# 80C51 FAMILY DERIVATIVES

## 8XC552/562 overview

1996 Aug 06

Philips
Semiconductors

PHILIPS

## 8XC552 OVERVIEW

The 8XC552 is a stand-alone high-performance microcontroller designed for use in real-time applications such as instrumentation, industrial control, and automotive control applications such as engine management and transmission control. The device provides, in addition to the 80C51 standard functions, a number of dedicated hardware functions for these applications.

The 8XC552 single-chip 8-bit microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 8XC552 uses the powerful instruction set of the 80C51. Additional special function registers are incorporated to control the on-chip peripherals. Three versions of the derivative exist although the generic term “8XC552” is used to refer to family members:

83C552: 8k bytes mask-programmable ROM, 256 bytes RAM

87C552: 8k bytes EPROM, 256 bytes RAM

80C552: ROMless version of the 83C552

The 8XC552 contains a nonvolatile 8k × 8 read-only program memory, a volatile 256 × 8 read/write data memory, five 8-bit I/O ports and one 8-bit input port, two 16-bit timer/event counters (identical to the timers of the 80C51), an additional 16-bit timer coupled to capture and compare latches, a fifteen-source, two-priority-level, nested interrupt structure, an 8-input ADC, a dual DAC pulse width modulated interface, two serial interfaces (UART and $I^2C$ bus), a “watchdog” timer, and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC552 can be expanded using standard TTL compatible memories and logic

The 8XC552 has two software selectable modes of reduced activity for further power reduction—Idle and Power-down. The idle mode freezes the CPU and resets Timer T2 and the ADC and PWM circuitry but allows the other timers, RAM, serial ports, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to become inoperative.

## 83C562 OVERVIEW

The 83C562 has been derived from the 8XC552 with the following changes:

- The SIO1 ($I^2C$) interface has been omitted.
- The output of port lines P1.6 and P1.7 have a standard configuration instead of open drain.
- The resolution of the A/D converter is decreased from 10 bits to 8 bits.
- The time of an A/D conversion has decreased from 50 machine cycles to 24 machine cycles.

All other functions, pinning and packaging are unchanged.

This chapter of the users’ guide can be used for the 83C562 by omitting or changing the following:

- Disregard the description of SIO1 ($I^2C$).
- The SFRs for the interface: S1ADR, S1DAT, S1STA, and S1CON are not implemented. The two SIO1 related flags ES1 in SFR IEN0 and PS1 in SFR IP0 are also not implemented. These two flag locations are undefined after RESET. The interrupt vector for SIO1 is not used.
- Port lines P1.6 and P1.7 are not open drain but have the same standard configuration and electrical characteristics as P1.0-P1.5. Port lines P1.6 and P1.7 have alternative functions.
- The A/D converter has a resolution of 8 bits instead of 10 bits and consequently the two high-order bits 6 and 7 of SFR ADCON are not implemented. These two locations are undefined after RESET. The 8-bit result of an A/D conversion is present in SFR ADCH. The result can always be calculated from the formula:

  $$256 \times \frac{V_{IN} - AV_{ref-}}{AV_{ref+} - AV_{ref-}}$$

  The A/D conversion time is 24 machine cycles instead of 50 machine cycles, and the sampling time is 6 machine cycles instead of 8 machine cycles. The conversion time takes 3 machine cycles per bit.
- The serial I/O function SIO0 and its SFRs S0BUF and S0CON are renamed to SIO, SBUF, and SCON. The interrupt related flags ES0 and PS0 are renamed ES and PS. Interrupt source S0 is renamed S. The serial I/O function remains the same.

## Differences From the 80C51

**Program Memory**

The 8XC552 contains 8k bytes of on-chip program memory which can be extended to 64k bytes with external memories (see Figure 1). When the $\overline{EA}$ pin is held high, the 8XC552 fetches instructions from internal ROM unless the address exceeds 1FFFH. Locations 2000H to FFFFH are fetched from external program memory. When the $\overline{EA}$ pin is held low, all instruction fetches are from external memory. ROM locations 0003H to 0073H are used by interrupt service routines.

**Data Memory**

The internal data memory is divided into 3 sections: the lower 128 bytes of RAM, the upper 128 bytes of RAM, and the 128-byte special function register areas. The lower 128 bytes of RAM are directly and indirectly addressable. While RAM locations 128 to 255 and the special function register area share the same address space, they are accessed through different addressing modes. RAM locations 128 to 255 are only indirectly addressable, and the special function registers are only directly addressable. All other aspects of the internal RAM are identical to the 8051.

