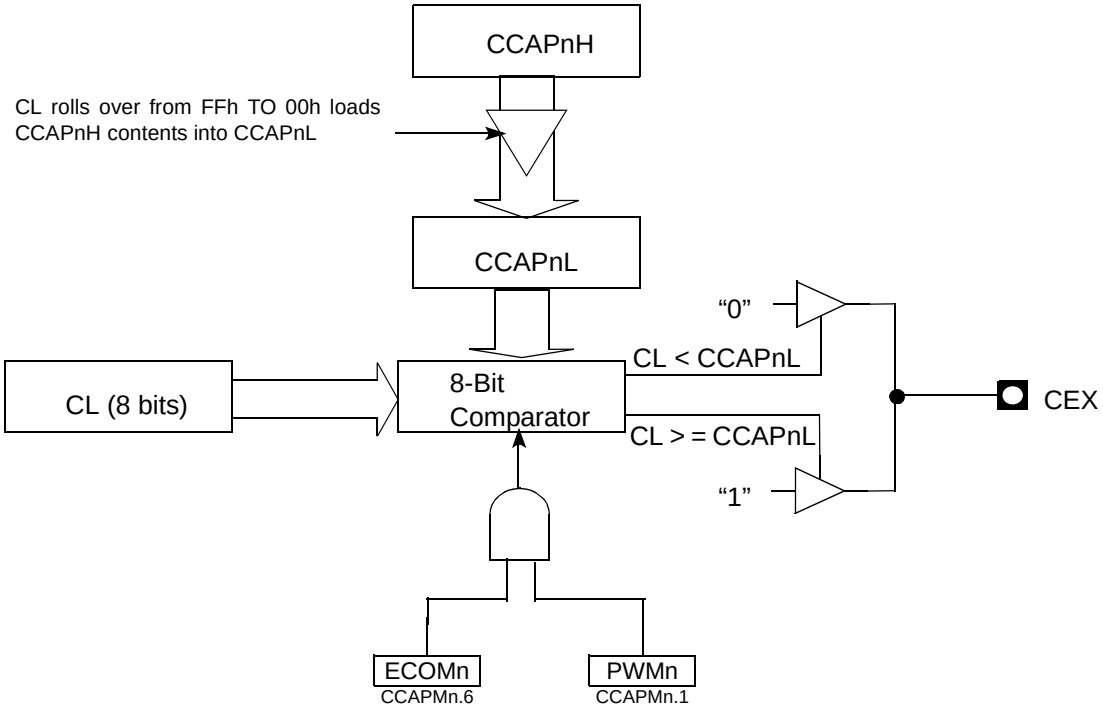
**Figure 71.** PCA High Speed Output Mode




## Pulse Width Modulator Mode

All the PCA modules can be used as PWM outputs. The output frequency depends on the source for the PCA timer. All the modules will have the same output frequency because they all share the PCA timer. The duty cycle of each module is independently variable using the module's capture register CCAPLn. When the value of the PCA CL SFR is less than the value in the module's CCAPLn SFR the output will be low, when it is equal to or greater than it, the output will be high. When CL overflows from FF to 00, CCAPLn is reloaded with the value in CCAPHn. the allows the PWM to be updated without glitches. The PWM and ECOM bits in the module's CCAPMn register must be set to enable the PWM mode.

**Figure 72.** PCA PWM Mode




## PCA WatchDog Timer

An on-board WatchDog timer is available with the PCA to improve system reliability without increasing chip count. WatchDog timers are useful for systems that are sensitive to noise, power glitches, or electrostatic discharge. Module 4 is the only PCA module that can be programmed as a WatchDog. However, this module can still be used for other modes if the WatchDog is not needed. The user pre-loads a 16-bit value in the compare registers. Just like the other compare modes, this 16-bit value is compared to the PCA timer value. If a match is allowed to occur, an internal reset will be generated. This will not cause the RST pin to be driven high.

To hold off the reset, the user has three options:

- periodically change the compare value so it will never match the PCA timer,
- periodically change the PCA timer value so it will never match the compare values, or
- disable the WatchDog by clearing the WDTE bit before a match occurs and then re-enable it.

The first two options are more reliable because the WatchDog timer is never disabled as in the third option. If the program counter ever goes astray, a match will eventually occur and cause an internal reset. If other PCA modules are being used the second option not recommended either. Remember, the PCA timer is the time base for all modules; changing the time base for other modules would not be a good idea. Thus, in most applications the first solution is the best option.

## PCA Registers

**Table 95.** CMOD Register

CMOD (S:D9h)
PCA Counter Mode Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CIDL | WDTE | - | - | - | CPS1 | CPS0 | ECF |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | CIDL | **PCA Counter Idle Control bit**<br>Clear to let the PCA run during Idle mode.<br>Set to stop the PCA when Idle mode is invoked. |
| 6 | WDTE | **WatchDog Timer Enable**<br>Clear to disable WatchDog Timer function on PCA Module 4,<br>Set to enable it. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 2 | CPS1 | **EWC Count Pulse Select bits**<br>CPS1 CPS0 Clock source<br>0 0 Internal Clock, FPca/6<br>0 1 Internal Clock, FPca/2<br>1 0 Timer 0 overflow<br>1 1 External clock at ECI/P1.2 pin (Max. Rate = FPca/4) |
| 1 | CPS0 | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 0 | ECF | **Enable PCA Counter Overflow Interrupt bit**<br>Clear to disable CF bit in CCON register to generate an interrupt.<br>Set to enable CF bit in CCON register to generate an interrupt. |

Reset Value = 00XX X000b

**Table 96.** CCON Register

CCON (S:D8h)
PCA Counter Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CF | CR | - | CCF4 | CCF3 | CCF2 | CCF1 | CCF0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | CF | **PCA Timer/Counter Overflow flag**<br>Set by hardware when the PCA Timer/Counter rolls over. This generates a PCA interrupt request if the ECF bit in CMOD register is set.<br>Must be cleared by software. |
| 6 | CR | **PCA Timer/Counter Run Control bit**<br>Clear to turn the PCA Timer/Counter off.<br>Set to turn the PCA Timer/Counter on. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | CCF4 | **PCA Module 4 Compare/Capture flag**<br>Set by hardware when a match or capture occurs. This generates a PCA interrupt request if the ECCF 4 bit in CCAPM 4 register is set.<br>Must be cleared by software. |
| 3 | CCF3 | **PCA Module 3 Compare/Capture flag**<br>Set by hardware when a match or capture occurs. This generates a PCA interrupt request if the ECCF 3 bit in CCAPM 3 register is set.<br>Must be cleared by software. |
| 2 | CCF2 | **PCA Module 2 Compare/Capture flag**<br>Set by hardware when a match or capture occurs. This generates a PCA interrupt request if the ECCF 2 bit in CCAPM 2 register is set.<br>Must be cleared by software. |
| 1 | CCF1 | **PCA Module 1 Compare/Capture flag**<br>Set by hardware when a match or capture occurs. This generates a PCA interrupt request if the ECCF 1 bit in CCAPM 1 register is set.<br>Must be cleared by software. |
| 0 | CCF0 | **PCA Module 0 Compare/Capture flag**<br>Set by hardware when a match or capture occurs. This generates a PCA interrupt request if the ECCF 0 bit in CCAPM 0 register is set.<br>Must be cleared by software. |

Reset Value = 00X0 0000b

**Table 97.** CCAPnH Registers

CCAP0H (S:FAh)
CCAP1H (S:FBh)
CCAP2H (S:FCh)
CCAP3H (S:FDh)
CCAP4H (S:FEh)
PCA High Byte Compare/Capture Module n Register (n=0..4)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CCAPnH 7 | CCAPnH 6 | CCAPnH 5 | CCAPnH 4 | CCAPnH 3 | CCAPnH 2 | CCAPnH 1 | CCAPnH 0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:0 | CCAPnH 7:0 | High byte of EWC-PCA comparison or capture values |

Reset Value = 0000 0000b

**Table 98.** CCAPnL Registers

CCAP0L (S:EAh)
CCAP1L (S:EBh)
CCAP2L (S:ECh)
CCAP3L (S:EDh)
CCAP4L (S:EEh)
PCA Low Byte Compare/Capture Module n Register (n=0..4)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CCAPnL 7 | CCAPnL 6 | CCAPnL 5 | CCAPnL 4 | CCAPnL 3 | CCAPnL 2 | CCAPnL 1 | CCAPnL 0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:0 | CCAPnL 7:0 | Low byte of EWC-PCA comparison or capture values |

Reset Value = 0000 0000b

**Table 99.** CCAPMn Registers

CCAPM0 (S:DAh)
CCAPM1 (S:DBh)
CCAPM2 (S:DCh)
CCAPM3 (S:DDh)
CCAPM4 (S:DEh)
PCA Compare/Capture Module n Mode registers (n=0..4)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | ECOMn | CAPPn | CAPNn | MATn | TOGn | PWMn | ECCFn |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The Value read from this bit is indeterminate. Do not set this bit. |
| 6 | ECOMn | **Enable Compare Mode Module x bit**<br>Clear to disable the Compare function.<br>Set to enable the Compare function.<br>The Compare function is used to implement the software Timer, the high-speed output, the Pulse Width Modulator (PWM) and the WatchDog Timer (WDT). |
| 5 | CAPPn | **Capture Mode (Positive) Module x bit**<br>Clear to disable the Capture function triggered by a positive edge on CEXx pin.<br>Set to enable the Capture function triggered by a positive edge on CEXx pin |
| 4 | CAPNn | **Capture Mode (Negative) Module x bit**<br>Clear to disable the Capture function triggered by a negative edge on CEXx pin.<br>Set to enable the Capture function triggered by a negative edge on CEXx pin. |
| 3 | MATn | **Match Module x bit**<br>Set when a match of the PCA Counter with the Compare/Capture register sets CCFx bit in CCON register, flagging an interrupt. |
| 2 | TOGn | **Toggle Module x bit**<br>The toggle mode is configured by setting ECOMx, MATx and TOGx bits.<br>Set when a match of the PCA Counter with the Compare/Capture register toggles the CEXx pin. |
| 1 | PWMn | **Pulse Width Modulation Module x Mode bit**<br>Set to configure the module x as an 8-bit Pulse Width Modulator with output waveform on CEXx pin. |
| 0 | ECCFn | **Enable CCFx Interrupt bit**<br>Clear to disable CCFx bit in CCON register to generate an interrupt request.<br>Set to enable CCFx bit in CCON register to generate an interrupt request. |

Reset Value = X000 0000b

**Table 100.** CH Register

CH (S:F9h)
PCA Counter Register High Value

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CH 7 | CH 6 | CH 5 | CH 4 | CH 3 | CH 2 | CH 1 | CH 0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:0 | CH 7:0 | High byte of Timer/Counter |

Reset Value = 0000 00000b

**Table 101.** CL Register

CL (S:E9h)
PCA counter Register Low Value

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CL 7 | CL 6 | CL 5 | CL 4 | CL 3 | CL 2 | CL 1 | CL 0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:0 | CL0 7:0 | Low byte of Timer/Counter |

Reset Value = 0000 00000b

## Analog-to-Digital Converter (ADC)

This section describes the on-chip 10 bit analog-to-digital converter of the AT89C51CC03. Eight ADC channels are available for sampling of the external sources AN0 to AN7. An analog multiplexer allows the single ADC converter to select one from the 8 ADC channels as ADC input voltage (ADCIN). ADCIN is converted by the 10-bit cascaded potentiometric ADC.

Two kinds of conversion are available:
- Standard conversion (8 bits).
- Precision conversion (10 bits) (Up to 85°C only).

For the precision conversion, set bit PSIDLE in ADCON register and start conversion. The device is in a pseudo-idle mode, the CPU does not run but the peripherals are always running. This mode allows digital noise to be as low as possible, to ensure high precision conversion.

For this mode it is necessary to work with end of conversion interrupt, which is the only way to wake the device up.

If another interrupt occurs during the precision conversion, it will be treated only after this conversion is ended.

### Features

- 8 channels with multiplexed inputs
- 10-bit cascaded potentiometric ADC
- Conversion time 16 micro-seconds (typ.)
- Zero Error (offset) ± 2 LSB max
- Positive External Reference Voltage Range (VREF) 2.4 to 3.0Volt (typ.)
- ADCIN Range 0 to 3Volt
- Integral non-linearity typical 1 LSB, max. 2 LSB
- Differential non-linearity typical 0.5 LSB, max. 1 LSB
- Conversion Complete Flag or Conversion Complete Interrupt
- Selectable ADC Clock

### ADC Port1 I/O Functions

Port 1 pins are general I/O that are shared with the ADC channels. The channel select bit in ADCF register define which ADC channel/port1 pin will be used as ADCIN. The remaining ADC channels/port1 pins can be used as general-purpose I/O or as the alternate function that is available.

**Figure 73.** ADC Description

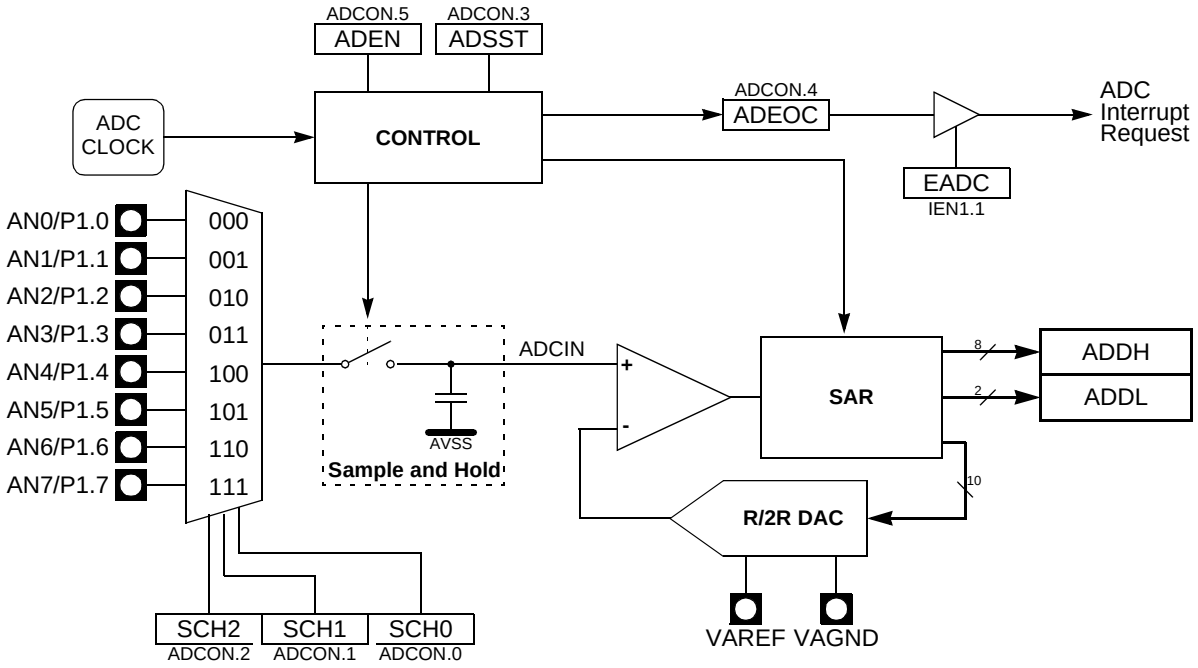
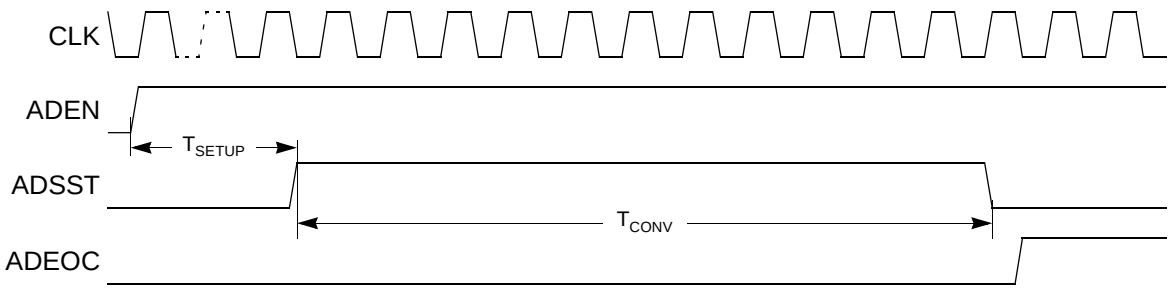



Figure 74 shows the timing diagram of a complete conversion. For simplicity, the figure depicts the waveforms in idealized form and do not provide precise timing information. For ADC characteristics and timing parameters refer to the Section “AC Characteristics” of the AT89C51CC03 datasheet.

**Figure 74.** Timing Diagram




Note: Tsetup min = 4 us
Tconv=11 clock ADC = 1sample and hold + 10 bit conversion
The user must ensure that 4 us minimum time between setting ADEN and the start of the first conversion.

## ADC Converter Operation

A start of single A/D conversion is triggered by setting bit ADSST (ADCON.3).

After completion of the A/D conversion, the ADSST bit is cleared by hardware.

The end-of-conversion flag ADEOC (ADCON.4) is set when the value of conversion is available in ADDH and ADDL, it must be cleared by software. If the bit EADC (IEN1.1) is set, an interrupt occur when flag ADEOC is set (see Figure 76). Clear this flag for re-arming the interrupt.

The bits SCH0 to SCH2 in ADCON register are used for the analog input channel selection.

**Table 102.** Selected Analog input

| SCH2 | SCH1 | SCH0 | Selected Analog input |
|---|---|---|---|
| 0 | 0 | 0 | AN0 |
| 0 | 0 | 1 | AN1 |
| 0 | 1 | 0 | AN2 |
| 0 | 1 | 1 | AN3 |
| 1 | 0 | 0 | AN4 |
| 1 | 0 | 1 | AN5 |
| 1 | 1 | 0 | AN6 |
| 1 | 1 | 1 | AN7 |

## Voltage Conversion

When the ADCIN is equals to VAREF the ADC converts the signal to 3FFh (full scale). If the input voltage equals VAGND, the ADC converts it to 000h. Input voltage between VAREF and VAGND are a straight-line linear conversion. All other voltages will result in 3FFh if greater than VAREF and 000h if less than VAGND.

Note that ADCIN should not exceed VAREF absolute maximum range! (See section “AC-DC”)

## Clock Selection

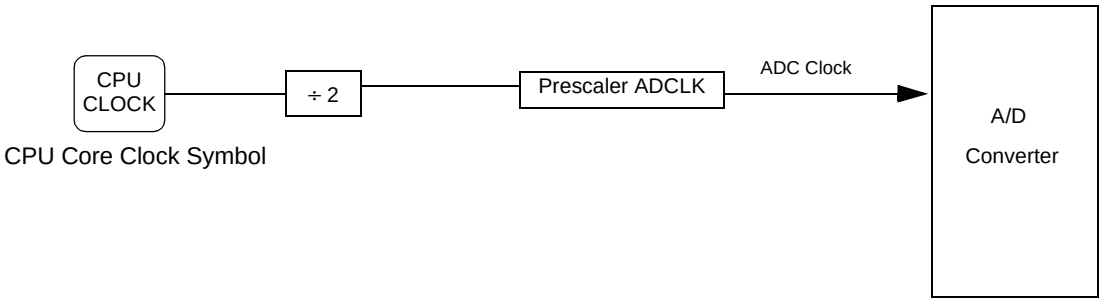
The ADC clock is the same as CPU.

The maximum clock frequency is defined in the DC parameters for A/D converter. A prescaler is featured (ADCCLH) to generate the ADC clock from the oscillator frequency.

if PRS = 0 then $F_{ADC} = F_{periph} / 64$

if PRS > 0 then $F_{ADC} = F_{periph} / 2 \times PRS$

**Figure 75.** A/D Converter Clock



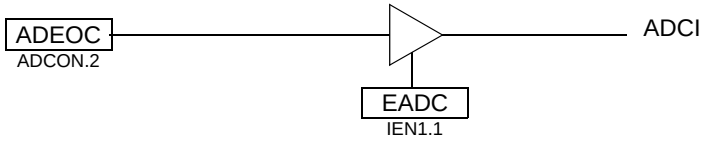

## ADC Standby Mode

When the ADC is not used, it is possible to set it in standby mode by clearing bit ADEN in ADCON register. In this mode its power dissipation is about 1 µW.

## IT ADC Management

An interrupt end-of-conversion will occurs when the bit ADEOC is activated and the bit EADC is set. For re-arming the interrupt the bit ADEOC must be cleared by software.

**Figure 76.** ADC Interrupt Structure




## Routines examples

1. Configure P1.2 and P1.3 in ADC channels

```
// configure channel P1.2 and P1.3 for ADC
 ADCF = 0Ch

// Enable the ADC
 ADCON = 20h
```

2. Start a standard conversion

```
// The variable "channel" contains the channel to convert
// The variable "value_converted" is an unsigned int
// Clear the field SCH[2:0]
 ADCON and = F8h
// Select channel
 ADCON | = channel
// Start conversion in standard mode
 ADCON | = 08h
// Wait flag End of conversion
 while((ADCON and 01h)! = 01h)
// Clear the End of conversion flag
 ADCON and = EFh
// read the value
 value_converted = (ADDH << 2)+(ADDL)
```

3. Start a precision conversion (need interrupt ADC)

```
// The variable "channel" contains the channel to convert
// Enable ADC
```

```
EADC = 1
// clear the field SCH[2:0]
ADCON and = F8h
// Select the channel
ADCON | = channel
// Start conversion in precision mode
ADCON | = 48h
```

Note: to enable the ADC interrupt:
EA = 1

## Registers

**Table 103.** ADCF Register

ADCF (S:F6h)
ADC Configuration

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CH 7 | CH 6 | CH 5 | CH 4 | CH 3 | CH 2 | CH 1 | CH 0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | CH 0:7 | **Channel Configuration**<br>Set to use P1.x as ADC input.<br>Clear to use P1.x as standart I/O port. |

Reset Value =0000 0000b

**Table 104.** ADCON Register

ADCON (S:F3h)
ADC Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | PSIDLE | ADEN | ADEOC | ADSST | SCH2 | SCH1 | SCH0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | |
| 6 | PSIDLE | **Pseudo Idle Mode (Best Precision)**<br>Set to put in idle mode during conversion<br>Clear to convert without idle mode. |
| 5 | ADEN | **Enable/Standby Mode**<br>Set to enable ADC<br>Clear for Standby mode (power dissipation 1 uW). |
| 4 | ADEOC | **End Of Conversion**<br>Set by hardware when ADC result is ready to be read. This flag can generate an interrupt.<br>Must be cleared by software. |
| 3 | ADSST | **Start and Status**<br>Set to start an A/D conversion.<br>Cleared by hardware after completion of the conversion |
| 2-0 | SCH2:0 | **Selection of Channel to Convert**<br>see Table 102 |

Reset Value =X000 0000b

**Table 105.** ADCLK Register

ADCLK (S:F2h)
ADC Clock Prescaler

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | PRS 4 | PRS 3 | PRS 2 | PRS 1 | PRS 0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-5 | - | **Reserved**<br>The value read from these bits are indeterminate. Do not set these bits. |
| 4-0 | PRS4:0 | **Clock Prescaler**<br>See Note (1) |

Reset Value = XXX0 0000b

Note: 1. In X1 mode:

For PRS > 0 $F_{ADC} = \frac{FXTAL}{4xPRS}$

For PRS = 0 $F_{ADC} = \frac{FXTAL}{128}$

In X2 mode:

For PRS > 0 $F_{ADC} = \frac{FXTAL}{2xPRS}$

For PRS = 0 $F_{ADC} = \frac{FXTAL}{64}$

**Table 106.** ADDH Register

ADDH (S:F5h Read Only)
ADC Data High Byte Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ADAT 9 | ADAT 8 | ADAT 7 | ADAT 6 | ADAT 5 | ADAT 4 | ADAT 3 | ADAT 2 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | ADAT9:2 | **ADC result**<br>bits 9-2 |

Reset Value = 00h

**Table 107.** ADDL Register

ADDL (S:F4h Read Only)
ADC Data Low Byte Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | ADAT 1 | ADAT 0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-2 | - | **Reserved**<br>The value read from these bits are indeterminate. Do not set these bits. |
| 1-0 | ADAT1:0 | **ADC result**<br>bits 1-0 |

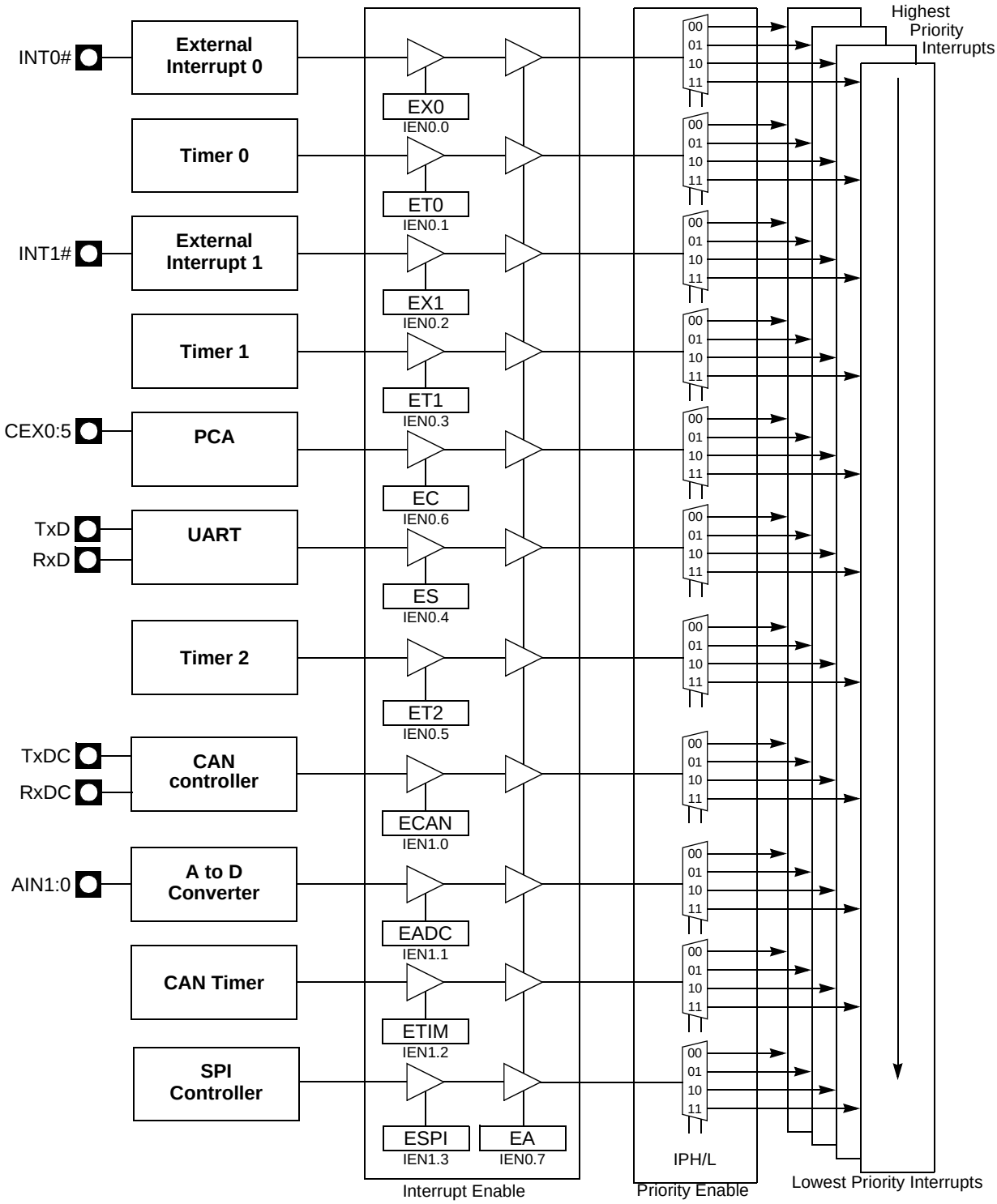
Reset Value = 00h

# Interrupt System

## Introduction

The CAN Controller has a total of 10 interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (timers 0, 1 and 2), a serial port interrupt, a PCA, a CAN interrupt, a timer overrun interrupt and an ADC. These interrupts are shown below.