The stack may be located anywhere in the internal RAM by loading the 8-bit stack pointer. Stack depth is 256 bytes maximum.

**Special Function Registers**

The special function registers (directly addressable only) contain all of the 8XC552 registers except the program counter and the four register banks. Most of the 56 special function registers are used to control the on-chip peripheral hardware. Other registers include arithmetic registers (ACC, B, PSW), stack pointer (SP), and data pointer registers (DHP, DPL). Sixteen of the SFRs contain 128 directly addressable bit locations. Table 1 lists the 8XC552’s special function registers.

The standard 80C51 SFRs are present and function identically in the 8XC552 except where noted in the following sections.

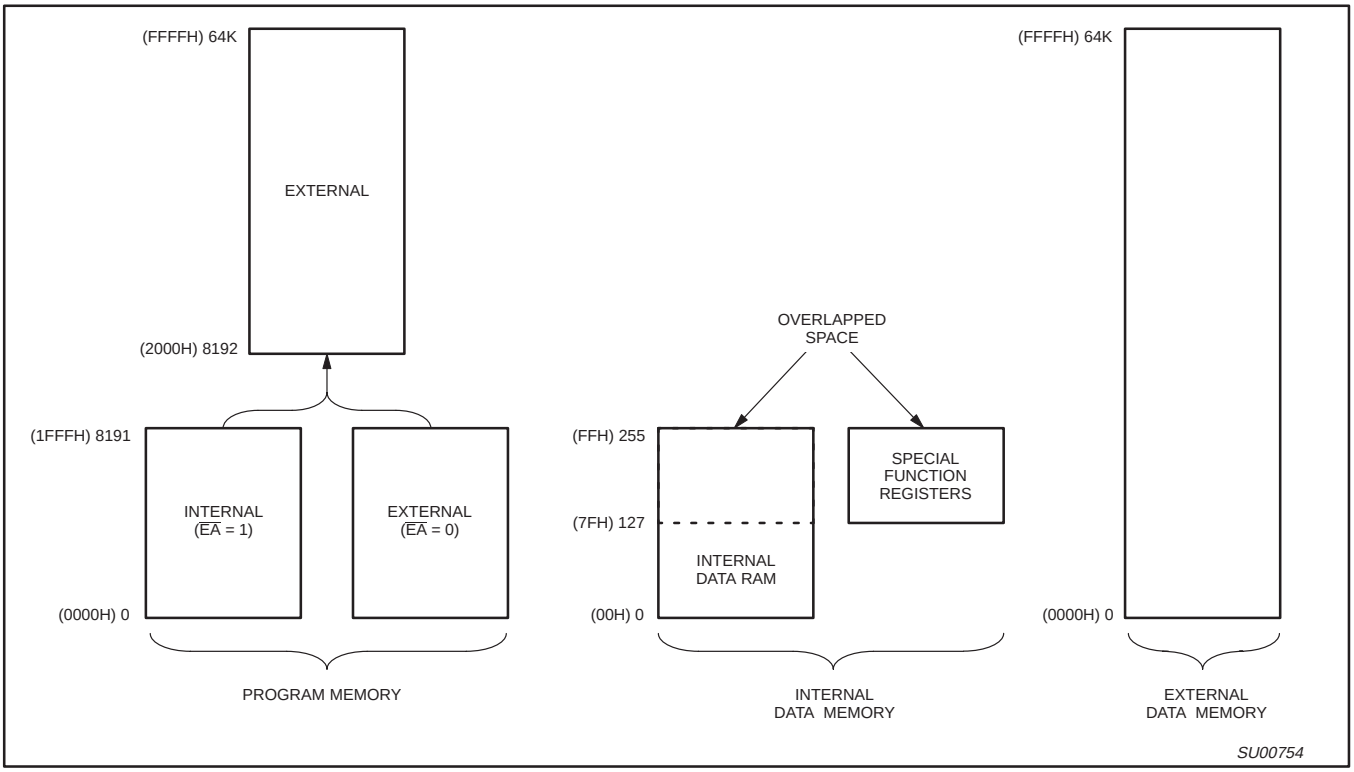



**Figure 1. Memory Map**

**Timer T2**

Timer T2 is a 16-bit timer consisting of two registers TMH2 (HIGH byte) and TML2 (LOW byte). The 16-bit timer/counter can be switched off or clocked via a prescaler from one of two sources: $f_{OSC}/12$ or an external signal. When Timer T2 is configured as a counter, the prescaler is clocked by an external signal on T2 (P1.4). A rising edge on T2 increments the prescaler, and the maximum repetition rate is one count per machine cycle (1MHz with a 12MHz oscillator).