**Figure 77.** Interrupt Control System



Each of the interrupt sources can be individually enabled or disabled by setting or clearing a bit in the Interrupt Enable register. This register also contains a global disable bit which must be cleared to disable all the interrupts at the same time.

Each interrupt source can also be individually programmed to one of four priority levels by setting or clearing a bit in the Interrupt Priority registers. The Table below shows the bit values and priority levels associated with each combination.

**Table 108.** Priority Level Bit Values

| IPH.x | IPL.x | Interrupt Level Priority |
|---|---|---|
| 0 | 0 | 0 (Lowest) |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 (Highest) |

A low-priority interrupt can be interrupted by a high priority interrupt but not by another low-priority interrupt. A high-priority interrupt cannot be interrupted by any other interrupt source.

If two interrupt requests of different priority levels are received simultaneously, the request of the higher priority level is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence, see Table 109.

**Table 109.** Interrupt priority Within level

| Interrupt Name | Interrupt Address Vector | Priority Number |
|---|---|---|
| external interrupt (INT0) | 0003h | 1 |
| Timer0 (TF0) | 000Bh | 2 |
| external interrupt (INT1) | 0013h | 3 |
| Timer1 (TF1) | 001Bh | 4 |
| PCA (CF or CCFn) | 0033h | 5 |
| UART (RI or TI) | 0023h | 6 |
| Timer2 (TF2) | 002Bh | 7 |
| CAN (Txok, Rxok, Err or OvrBuf) | 003Bh | 8 |
| ADC (ADCI) | 0043h | 9 |
| CAN Timer Overflow (OVRTIM) | 004Bh | 10 |
| SPI interrupt | 0053h | 11 |

## Registers

**Table 110.** IEN0 Register

IEN0 (S:A8h)
Interrupt Enable Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **EA** | **EC** | **ET2** | **ES** | **ET1** | **EX1** | **ET0** | **EX0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | EA | **Enable All Interrupt bit**<br>Clear to disable all interrupts.<br>Set to enable all interrupts.<br>If EA=1, each interrupt source is individually enabled or disabled by setting or clearing its interrupt enable bit. |
| 6 | EC | **PCA Interrupt Enable**<br>Clear to disable the PCA interrupt.<br>Set to enable the PCA interrupt. |
| 5 | ET2 | **Timer 2 Overflow Interrupt Enable bit**<br>Clear to disable Timer 2 overflow interrupt.<br>Set to enable Timer 2 overflow interrupt. |
| 4 | ES | **Serial Port Enable bit**<br>Clear to disable serial port interrupt.<br>Set to enable serial port interrupt. |
| 3 | ET1 | **Timer 1 Overflow Interrupt Enable bit**<br>Clear to disable timer 1 overflow interrupt.<br>Set to enable timer 1 overflow interrupt. |
| 2 | EX1 | **External Interrupt 1 Enable bit**<br>Clear to disable external interrupt 1.<br>Set to enable external interrupt 1. |
| 1 | ET0 | **Timer 0 Overflow Interrupt Enable bit**<br>Clear to disable timer 0 overflow interrupt.<br>Set to enable timer 0 overflow interrupt. |
| 0 | EX0 | **External Interrupt 0 Enable bit**<br>Clear to disable external interrupt 0.<br>Set to enable external interrupt 0. |

Reset Value = 0000 0000b
bit addressable

**Table 111.** IEN1 Register

IEN1 (S:E8h)
Interrupt Enable Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | ESPI | ETIM | EADC | ECAN |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | ESPI | **SPI Interrupt Enable bit**<br>Clear to disable the SPI interrupt.<br>Set to enable the SPI interrupt. |
| 2 | ETIM | **TImer Overrun Interrupt Enable bit**<br>Clear to disable the timer overrun interrupt.<br>Set to enable the timer overrun interrupt. |
| 1 | EADC | **ADC Interrupt Enable bit**<br>Clear to disable the ADC interrupt.<br>Set to enable the ADC interrupt. |
| 0 | ECAN | **CAN Interrupt Enable bit**<br>Clear to disable the CAN interrupt.<br>Set to enable the CAN interrupt. |

Reset Value = xxxx 0000b
bit addressable

**Table 112.** IPL0 Register

IPL0 (S:B8h)
Interrupt Enable Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | PPC | PT2 | PS | PT1 | PX1 | PT0 | PX0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | PPC | **PCA Interrupt Priority bit**<br>Refer to PPCH for priority level |
| 5 | PT2 | **Timer 2 Overflow Interrupt Priority bit**<br>Refer to PT2H for priority level. |
| 4 | PS | **Serial Port Priority bit**<br>Refer to PSH for priority level. |
| 3 | PT1 | **Timer 1 Overflow Interrupt Priority bit**<br>Refer to PT1H for priority level. |
| 2 | PX1 | **External Interrupt 1 Priority bit**<br>Refer to PX1H for priority level. |
| 1 | PT0 | **Timer 0 Overflow Interrupt Priority bit**<br>Refer to PT0H for priority level. |
| 0 | PX0 | **External Interrupt 0 Priority bit**<br>Refer to PX0H for priority level. |

Reset Value = X000 0000b
bit addressable

**Table 113.** IPL1 Register

IPL1 (S:F8h)
Interrupt Priority Low Register 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | SPIL | POVRL | PADCL | PCANL |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | SPIL | **SPI Interrupt Priority Level Less Significant Bit**<br>Refer to SPIH for priority level. |
| 2 | POVRL | **Timer Overrun Interrupt Priority Level Less Significant Bit**<br>Refer to PI2CH for priority level. |
| 1 | PADCL | **ADC Interrupt Priority Level Less Significant Bit**<br>Refer to PSPIH for priority level. |
| 0 | PCANL | **CAN Interrupt Priority Level Less Significant Bit**<br>Refer to PKBH for priority level. |

Reset Value = XXXX 0000b
bit addressable

**Table 114.** IPL0 Register

IPH0 (B7h)
Interrupt High Priority Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | PPCH | PT2H | PSH | PT1H | PX1H | PT0H | PX0H |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | PPCH | **PCA Interrupt Priority Level Most Significant bit**<br>PPCH PPC Priority level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest priority |
| 5 | PT2H | **Timer 2 Overflow Interrupt High Priority bit**<br>PT2H PT2 Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 4 | PSH | **Serial Port High Priority bit**<br>PSH PS Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 3 | PT1H | **Timer 1 Overflow Interrupt High Priority bit**<br>PT1H PT1 Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 2 | PX1H | **External Interrupt 1 High Priority bit**<br>PX1H PX1 Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 1 | PT0H | **Timer 0 Overflow Interrupt High Priority bit**<br>PT0H PT0 Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 0 | PX0H | **External Interrupt 0 high priority bit**<br>PX0H PX0 Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |

Reset Value = X000 0000b

**Table 115.** IPH1 Register

IPH1 (S:F7h)
Interrupt High Priority Register 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | SPIH | POVRH | PADCH | PCANH |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | SPIH | **SPI Interrupt Priority Level Most Significant bit**<br>SPIH SPIL Priority level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 2 | POVRH | **Timer overrun Interrupt Priority Level Most Significant bit**<br>POVRH POVRL Priority level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 1 | PADCH | **ADC Interrupt Priority Level Most Significant bit**<br>PADCH PADCL Priority level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 0 | PCANH | **CAN Interrupt Priority Level Most Significant bit**<br>PCANH PCANL Priority level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |

Reset Value = XXXX 0000b

# Electrical Characteristics

## Absolute Maximum Ratings

| |
|---|
| Ambiant Temperature Under Bias: |
| I = industrial.....-40°C to 85°C<br>A = automotive.....-40°C to +125°C |
| Voltage on $V_{CC}$ from $V_{SS}$ .....-0.5V to + 6V |
| Voltage on Any Pin from $V_{SS}$..... -0.5V to $V_{CC}$ + 0.2V |
| Power Dissipation ..... 1 W |

Note: Stresses at or above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions may affect device reliability.
The power dissipation is based on the maximum allowable die temperature and the thermal resistance of the package.

## ICCOP Test Conditions

**Power Consumption Management**

Since the introduction of the first C51 device, every manufacturer made operating $I_{CC}$ measurements under Reset, which made sense for the designs where the CPU was running under reset. In our new devices, the CPU is no longer active during reset, so the power consumption is very low but not representative of what will happen in the customer system. Thus, while keeping measurements under Reset, we present a new way to measure the operating $I_{CC}$.

Using an internal test ROM, the following code is executed.

Label: SJMP Label (80FE)

Ports 1 and 4 are disconnected, RST = Vcc, XTAL2 is not connected and XTAL1 is driven by the clock.

This is much more representative of the real operating Icc.

## DC Parameters for Standard Voltage

Industrial $T_A$ = -40°C to +85°C; $V_{SS}$ = 0V;
Automotive $T_A$ = -40°C to +125°C; $V_{SS}$ = 0V

$V_{CC}$ =3.0V to 5.5V and F = 0 to 40 MHz (both internal and external code execution)

$V_{CC}$ =4.5V to 5.5V and F = 0 to 60 MHz (internal code execution only)

**Table 116.** DC Parameters in Standard Voltage

| Symbol | Parameter | Min | Typ(5) | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Input Low Voltage | -0.5 | | 0.2Vcc - 0.1 | V | |
| $V_{IH}$ | Input High Voltage except XTAL1, RST | 0.2 $V_{CC}$ + 0.9 | | $V_{CC}$ + 0.5 | V | |
| $V_{IH1}$ | Input High Voltage, XTAL1, RST | 0.7 $V_{CC}$ | | $V_{CC}$ + 0.5 | V | |
| $V_{OL}$ | Output Low Voltage, ports 1, 2, 3 and 4(6) | | | 0.3<br>0.45<br>1.0 | V<br>V<br>V | $I_{OL}$ = 100 µA(4)<br>$I_{OL}$ = 1.6 mA(4)<br>$I_{OL}$ = 3.5 mA(4) |
| $V_{OL1}$ | Output Low Voltage, port 0, ALE, $\overline{PSEN}$ (6) | | | 0.3<br>0.45<br>1.0 | V<br>V<br>V | $I_{OL}$ = 200 µA(4)<br>$I_{OL}$ = 3.2 mA(4)<br>$I_{OL}$ = 7.0 mA(4) |

**Table 116.** DC Parameters in Standard Voltage (Continued)

| Symbol | Parameter | Min | Typ[5] | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|
| $V_{OH}$ | Output High Voltage, ports 1, 2, 3, and 4 | $V_{CC}$ - 0.3<br>$V_{CC}$ - 0.7<br>$V_{CC}$ - 1.5 | | | V<br>V<br>V | $I_{OH}$ = -10 µA<br>$I_{OH}$ = -30 µA<br>$I_{OH}$ = -60 µA<br>$V_{CC}$ = 3V to 5.5V |
| $V_{OH1}$ | Output High Voltage, port 0, ALE, $\overline{PSEN}$ | $V_{CC}$ - 0.3<br>$V_{CC}$ - 0.7<br>$V_{CC}$ - 1.5 | | | V<br>V<br>V | $I_{OH}$ = -200 µA<br>$I_{OH}$ = -3.2 mA<br>$I_{OH}$ = -7.0 mA<br>$V_{CC}$ = 5V ± 10% |
| $R_{RST}$ | RST Pulldown Resistor | 20 | 100 | 200 | kΩ | |
| $I_{IL}$ | Logical 0 Input Current ports 1, 2, 3 and 4 | | | -50 | µA | Vin = 0.45V |
| $I_{LI}$ | Input Leakage Current | | | ±10 | µA | 0.45V < Vin < $V_{CC}$ |
| $I_{TL}$ | Logical 1 to 0 Transition Current, ports 1, 2, 3 and 4 | | | -650 | µA | Vin = 2.0V |
| $C_{IO}$ | Capacitance of I/O Buffer | | | 10 | pF | Fc = 1 MHz<br>$T_A$ = 25°C |
| $I_{PD}$ | Power-down Current Industrial | | 75 | 150 | µA | 3V < $V_{CC}$ < 5.5V[3] |
| | Power-down Current Automotive | | 100 | 350 | µA | 3V < $V_{CC}$ < 5.5V[3] |
| $I_{CC}$ | Power Supply Current | $I_{CCOP}$ = 0.4 Frequency (MHz) + 8<br>$I_{CCIDLE}$ = 0.2 Frequency (MHz) + 8 | | | mA | Vcc = 5.5V[1][2] |
| $I_{CCWRITE}$ | Power Supply Current on flash or EEdata write | | | 0.8 x Frequency (MHz) + 15 | mA | $V_{CC}$ = 5.5V |

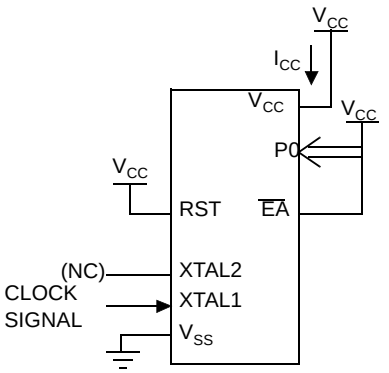
Notes:
1. Operating $I_{CC}$ is measured with all output pins disconnected; XTAL1 driven with $T_{CLCH}$, $T_{CHCL}$ = 5 ns (see Figure 81.), $V_{IL}$ = $V_{SS}$ + 0.5V,
   $V_{IH}$ = $V_{CC}$ - 0.5V; XTAL2 N.C.; $\overline{EA}$ = RST = Port 0 = $V_{CC}$. $I_{CC}$ would be slightly higher if a crystal oscillator used (see Figure 78.).
2. Idle $I_{CC}$ is measured with all output pins disconnected; XTAL1 driven with $T_{CLCH}$, $T_{CHCL}$ = 5 ns, $V_{IL}$ = $V_{SS}$ + 0.5V, $V_{IH}$ = $V_{CC}$ - 0.5V; XTAL2 N.C; Port 0 = $V_{CC}$; $\overline{EA}$ = RST = $V_{SS}$ (see Figure 79.).
3. Power-down $I_{CC}$ is measured with all output pins disconnected; $\overline{EA}$ = $V_{CC}$, PORT 0 = $V_{CC}$; XTAL2 NC.; RST = $V_{SS}$ (see Figure 80.). In addition, the WDT must be inactive and the POF flag must be set.
4. Capacitance loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the $V_{OL}$s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1 to 0 transitions during bus operation. In the worst cases (capacitive loading 100pF), the noise pulse on the ALE line may exceed 0.45V with maxi $V_{OL}$ peak 0.6V. A Schmitt Trigger use is not necessary.
5. Typicals are based on a limited number of samples and are not guaranteed. The values listed are at room temperature.
6. Under steady state (non-transient) conditions, $I_{OL}$ must be externally limited as follows:
   Maximum $I_{OL}$ per port pin: 10 mA
   Maximum $I_{OL}$ per 8-bit port:
   Port 0: 26 mA
   Ports 1, 2, 3 and 4: 15 mA
   Maximum total $I_{OL}$ for all output pins: 71 mA
   If $I_{OL}$ exceeds the test condition, $V_{OL}$ may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

**Power Fail Detect at Ambiant Temperatures**

| VPFDP(1) | VPFDM(2) | Hysterisis |
|---|---|---|
| 2.5V typ | 2.35V typ | 100mV min. |

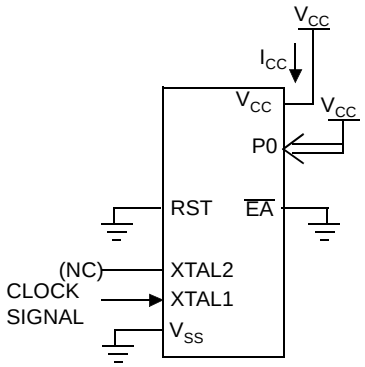
Note: 1. Threshold Voltage for PFD Release
2. Threshold Voltage for PFD Activation

**Figure 78.** $I_{CC}$ Test Condition, Active Mode



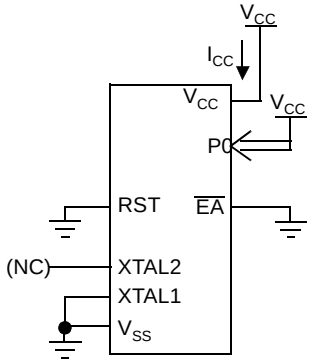

All other pins are disconnected.

**Figure 79.** $I_{CC}$ Test Condition, Idle Mode



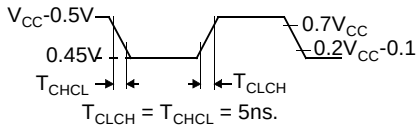

All other pins are disconnected.

**Figure 80.** $I_{CC}$ Test Condition, Power-Down Mode




All other pins are disconnected.

**Figure 81.** Clock Signal Waveform for $I_{CC}$ Tests in Active and Idle Modes




## DC Parameters for A/D Converter

**Table 117.** DC Parameters for AD Converter in Precision Conversion

| Symbol | Parameter | Min | Typ(1),(2) | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|
| AVin | Analog input voltage | Vss- 0.2 | | Vref + 0.2 | V | |
| Rref | Resistance between Vref and Vss | 12 | 16 | 24 | kΩ | |
| Vref(3) | Reference voltage | 2.40 | | 3.00 | V | |
| Cai | Analog input Capacitance | | 60 | | pF | During sampling |
| Rai | Analog input Resistor | | | 400 | Ω | During sampling |
| INL | Integral non linearity | | 1 | 2 | lsb | |
| | | | | 3 | | Automotive |
| DNL | Differential non linearity | | 0.5 | 1 | lsb | |
| OE | Offset error | -2 | | 2 | lsb | |

Note: 1. Typicals are based on a limited number of samples and are not guaranteed.
2. For temperatures higher than 85°C, use standard conversion (8-bit only) and PRS ≥ 2.
3. $V_{REF} \leq V_{CC}$ + 0.2V for temperatures higher than 85.

## AC Parameters

### Explanation of the AC Symbols

Each timing symbol has 5 characters. The first character is always a “T” (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

Example: $T_{AVLL}$ = Time for Address Valid to ALE Low.
$T_{LLPL}$ = Time for ALE Low to PSEN Low.

$T_A$ = -40°C to +85°C; $V_{SS}$ = 0V; $V_{CC}$ = 3V to 5.5V; F = 0 to 40 MHz.

(Load Capacitance for port 0, ALE and PSEN = 60 pF; Load Capacitance for all other outputs = 60 pF.)

Table 118, Table 121 and Table 124 give the description of each AC symbols.

Table 119, Table 123 and Table 125 give for each range the AC parameter.

Table 120, Table 123 and Table 126 give the frequency derating formula of the AC parameter for each speed range description. To calculate each AC symbols: Take the x value and use this value in the formula.

Example: $T_{LLIV}$ and 20 MHz, Standard clock.
x = 30 ns
T = 50 ns
$T_{CCIV}$ = 4T - x = 170 ns

## External Program Memory Characteristics

**Table 118.** Symbol Description

| Symbol | Parameter |
|---|---|
| T | Oscillator clock period |
| $T_{LHLL}$ | ALE pulse width |
| $T_{AVLL}$ | Address Valid to ALE |
| $T_{LLAX}$ | Address Hold After ALE |
| $T_{LLIV}$ | ALE to Valid Instruction In |
| $T_{LLPL}$ | ALE to $\overline{PSEN}$ |
| $T_{PLPH}$ | $\overline{PSEN}$ Pulse Width |
| $T_{PLIV}$ | $\overline{PSEN}$ to Valid Instruction In |
| $T_{PXIX}$ | Input Instruction Hold After $\overline{PSEN}$ |
| $T_{PXIZ}$ | Input Instruction Float After $\overline{PSEN}$ |
| $T_{AVIV}$ | Address to Valid Instruction In |
| $T_{PLAZ}$ | $\overline{PSEN}$ Low to Address Float |

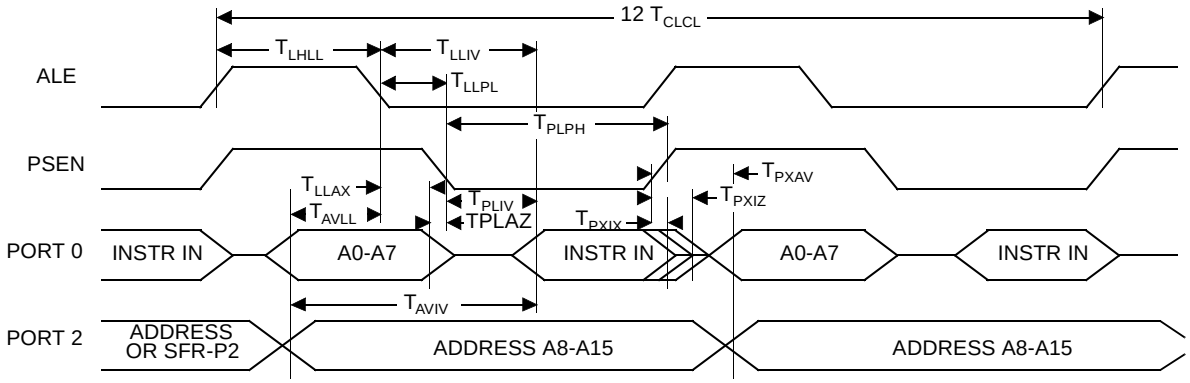
**Table 119.** AC Parameters for a Fix Clock (F = 40 MHz)

| Symbol | Min | Max | Units |
|---|---|---|---|
| T | 25 | | ns |
| $T_{LHLL}$ | 40 | | ns |
| $T_{AVLL}$ | 10 | | ns |
| $T_{LLAX}$ | 10 | | ns |
| $T_{LLIV}$ | | 70 | ns |
| $T_{LLPL}$ | 15 | | ns |
| $T_{PLPH}$ | 55 | | ns |
| $T_{PLIV}$ | | 35 | ns |
| $T_{PXIX}$ | 0 | | ns |
| $T_{PXIZ}$ | | 18 | ns |
| $T_{AVIV}$ | | 85 | ns |
| $T_{PLAZ}$ | | 10 | ns |

**Table 120.** AC Parameters for a Variable Clock

| Symbol | Type | Standard Clock | X2 Clock | X parameter | Units |
|---|---|---|---|---|---|
| $T_{LHLL}$ | Min | 2 T - x | T - x | 10 | ns |
| $T_{AVLL}$ | Min | T - x | 0.5 T - x | 15 | ns |
| $T_{LLAX}$ | Min | T - x | 0.5 T - x | 15 | ns |
| $T_{LLIV}$ | Max | 4 T - x | 2 T - x | 30 | ns |
| $T_{LLPL}$ | Min | T - x | 0.5 T - x | 10 | ns |
| $T_{PLPH}$ | Min | 3 T - x | 1.5 T - x | 20 | ns |
| $T_{PLIV}$ | Max | 3 T - x | 1.5 T - x | 40 | ns |
| $T_{PXIX}$ | Min | x | x | 0 | ns |
| $T_{PXIZ}$ | Max | T - x | 0.5 T - x | 7 | ns |
| $T_{AVIV}$ | Max | 5 T - x | 2.5 T - x | 40 | ns |
| $T_{PLAZ}$ | Max | x | x | 10 | ns |

### External Program Memory Read Cycle



## External Data Memory Characteristics

**Table 121.** Symbol Description

| Symbol | Parameter |
|---|---|
| $T_{RLRH}$ | $\overline{RD}$ Pulse Width |
| $T_{WLWH}$ | $\overline{WR}$ Pulse Width |
| $T_{RLDV}$ | $\overline{RD}$ to Valid Data In |
| $T_{RHDX}$ | Data Hold After $\overline{RD}$ |
| $T_{RHDZ}$ | Data Float After $\overline{RD}$ |
| $T_{LLDV}$ | ALE to Valid Data In |
| $T_{AVDV}$ | Address to Valid Data In |
| $T_{LLWL}$ | ALE to $\overline{WR}$ or $\overline{RD}$ |
| $T_{AVWL}$ | Address to $\overline{WR}$ or $\overline{RD}$ |
| $T_{QVWX}$ | Data Valid to $\overline{WR}$ Transition |
| $T_{QVWH}$ | Data set-up to $\overline{WR}$ High |
| $T_{WHQX}$ | Data Hold After $\overline{WR}$ |
| $T_{RLAZ}$ | $\overline{RD}$ Low to Address Float |
| $T_{WHLH}$ | $\overline{RD}$ or $\overline{WR}$ High to ALE high |

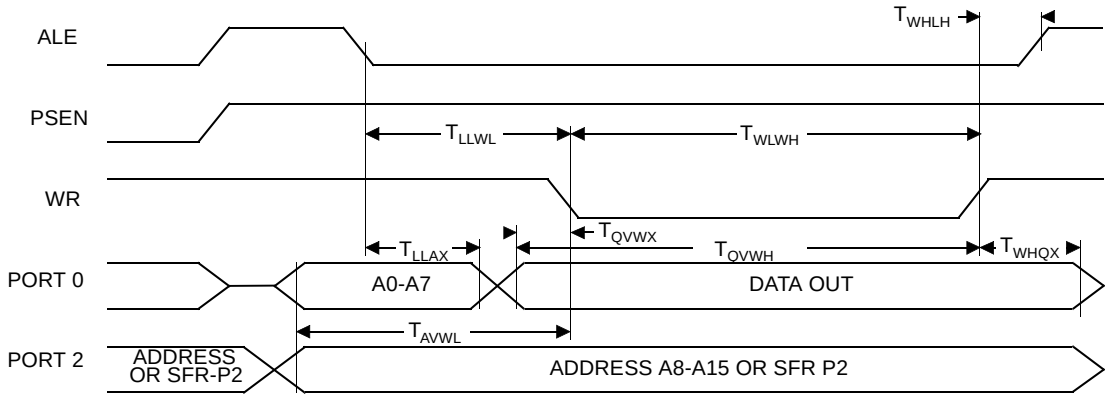
**Table 122.** AC Parameters for a Fix Clock (F=40MHz)

| Symbol | Min | Max | Units |
|---|---|---|---|
| $T_{RLRH}$ | 130 | | ns |
| $T_{WLWH}$ | 130 | | ns |
| $T_{RLDV}$ | | 100 | ns |
| $T_{RHDX}$ | 0 | | ns |
| $T_{RHDZ}$ | | 30 | ns |
| $T_{LLDV}$ | | 160 | ns |
| $T_{AVDV}$ | | 165 | ns |
| $T_{LLWL}$ | 50 | 100 | ns |
| $T_{AVWL}$ | 75 | | ns |
| $T_{QVWX}$ | 10 | | ns |
| $T_{QVWH}$ | 160 | | ns |
| $T_{WHQX}$ | 15 | | ns |
| $T_{RLAZ}$ | | 0 | ns |
| $T_{WHLH}$ | 10 | 40 | ns |