The maximum repetition rate for Timer T2 is twice the maximum repetition rate for Timer 0 and Timer 1. T2 (P1.4) is sampled at S2P1 and again at S5P1 (i.e., twice per machine cycle). A rising edge is detected when T2 is LOW during one sample and HIGH during the next sample. To ensure that a rising edge is detected, the input signal must be LOW for at least 1/2 cycle and then HIGH for at least 1/2 cycle. If a rising edge is detected before the end of S2P1, the timer will be incremented during the following cycle; otherwise it will be incremented one cycle later. The prescaler has a programmable division factor of 1, 2, 4, or 8 and is cleared if its division factor or input source is changed, or if the timer/counter is reset.

Timer T2 may be read “on the fly” but possesses no extra read latches, and software precautions may have to be taken to avoid misinterpretation in the event of an overflow from least to most significant byte while Timer T2 is being read. Timer T2 is not loadable and is reset by the RST signal or by a rising edge on the input signal RT2, if enabled. RT2 is enabled by setting bit T2ER (TM2CON.5).

When the least significant byte of the timer overflows or when a 16-bit overflow occurs, an interrupt request may be generated. Either or both of these overflows can be programmed to request an interrupt. In both cases, the interrupt vector will be the same. When the lower byte (TML2) overflows, flag T2B0 (TM2CON) is set and flag T20V (TM2IR) is set when TMH2 overflows. These flags are set one cycle after an overflow occurs. Note that when T20V is set, T2B0 will also be set. To enable the byte overflow interrupt, bits ET2 (IEN1.7, enable overflow interrupt, see Figure 2) and T2IS0 (TM2CON.6, byte overflow interrupt select) must be set. Bit TWB0 (TM2CON.4) is the Timer T2 byte overflow flag.

To enable the 16-bit overflow interrupt, bits ET2 (IE1.7, enable overflow interrupt) and T2IS1 (TM2CON.7, 16-bit overflow interrupt select) must be set. Bit T2OV (TM2IR.7) is the Timer T2 16-bit overflow flag. All interrupt flags must be reset by software. To enable both byte and 16-bit overflow, T2IS0 and T2IS1 must be set and two interrupt service routines are required. A test on the overflow flags indicates which routine must be executed. For each routine, only the corresponding overflow flag must be cleared.

Timer T2 may be reset by a rising edge on RT2 (P1.5) if the Timer T2 external reset enable bit (T2ER) in T2CON is set. This reset also clears the prescaler. In the idle mode, the timer/counter and prescaler are reset and halted. Timer T2 is controlled by the TM2CON special function register (see Figure 3).