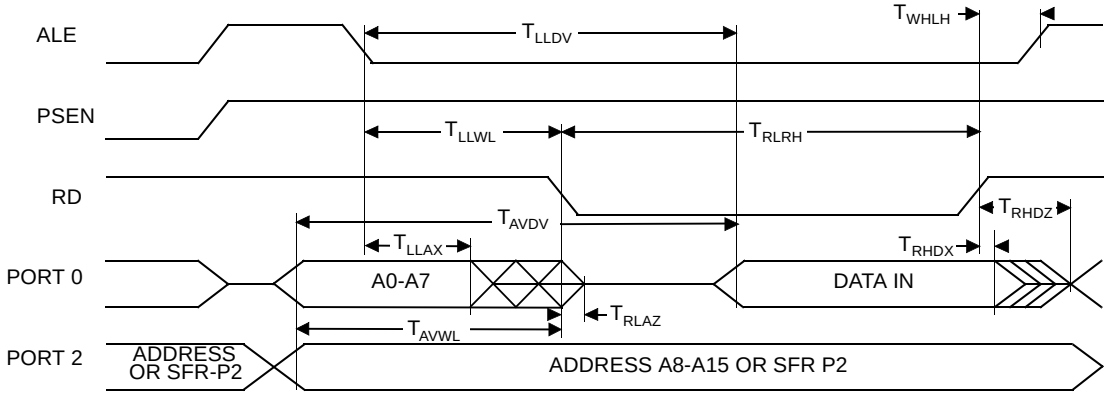
**Table 123.** AC Parameters for a Variable Clock

| Symbol | Type | Standard Clock | X2 Clock | X parameter | Units |
|---|---|---|---|---|---|
| $T_{RLRH}$ | Min | 6 T - x | 3 T - x | 20 | ns |
| $T_{WLWH}$ | Min | 6 T - x | 3 T - x | 20 | ns |
| $T_{RLDV}$ | Max | 5 T - x | 2.5 T - x | 25 | ns |
| $T_{RHDX}$ | Min | x | x | 0 | ns |
| $T_{RHDZ}$ | Max | 2 T - x | T - x | 20 | ns |
| $T_{LLDV}$ | Max | 8 T - x | 4T -x | 40 | ns |
| $T_{AVDV}$ | Max | 9 T - x | 4.5 T - x | 60 | ns |
| $T_{LLWL}$ | Min | 3 T - x | 1.5 T - x | 25 | ns |
| $T_{LLWL}$ | Max | 3 T + x | 1.5 T + x | 25 | ns |
| $T_{AVWL}$ | Min | 4 T - x | 2 T - x | 25 | ns |
| $T_{QVWX}$ | Min | T - x | 0.5 T - x | 15 | ns |
| $T_{QVWH}$ | Min | 7 T - x | 3.5 T - x | 25 | ns |
| $T_{WHQX}$ | Min | T - x | 0.5 T - x | 10 | ns |
| $T_{RLAZ}$ | Max | x | x | 0 | ns |
| $T_{WHLH}$ | Min | T - x | 0.5 T - x | 15 | ns |
| $T_{WHLH}$ | Max | T + x | 0.5 T + x | 15 | ns |

### External Data Memory Write Cycle




### External Data Memory Read Cycle




### Serial Port Timing – Shift Register Mode

**Table 124.** Symbol Description (F = 40 MHz)

| Symbol | Parameter |
|---|---|
| $T_{XLXL}$ | Serial port clock cycle time |
| $T_{QVHX}$ | Output data set-up to clock rising edge |
| $T_{XHQX}$ | Output data hold after clock rising edge |
| $T_{XHDX}$ | Input data hold after clock rising edge |
| $T_{XHDV}$ | Clock rising edge to input data valid |

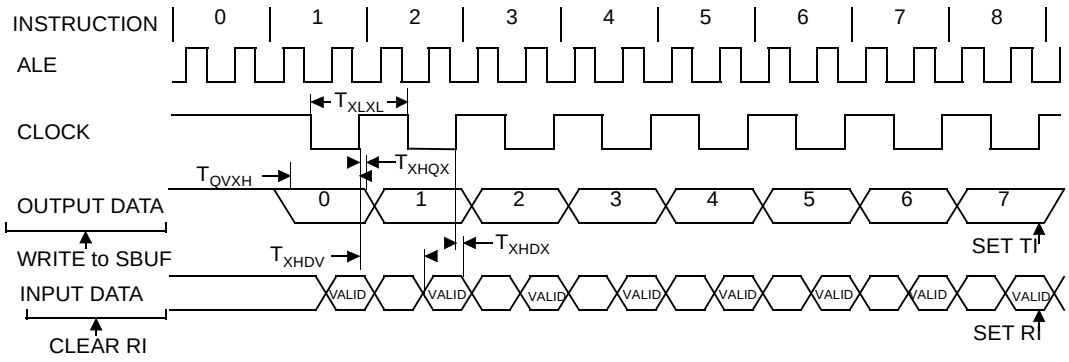
**Table 125.** AC Parameters for a Fix Clock (F = 40 MHz)

| Symbol | Min | Max | Units |
|---|---|---|---|
| $T_{XLXL}$ | 300 | | ns |
| $T_{QVHX}$ | 200 | | ns |
| $T_{XHQX}$ | 30 | | ns |
| $T_{XHDX}$ | 0 | | ns |
| $T_{XHDV}$ | | 117 | ns |

**Table 126.** AC Parameters for a Variable Clock

| Symbol | Type | Standard Clock | X2 Clock | X parameter for -M range | Units |
|---|---|---|---|---|---|
| $T_{XLXL}$ | Min | 12 T | 6 T | | ns |
| $T_{QVHX}$ | Min | 10 T - x | 5 T - x | 50 | ns |
| $T_{XHQX}$ | Min | 2 T - x | T - x | 20 | ns |
| $T_{XHDX}$ | Min | x | x | 0 | ns |
| $T_{XHDV}$ | Max | 10 T - x | 5 T- x | 133 | ns |

## Shift Register Timing Waveforms

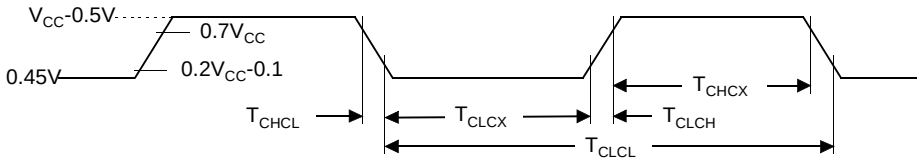



## External Clock Drive Characteristics (XTAL1)

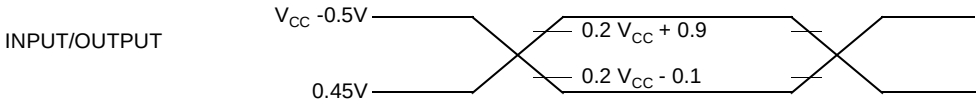
**Table 127.** AC Parameters

| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $T_{CLCL}$ | Oscillator Period | 25 | | ns |
| $T_{CHCX}$ | High Time | 5 | | ns |
| $T_{CLCX}$ | Low Time | 5 | | ns |
| $T_{CLCH}$ | Rise Time | | 5 | ns |
| $T_{CHCL}$ | Fall Time | | 5 | ns |
| $T_{CHCX}/T_{CLCX}$ | Cyclic ratio in X2 mode | 40 | 60 | % |

### External Clock Drive Waveforms

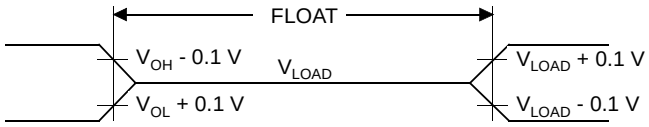



### AC Testing Input/Output Waveforms



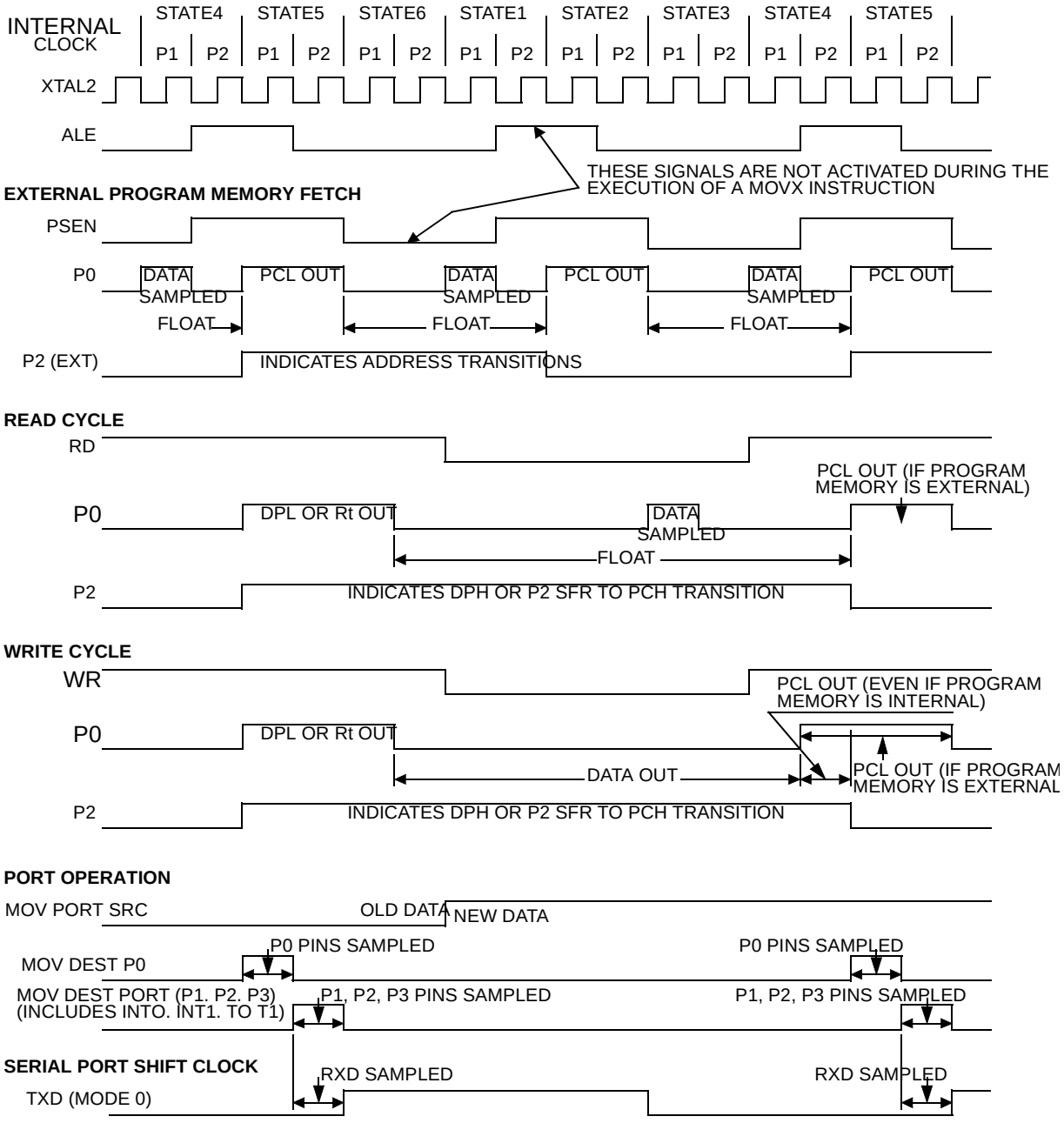

AC inputs during testing are driven at $V_{CC}$ - 0.5 for a logic “1” and 0.45V for a logic “0”. Timing measurement are made at $V_{IH}$ min for a logic “1” and $V_{IL}$ max for a logic “0”.

### Float Waveforms




For timing purposes as port pin is no longer floating when a 100 mV change from load voltage occurs and begins to float when a 100 mV change from the loaded $V_{OH}/V_{OL}$ level occurs. $I_{OL}/I_{OH} \geq \pm$ 20 mA.

**Clock Waveforms** Valid in normal clock mode. In X2 mode XTAL2 must be changed to XTAL2/2.




This diagram indicates when signals are clocked internally. The time it takes the signals to propagate to the pins, however, ranges from 25 to 125 ns. This propagation delay is dependent on variables such as temperature and pin loading. Propagation also varies from output to output and component. Typically though ($T_A$=25°C fully loaded) RD and WR propagation delays are approximately 50ns. The other signals are typically 85 ns. Propagation delays are incorporated in the AC specifications.

## Flash/EEPROM Memory

**Table 128.** Timing Symbol Definitions

| Signals | |
|---|---|
| S (Hardware condition) | PSEN#,EA |
| R | RST |
| B | FBUSY flag |

| Conditions | |
|---|---|
| L | Low |
| V | Valid |
| X | No Longer Valid |

**Table 129.** Memory AC Timing

VDD = 3V to 5.5V, TA = -40 to +85°C

| Symbol | Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| $T_{SVRL}$ | Input PSEN# Valid to RST Edge | 50 | | | ns |
| $T_{RLSX}$ | Input PSEN# Hold after RST Edge | 50 | | | ns |
| $T_{BHBL}$ | Flash/EEPROM Internal Busy (Programming) Time | | 10 | | ms |
| $N_{FCY}$ | Number of Flash/EEPROM Erase/Write Cycles | 100 000 | | | cycles |
| $T_{FDR}$ | Flash/EEPROM Data Retention Time | 10 | | | years |

**Figure 82.** Flash Memory – ISP Waveforms

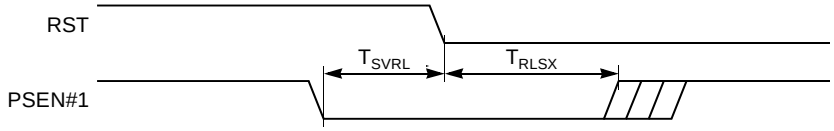
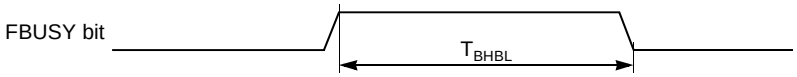



**Figure 83.** Flash Memory – Internal Busy Waveforms




## A/D Converter

**Table 130.** AC Parameters for A/D Conversion

| Symbol | Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| $T_{SETUP}$ | | 4 | | | µs |
| ADC Clock Frequency | | | 700 | | KHz |

## Timings

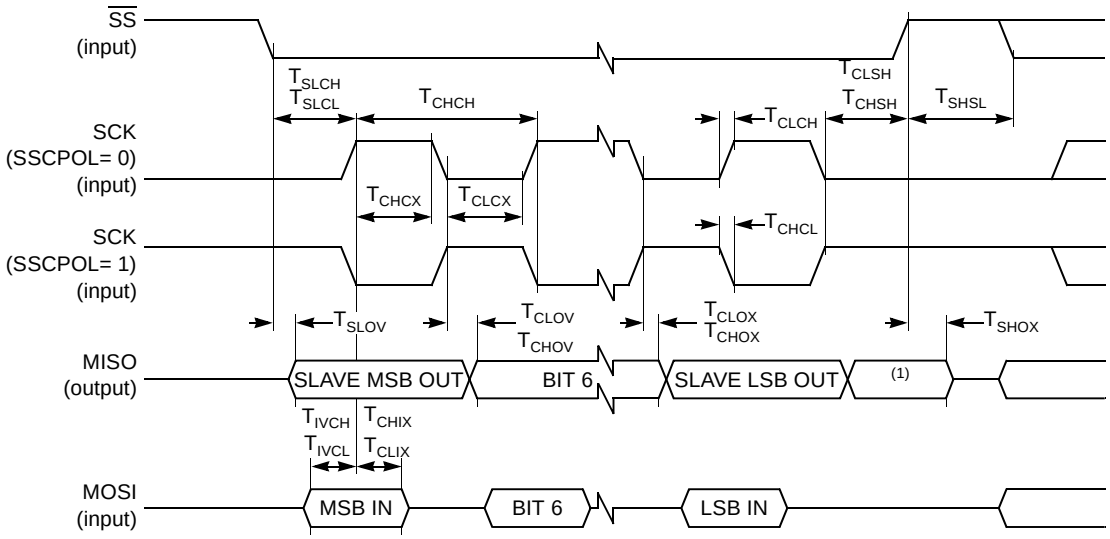
Test conditions: capacitive load on all pins= 60 pF.

**Table 1.** SPI Interface Master AC Timing
**$V_{DD}$ = 2.7 to 3.3 V, $T_A$ = -40 to +85°C**

| Symbol | Parameter | Min | Max | Unit |
|---|---|---|---|---|
| **Slave Mode** | | | | |
| $T_{CHCH}$ | Clock Period | $6^{(1)}$ | | $T_{PER}$ |
| $T_{CHCX}$ | Clock High Time | $3^{(1)}$ | | $T_{PER}$ |
| $T_{CLCX}$ | Clock Low Time | $3^{(1)}$ | | $T_{PER}$ |
| $T_{SLCH}$, $T_{SLCL}$ | $\overline{SS}$ Low to Clock edge | $4T_{PER}$-20ns$^{(1)}$ | | ns |
| $T_{IVCL}$, $T_{IVCH}$ | Input Data Valid to Clock Edge | 50 | | ns |
| $T_{CLIX}$, $T_{CHIX}$ | Input Data Hold after Clock Edge | 50 | | ns |
| $T_{CLOV,}$ $T_{CHOV}$ | Output Data Valid after Clock Edge | | 50 | ns |
| $T_{CLOX}$, $T_{CHOX}$ | Output Data Hold Time after Clock Edge | 0 | | ns |
| $T_{CLSH}$, $T_{CHSH}$ | $\overline{SS}$ High after Clock Edge | $4T_{PER}$+20ns$^{(1)}$ | | ns |
| $T_{SLOV}$ | $\overline{SS}$ Low to Output Data Valid | | $4T_{PER}$+20ns$^{(1)}$ | ns |
| $T_{SHOX}$ | Output Data Hold after $\overline{SS}$ High | | $2T_{PER}$+100ns$^{(1)}$ | ns |
| $T_{SHSL}$ | $\overline{SS}$ High to $\overline{SS}$ Low | $2T_{PER}$+120ns$^{(1)}$ | | |
| $T_{OLOH}$ | Output Rise time | | 100 | ns |
| $T_{OHOL}$ | Output Fall Time | | 100 | ns |
| **Master Mode** | | | | |
| $T_{CHCH}$ | Clock Period | $4^{(1)}$ | | $T_{PER}$ |
| $T_{CHCX}$ | Clock High Time | $2T_{PER}$-20ns$^{(1)}$ | | $T_{PER}$ |
| $T_{CLCX}$ | Clock Low Time | $2T_{PER}$-20ns$^{(1)}$ | | $T_{PER}$ |
| $T_{IVCL}$, $T_{IVCH}$ | Input Data Valid to Clock Edge | 50 | | ns |
| $T_{CLIX}$, $T_{CHIX}$ | Input Data Hold after Clock Edge | 0 | | ns |
| $T_{CLOV,}$ $T_{CHOV}$ | Output Data Valid after Clock Edge | | 20 | ns |
| $T_{CLOX}$, $T_{CHOX}$ | Output Data Hold Time after Clock Edge | 0 | | ns |
| $T_{CLCH}$ | Output Data Rise time | | 100 | ns |
| $T_{CHCL}$ | Output Data Fall Time | | 100 | ns |

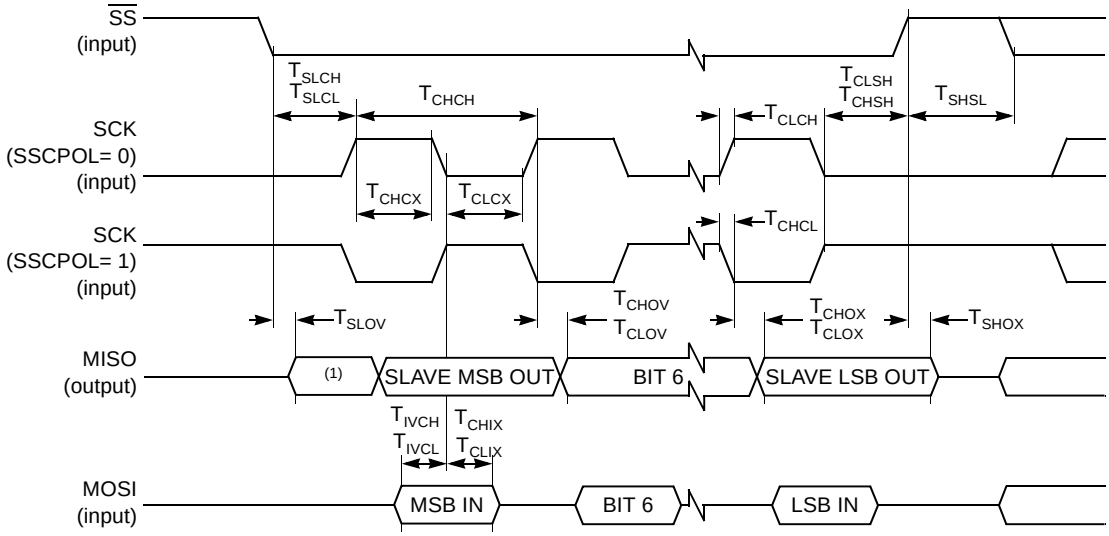
Note: 1. Value of this parameter depends on prescacler ratio defined in bits 0,1 and 7 of SCON Register.In the above table, the ratio used is 4. As it can be set also to 8, 16, 32, 64 or 128, the factor of $T_{PER}$ must be changed according to the new ratio.E.g. 2TPER-20ns(1) will be changed to 4TPER-20ns(1) if the prescaler ratio equals 8.

**Figure 84.** SPI Slave Waveforms (SSCPHA= 0)



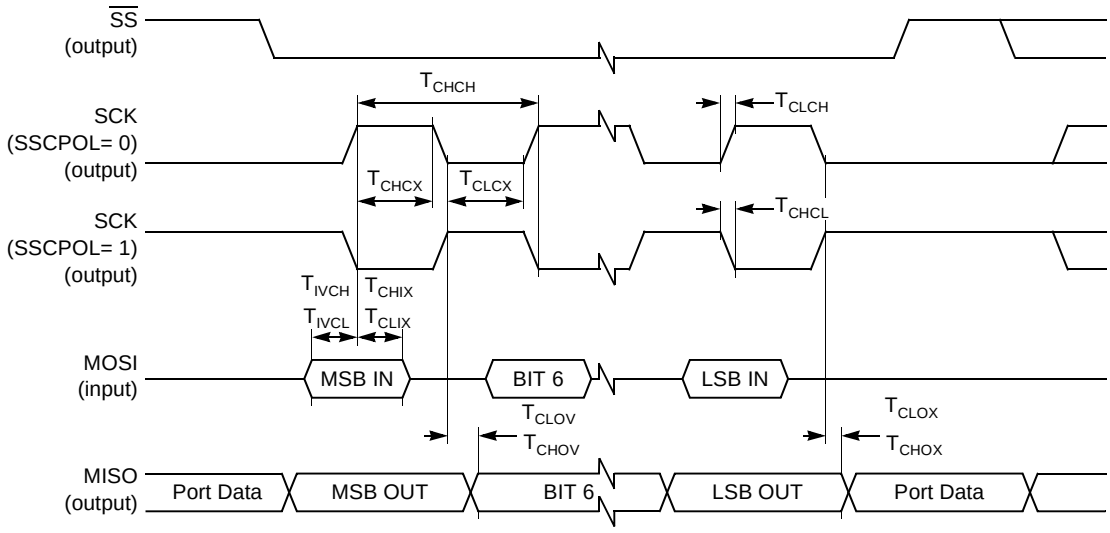

Note: 1. Not Defined but generally the MSB of the character which has just been received.

**Figure 85.** SPI Slave Waveforms (SSCPHA= 1)



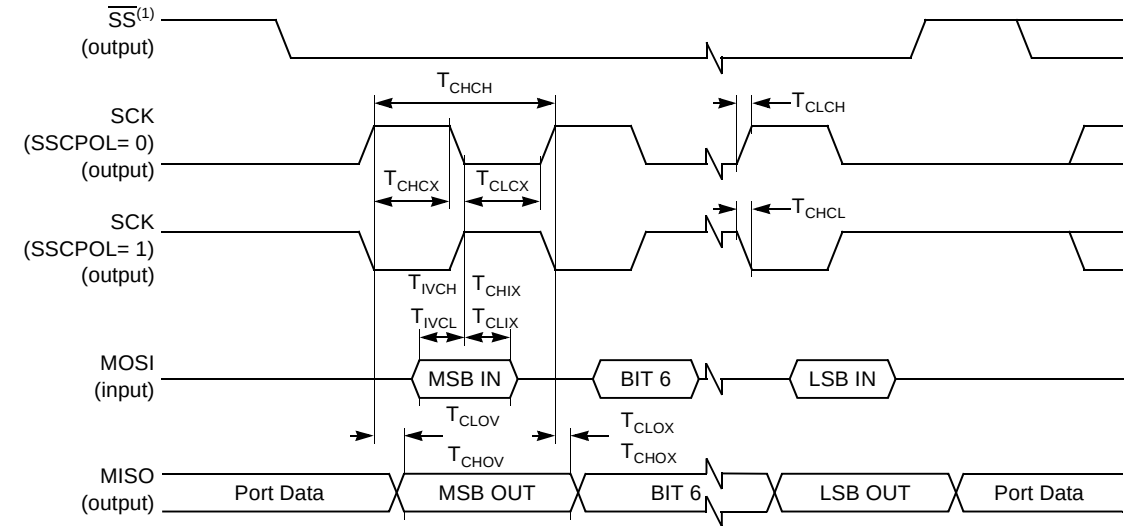

Note: 1. Not Defined but generally the LSB of the character which has just been received.

**Figure 86.** SPI Master Waveforms (SSCPHA= 0)




Note: 1. $\overline{SS}$ handled by software using general purpose port pin.

**Figure 87.** SPI Master Waveforms (SSCPHA= 1)




Note: 1. $\overline{SS}$ handled by software using general purpose port pin.