**Table 1. 8XC552 Special Function Registers**

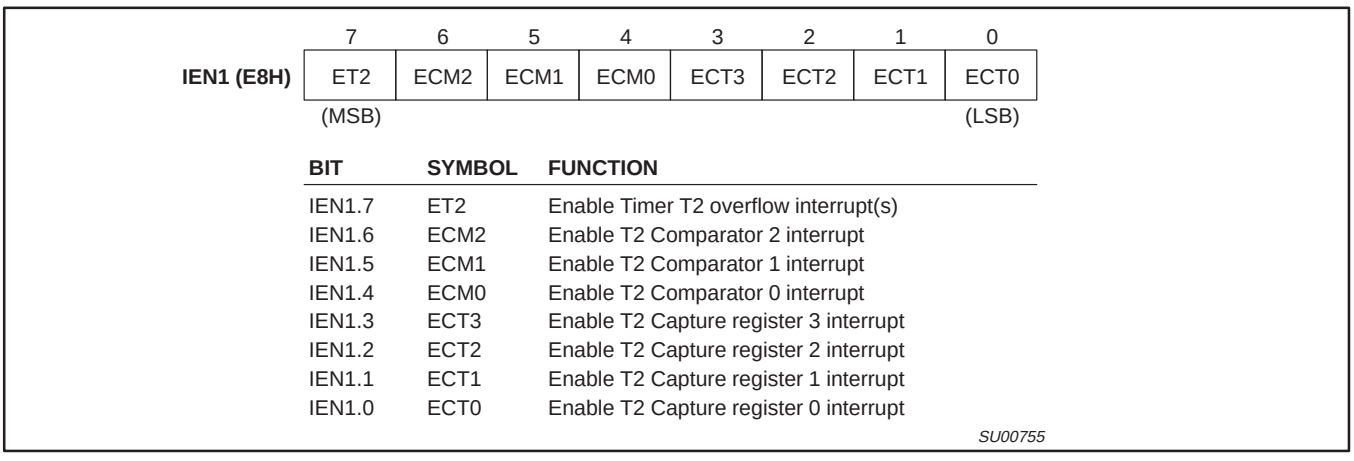
| SYMBOL | DESCRIPTION | DIRECT ADDRESS | BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB | | | | | | | LSB | RESET VALUE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ACC* | Accumulator | E0H | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 | 00H |
| ADCH# | A/D converter high | C6H | | | | | | | | | xxxxxxxxB |
| ADCON# | Adc control | C5H | ADC.1 | ADC.0 | ADEX | ADCI | ADCS | AADR2 | AADR1 | AADR0 | xx000000B |
| B* | B register | F0H | F7 | F6 | F5 | F4 | F3 | F2 | F1 | F0 | 00H |
| CTCON# | Capture control | EBH | CTN3 | CTP3 | CTN2 | CTP2 | CTN1 | CTP1 | CTN0 | CTP0 | 00H |
| CTH3# | Capture high 3 | CFH | | | | | | | | | xxxxxxxxB |
| CTH2# | Capture high 2 | CEH | | | | | | | | | xxxxxxxxB |
| CTH1# | Capture high 1 | CDH | | | | | | | | | xxxxxxxxB |
| CTH0# | Capture high 0 | CCH | | | | | | | | | xxxxxxxxB |
| CMH2# | Compare high 2 | CBH | | | | | | | | | 00H |
| CMH1# | Compare high 1 | CAH | | | | | | | | | 00H |
| CMH0# | Compare high 0 | C9H | | | | | | | | | 00H |
| CTL3# | Capture low 3 | AFH | | | | | | | | | xxxxxxxxB |
| CTL2# | Capture low 2 | AEH | | | | | | | | | xxxxxxxxB |
| CTL1# | Capture low 1 | ADH | | | | | | | | | xxxxxxxxB |
| CTL0# | Capture low 0 | ACH | | | | | | | | | xxxxxxxxB |
| CML2# | Compare low 2 | ABH | | | | | | | | | 00H |
| CML1# | Compare low 1 | AAH | | | | | | | | | 00H |
| CML0# | Compare low 0 | A9H | | | | | | | | | 00H |
| DPTR: | Data pointer (2 bytes) | | | | | | | | | | |
| DPH | Data pointer high | 83H | | | | | | | | | 00H |
| DPL | Data pointer low | 82H | | | | | | | | | 00H |
| | | | AF | AE | AD | AC | AB | AA | A9 | A8 | |
| IEN0*# | Interrupt enable 0 | A8H | EA | EAD | ES1 | ES0 | ET1 | EX1 | ET0 | EX0 | 00H |
| | | | EF | EE | ED | EC | EB | EA | E9 | E8 | |
| IEN1*# | Interrupt enable 1 | E8H | ET2 | ECM2 | ECM1 | ECM0 | ECT3 | ECT2 | ECT1 | ECT0 | 00H |
| | | | BF | BE | BD | BC | BB | BA | B9 | B8 | |
| IP0*# | Interrupt priority 0 | B8H | – | PAD | PS1 | PS0 | PT1 | PX1 | PT0 | PX0 | x0000000B |
| | | | FF | FE | FD | FC | FB | FA | F9 | F8 | |
| IP1*# | Interrupt priority 1 | F8H | PT2 | PCM2 | PCM1 | PCM0 | PCT3 | PCT2 | PCT1 | PCT0 | 00H |
| P5# | Port 5 | C4H | ADC7 | ADC6 | ADC5 | ADC4 | ADC3 | ADC2 | ADC1 | ADC0 | xxxxxxxxB |
| | | | C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 | |
| P4# | Port 4 | C0H | CMT1 | CMT0 | CMSR5 | CMSR4 | CMSR3 | CMSR2 | CMSR1 | CMSR0 | FFH |
| | | | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | |
| P3* | Port 3 | B0H | $\overline{RD}$ | $\overline{WR}$ | T1 | T0 | $\overline{INT1}$ | $\overline{INT0}$ | TXD | RXD | FFH |
| | | | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | |
| P2* | Port 2 | A0H | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | FFH |
| | | | 97 | 96 | 95 | 94 | 93 | 92 | 91 | 90 | |
| P1* | Port 1 | 90H | SDA | SCL | RT2 | T2 | CT3I | CT2I | CT1I | CT0I | FFH |
| | | | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | |
| P0* | Port 0 | 80H | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 | FFH |
| PCON# | Power control | 87H | SMOD | – | – | WLE | GF1 | GF0 | PD | IDL | 00xx0000B |
| | | | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |
| PSW* | Program status word | D0H | CY | AC | F0 | RS1 | RS0 | OV | F1 | P | 00H |