Note:

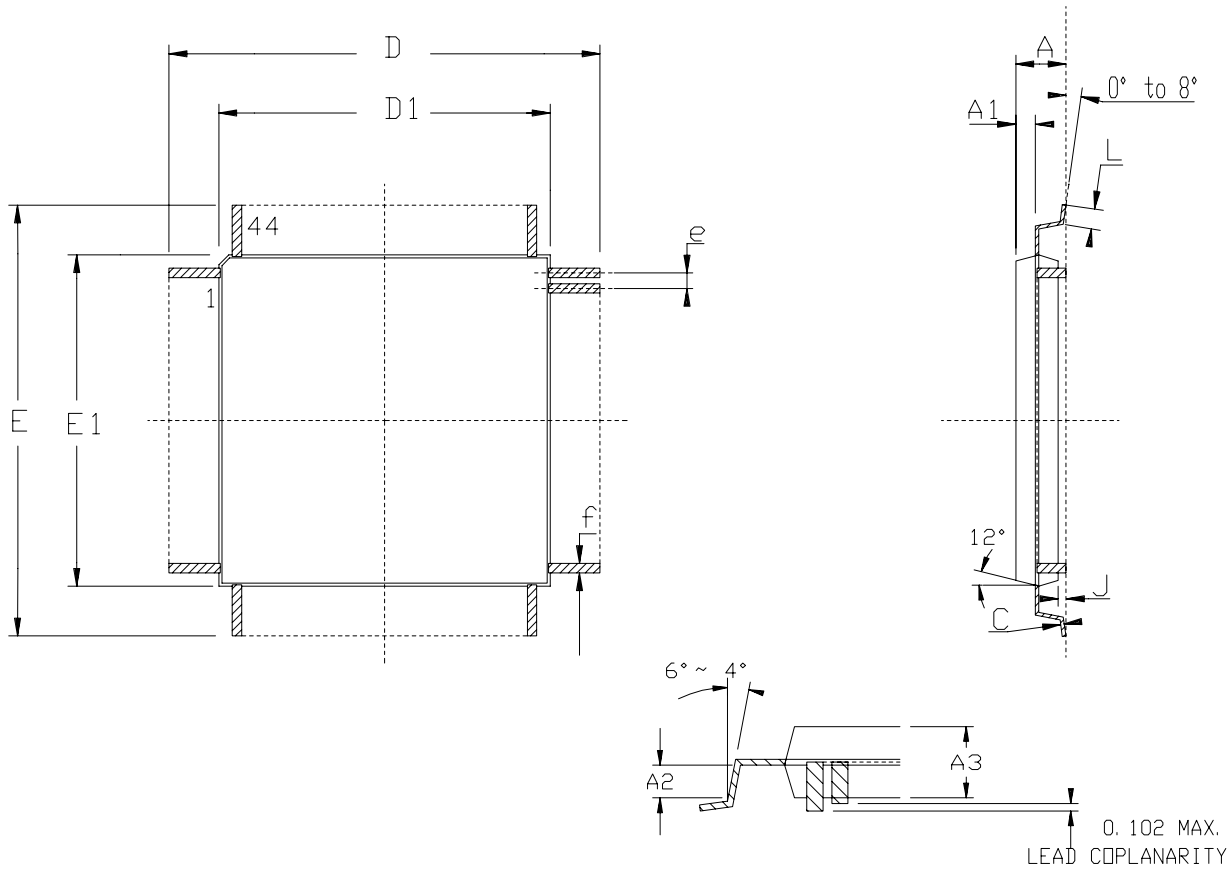
## Ordering Information

**Table 131.** Possible Order Entries

| Part Number | Boot Loader | Temperature Range | Quality Grade | Package | Packing | Product Marking |
|---|---|---|---|---|---|---|
| AT89C51CC03U-7CTIM | OBSOLETE | | | | | |
| AT89C51CC03U-RLTIM | | | | | | |
| AT89C51CC03U-SLSIM | | | | | | |
| AT89C51CC03C-7CTIM | | | | | | |
| AT89C51CC03C-RLTIM | | | | | | |
| AT89C51CC03C-SLSIM | | | | | | |
| AT89C51CC03U-RDTIM | | | | | | |
| AT89C51CC03U-S3SIM | | | | | | |
| AT89C51CC03C-RDTIM | | | | | | |
| AT89C51CC03C-S3SIM | | | | | | |
| | | | | | | |
| AT89C51CC03UA-RLTUM | UART | -40 to +85°C | Industrial & Green | VQFP44 | Tray | 89C51CC03UA-UM |
| AT89C51CC03UA-SLSUM | UART | -40 to +85°C | Industrial & Green | PLCC44 | Stick | 89C51CC03UA-UM |
| AT89C51CC03CA-RLTUM | CAN | -40 to +85°C | Industrial & Green | VQFP44 | Tray | 89C51CC03CA-UM |
| AT89C51CC03CA-SLSUM | CAN | -40 to +85°C | Industrial & Green | PLCC44 | Stick | 89C51CC03CA-UM |
| AT89C51CC03UA-RDTUM | UART | -40 to +85°C | Industrial & Green | VQFP64 | Tray | 89C51CC03UA-UM |
| AT89C51CC03UA-S3SUM | UART | -40 to +85°C | Industrial & Green | PLCC52 | Stick | 89C51CC03UA-UM |
| AT89C51CC03CA-S3SUM | CAN | -40 to +85°C | Industrial & Green | PLCC52 | Stick | 89C51CC03CA-UM |
| AT89C51CC03CA-RDTUM | CAN | -40 to +85°C | Industrial & Green | VQFP64 | Tray | 89C51CC03CA-UM |

## Package Drawings

### VQFP44




| | MM | | INCH | |
|---|---|---|---|---|
| | Min | Max | Min | Max |
| A | – | 1.60 | – | .063 |
| A1 | 0.64 REF | | .025 REF | |
| A2 | 0.64 REF | | .025 REF | |
| A3 | 1.35 | 1.45 | .053 | .057 |
| D | 11.90 | 12.10 | .468 | .476 |
| D1 | 9.90 | 10.10 | .390 | .398 |
| E | 11.90 | 12.10 | .468 | .476 |
| E1 | 9.90 | 10.10 | .390 | .398 |
| J | 0.05 | – | .002 | – |
| L | 0.45 | 0.75 | .018 | .030 |
| e | 0.80 BSC | | .0315 BSC | |
| f | 0.35 BSC | | .014 BSC | |

## STANDARD NOTES FOR PQFP/ VQFP / TQFP / DQFP

1/ CONTROLLING DIMENSIONS : INCHES

2/ ALL DIMENSIONING AND TOLERANCING CONFORM TO ANSI Y 14.5M - 1982.

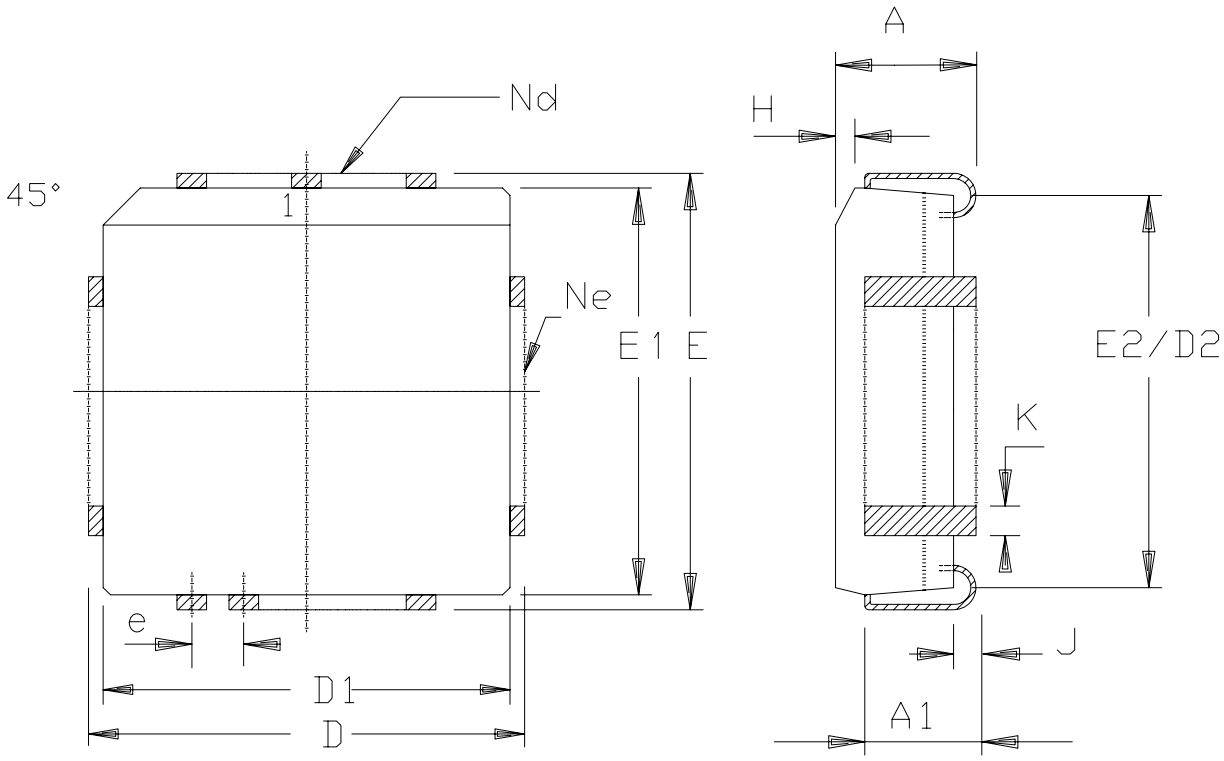
3/ "D1 AND E1" DIMENSIONS DO NOT INCLUDE MOLD PROTUSIONS. MOLD PROTUSIONS SHALL NOT EXCEED 0.25 mm (0.010 INCH). THE TOP PACKAGE BODY SIZE MAY BE SMALLER THAN THE BOTTOM PACKAGE BODY SIZE BY AS MUCH AS 0.15 mm.

4/ DATUM PLANE "H" LOCATED AT MOLD PARTING LINE AND COINCIDENT WITH LEAD, WHERE LEAD EXITS PLASTIC BODY AT BOTTOM OF PARTING LINE.

5/ DATUM "A" AND "D" TO BE DETERMINED AT DATUM PLANE H.

6/ DIMENSION " f " DOES NOT INCLUDE DAMBAR PROTUSION ALLOWABLE DAMBAR PROTUSION SHALL BE 0.08mm/.003" TOTAL IN EXCESS OF THE " f " DIMENSION AT MAXIMUM MATERIAL CONDITION . DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.

**PLCC44**




| | MM | | INCH | |
|---|---|---|---|---|
| A | 4.20 | 4.57 | .165 | .180 |
| A1 | 2.29 | 3.04 | .090 | .120 |
| D | 17.40 | 17.65 | .685 | .695 |
| D1 | 16.44 | 16.66 | .647 | .656 |
| D2 | 14.99 | 16.00 | .590 | .630 |
| E | 17.40 | 17.65 | .685 | .695 |
| E1 | 16.44 | 16.66 | .647 | .656 |
| E2 | 14.99 | 16.00 | .590 | .630 |
| e | 1.27 | BSC | .050 | BSC |
| H | 1.07 | 1.42 | .042 | .056 |
| J | 0.51 | - | .020 | - |
| K | 0.33 | 0.53 | .013 | .021 |
| Nd | 11 | | 11 | |
| Ne | 11 | | 11 | |
| PKG STD | 00 | | | |

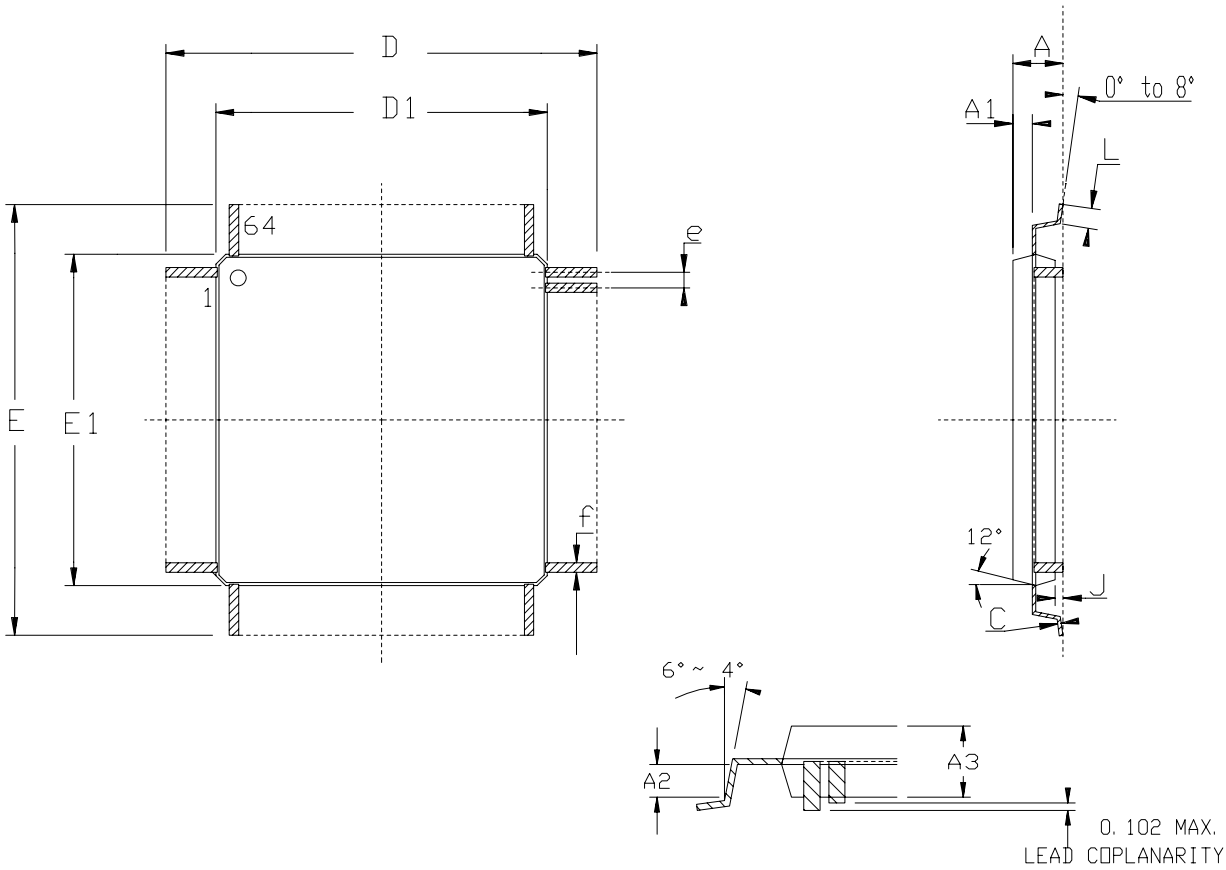
**STANDARD NOTES FOR PLCC**

**1/ CONTROLLING DIMENSIONS : INCHES**

**2/ DIMENSIONING AND TOLERANCING PER ANSI Y 14.5M - 1982.**

**3/ "D" AND "E1" DIMENSIONS DO NOT INCLUDE MOLD FLASH OR PROTUSIONS. MOLD FLASH OR PROTUSIONS SHALL NOT EXCEED 0.20 mm (.008 INCH) PER SIDE.**

## VQFP64




| | MM | | INCH | |
|---|---|---|---|---|
| | Min | Max | Min | Max |
| A | – | 1.60 | – | .063 |
| A1 | 0.64 REF | | .025 REF | |
| A2 | 0.64 REF | | .025 REF | |
| A3 | 1.35 | 1.45 | .053 | .057 |
| D | 11.75 | 12.25 | .463 | .483 |
| D1 | 9.90 | 10.10 | .390 | .398 |
| E | 11.75 | 12.25 | .463 | .483 |
| E1 | 9.90 | 10.10 | .390 | .398 |
| J | 0.05 | – | .002 | – |
| L | 0.45 | 0.75 | .018 | .030 |
| e | 0.50 BSC | | .0197 BSC | |
| f | 0.25 BSC | | .010 BSC | |

## STANDARD NOTES FOR PQFP/ VQFP / TQFP / DQFP

**1/ CONTROLLING DIMENSIONS : INCHES**

**2/ ALL DIMENSIONING AND TOLERANCING CONFORM TO ANSI Y 14.5M - 1982.**

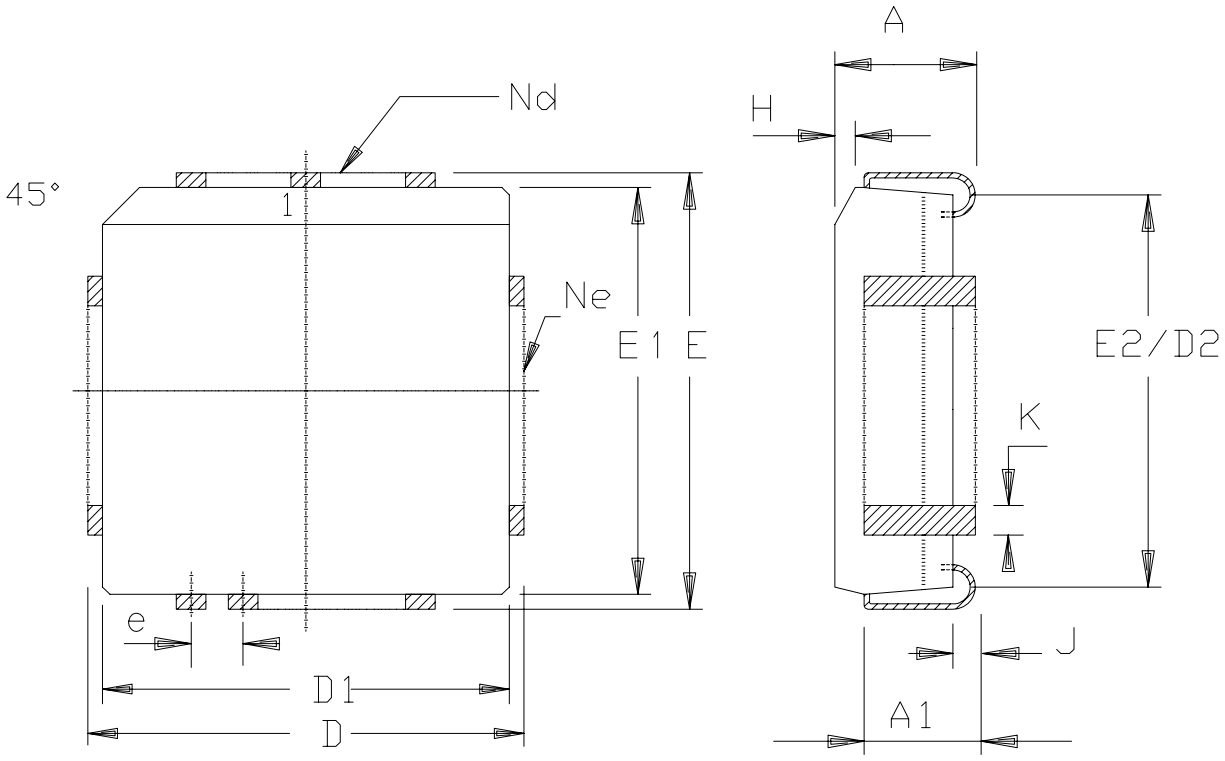
**3/ "D1 AND E1" DIMENSIONS DO NOT INCLUDE MOLD PROTUSIONS. MOLD PROTUSIONS SHALL NOT EXCEED 0.25 mm (0.010 INCH). THE TOP PACKAGE BODY SIZE MAY BE SMALLER THAN THE BOTTOM PACKAGE BODY SIZE BY AS MUCH AS 0.15 mm.**

**4/ DATUM PLANE "H" LOCATED AT MOLD PARTING LINE AND COINCIDENT WITH LEAD, WHERE LEAD EXITS PLASTIC BODY AT BOTTOM OF PARTING LINE.**

**5/ DATUM "A" AND "D" TO BE DETERMINED AT DATUM PLANE H.**

**6/ DIMENSION " f " DOES NOT INCLUDE DAMBAR PROTUSION ALLOWABLE DAMBAR PROTUSION SHALL BE 0.08mm/.003" TOTAL IN EXCESS OF THE " f " DIMENSION AT MAXIMUM MATERIAL CONDITION . DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.**

**PLCC52**




| | MM | | INCH | |
|---|---|---|---|---|
| A | 4.20 | 4.57 | .165 | .180 |
| A1 | 2.29 | 3.30 | .090 | .130 |
| D | 19.94 | 20.19 | .785 | .795 |
| D1 | 19.05 | 19.25 | .750 | .758 |
| D2 | 17.53 | 18.54 | .690 | .730 |
| E | 19.94 | 20.19 | .785 | .795 |
| E1 | 19.05 | 19.25 | .750 | .758 |
| E2 | 17.53 | 18.54 | .690 | .730 |
| e | 1.27 | BSC | .050 | BSC |
| H | 1.07 | 1.42 | .042 | .056 |
| J | 0.51 | – | .020 | – |
| K | 0.33 | 0.53 | .013 | .021 |
| Nd | 13 | | 13 | |
| Ne | 13 | | 13 | |
| PKG STD | 00 | | | |

## Datasheet Change Log

**Changes from 4182B - 09/03 to 4182C 12/03**

1. Added Icc Idle, IPD, and Rrst value in “DC Parameters for A/D Converter” on page 171.

**Changes from 4182C - 12/03 to 4182D 01/04**

1. Updated SFR Table.
   - SFR : SPSTR changed to SPSCR
   - CANSTMH changed to CANSTMPH p15
   - CANSTML changed to CANSTMPL p15
   - CANCONC changed to CANCONCH p15
2. AC/DC - p.160 IccOP and ICCIdle formulas changed
3. Changed maximum frequency to 60MHz in internal code execution.

**Changes from 4182D - 01/04 to 4182E 05/04**

1. Added Automotive temperature range.

**Changes from 4182E - 05/04 to 4182F 10/04**

1. Various minor corrections throughout the document.

**Changes from 4182F - 10/04 to 4182G 03/05**

1. Change to Watchdog formula, Section “Watchdog Programming”, page 83.

**Changes from 4182G 03/05 to 4182H 04/05**

1. Refined automotive temperature values.

**Changes from 4182H 04/05 to 4182I 06/05**

1. Added Green product ordering information.
2. Clarification in Waveform diagram, page 20.

**Changes from 4182I 06/05 to 4182J 03/06**

1. Additional part numbers added to ordering information.

**Changes from 4182J 03/06 to 4182K 04/06**

1. Minor corrections throughout the document to incorrect values.

**Changes from 4182K 04/06 to 4182L 06/07**

1. Modification to ordering information, removed Automotive product versions.

**Changes from 4182L 06/07 to 4182M 02/08**

1. Modification to ordering information, removed non green product versions.

**Changes from 4182M 02/087 to 4182N 03/08**

1. Removed CA-BGA package offering from ordering information.
2. Updated package drawings.

### Changes from 4182N 03/08 to 4182O 09/08

1. Correction to SPDT register address Table 94 on page 139.

# Table of Contents


***Features ..... 1***

***Description ..... 2***

***Block Diagram ..... 2***

***Pin Configuration ..... 3***

I/O Configurations ..... 7
Port 1, Port 3 and Port 4 ..... 7
Port 0 and Port 2 ..... 8
Read-Modify-Write Instructions ..... 9
Quasi-Bidirectional Port Operation ..... 10

***SFR Mapping ..... 11***

***Clock ..... 17***

Description ..... 17
Registers ..... 20

***Data Memory ..... 22***

Internal Space ..... 23
External Space ..... 24
Dual Data Pointer ..... 26
Registers ..... 27

***Power Monitor ..... 29***

Description ..... 29

***Reset ..... 31***

Introduction ..... 31
Reset Input ..... 31
Reset Output ..... 32

***Power Management ..... 33***

Introduction ..... 33
Idle Mode ..... 33
Power-Down Mode ..... 33
Registers ..... 36

***EEPROM Data Memory ..... 37***

Write Data in the Column Latches ..... 37
Programming ..... 37
Read Data ..... 37
Examples ..... 38

Registers ..... 39

***Program/Code Memory ..... 40***

External Code Memory Access ..... 41
Flash Memory Architecture ..... 42
Overview of FM0 Operations ..... 46

***Operation Cross Memory Access ..... 55***

***Sharing Instructions ..... 56***

***In-System Programming (ISP) ..... 58***

Flash Programming and Erasure ..... 58
Boot Process ..... 58
Application Programming Interface ..... 60
XROW Bytes ..... 60
Hardware Security Byte ..... 61

***Serial I/O Port ..... 62***

Framing Error Detection ..... 62
Automatic Address Recognition ..... 63
Given Address ..... 64
Broadcast Address ..... 64
Registers ..... 65

***Timers/Counters ..... 68***

Timer/Counter Operations ..... 68
Timer 0 ..... 68
Timer 1 ..... 71
Interrupt ..... 72
Registers ..... 72

***Timer 2 ..... 76***

Auto-Reload Mode ..... 76
Programmable Clock-Output ..... 77
Registers ..... 78

***Watchdog Timer ..... 81***

Watchdog Programming ..... 82
Watchdog Timer During Power-down Mode and Idle ..... 83

***CAN Controller ..... 85***

CAN Protocol ..... 85
CAN Controller Description ..... 89
CAN Controller Mailbox and Registers Organization ..... 90
CAN Controller Management ..... 92

IT CAN Management ..... 94
Bit Timing and Baud Rate ..... 96
Fault Confinement ..... 98
Acceptance Filter ..... 99
Data and Remote Frame ..... 100
Time Trigger Communication (TTC) and Message Stamping ..... 101
CAN Autobaud and Listening Mode ..... 102
Routines Examples ..... 102
CAN SFR's ..... 105
Registers ..... 106

***Serial Port Interface (SPI) ..... 129***
Features ..... 129
Signal Description ..... 129
Functional Description ..... 131

***Programmable Counter Array (PCA) ..... 141***
PCA Timer ..... 141
PCA Modules ..... 142
PCA Interrupt ..... 143
PCA Capture Mode ..... 143
16-bit Software Timer Mode ..... 144
High Speed Output Mode ..... 145
Pulse Width Modulator Mode ..... 145
PCA WatchDog Timer ..... 146
PCA Registers ..... 147

***Analog-to-Digital Converter (ADC) ..... 152***
Features ..... 152
ADC Port1 I/O Functions ..... 152
ADC Converter Operation ..... 154
Voltage Conversion ..... 154
Clock Selection ..... 154
ADC Standby Mode ..... 155
IT ADC Management ..... 155
Routines examples ..... 155
Registers ..... 157

***Interrupt System ..... 160***
Introduction ..... 160
Registers ..... 162

***Electrical Characteristics ..... 168***
Absolute Maximum Ratings ..... 168
ICCOP Test Conditions ..... 168
DC Parameters for Standard Voltage ..... 168

DC Parameters for A/D Converter ..... 171
AC Parameters ..... 171
Timings ..... 181

***Ordering Information ..... 184***

***Package Drawings ..... 185***
VQFP44 ..... 185
PLCC44 ..... 187
VQFP64 ..... 189
PLCC52 ..... 191

***Datasheet Change Log ..... 192***
Changes from 4182B - 09/03 to 4182C 12/03 ..... 192
Changes from 4182C - 12/03 to 4182D 01/04 ..... 192
Changes from 4182D - 01/04 to 4182E 05/04 ..... 192
Changes from 4182E -05/04 to 4182F 10/04 ..... 192
Changes from 4182F - 10/04 to 4182G 03/05 ..... 192
Changes from 4182G 03/05 to 4182H 04/05 ..... 192
Changes from 4182H 04/05 to 4182I 06/05 ..... 192
Changes from 4182I 06/05 to 4182J 03/06 ..... 192
Changes from 4182J 03/06 to 4182K 04/06 ..... 192
Changes from 4182K 04/06 to 4182L 06/07 ..... 192
Changes from 4182L 06/07 to 4182M 02/08 ..... 192
Changes from 4182M 02/087 to 4182N 03/08 ..... 192
Changes from 4182N 03/08 to 4182O 09/08 ..... 193

***Table of Contents ..... i***

## Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

## Regional Headquarters

***Europe***
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

***Asia***
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

***Japan***
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

## Atmel Operations

***Memory***
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

***Microcontrollers***
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

***ASIC/ASSP/Smart Cards***
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

***RF/Automotive***
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

***Biometrics/Imaging/Hi-Rel MPU/***
***High Speed Converters/RF Datacom***
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

***Literature Requests***
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise,to anyintellectualproperty right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDI-TIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORYWARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULARPURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDEN-TAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUTOF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes norepresentationsor warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specificationsand product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

**©2008 Atmel Corporation**. **All rights reserved.** Atmel®, logo and combinations thereof, are registered trademarks, or the trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

Printed on recycled paper.

4182O–CAN–09/08

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View AT89C51CC03UA-RLTUM on WIN SOURCE
- Atmel Information

## Optimize Your Supply Chain with WIN SOURCE Solutions

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