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

**Table 1. 8XC552 Special Function Registers** (Continued)

| SYMBOL | DESCRIPTION | DIRECT ADDRESS | BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB | | | | | | | LSB | RESET VALUE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PWMP# | PWM prescaler | FEH | | | | | | | | | 00H |
| PWM1# | PWM register 1 | FDH | | | | | | | | | 00H |
| PWM0# | PWM register 0 | FCH | | | | | | | | | 00H |
| RTE# | Reset/toggle enable | EFH | TP47 | TP46 | RP45 | RP44 | RP43 | RP42 | RP41 | RP40 | 00H |
| SP | Stack pointer | 81H | | | | | | | | | 07H |
| S0BUF | Serial 0 data buffer | 99H | | | | | | | | | xxxxxxxxB |
| | | | 9F | 9E | 9D | 9C | 9B | 9A | 99 | 98 | |
| S0CON* | Serial 0 control | 98H | SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI | 00H |
| S1ADR# | Serial 1 address | DBH | ——————— SLAVE ADDRESS ——————— | | | | | | | GC | 00H |
| SIDAT# | Serial 1 data | DAH | | | | | | | | | 00H |
| S1STA# | Serial 1 status | D9H | SC4 | SC3 | SC2 | SC1 | SC0 | 0 | 0 | 0 | F8H |
| | | | DF | DE | DD | DC | DB | DA | D9 | D8 | |
| SICON#* | Serial 1 control | D8H | CR2 | ENS1 | STA | ST0 | SI | AA | CR1 | CR0 | 00H |
| STE# | Set enable | EEH | TG47 | TG46 | SP45 | SP44 | SP43 | SP42 | SP41 | SP40 | C0H |
| TH1 | Timer high 1 | 8DH | | | | | | | | | 00H |
| TH0 | Timer high 0 | 8CH | | | | | | | | | 00H |
| TL1 | Timer low 1 | 8BH | | | | | | | | | 00H |
| TL0 | Timer low 0 | 8AH | | | | | | | | | 00H |
| TMH2# | Timer high 2 | EDH | | | | | | | | | 00H |
| TML2# | Timer low 2 | ECH | | | | | | | | | 00H |
| TMOD | Timer mode | 89H | GATE | C/$\overline{T}$ | M1 | M0 | GATE | C/$\overline{T}$ | M1 | M0 | 00H |
| | | | 8F | 8E | 8D | 8C | 8B | 8A | 89 | 88 | |
| TCON* | Timer control | 88H | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 | 00H |
| TM2CON# | Timer 2 control | EAH | T2IS1 | T2IS0 | T2ER | T2B0 | T2P1 | T2P0 | T2MS1 | T2MS0 | 00H |
| | | | CF | CE | CD | CC | CB | CA | C9 | C8 | |
| TM2IR#* | Timer 2 int flag reg | C8H | T20V | CMI2 | CMI1 | CMI0 | CTI3 | CTI2 | CTI1 | CTI0 | 00H |
| T3# | Timer 3 | FFH | | | | | | | | | 00H |

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.




| BIT | SYMBOL | FUNCTION |
|---|---|---|
| IEN1.7 | ET2 | Enable Timer T2 overflow interrupt(s) |
| IEN1.6 | ECM2 | Enable T2 Comparator 2 interrupt |
| IEN1.5 | ECM1 | Enable T2 Comparator 1 interrupt |
| IEN1.4 | ECM0 | Enable T2 Comparator 0 interrupt |
| IEN1.3 | ECT3 | Enable T2 Capture register 3 interrupt |
| IEN1.2 | ECT2 | Enable T2 Capture register 2 interrupt |
| IEN1.1 | ECT1 | Enable T2 Capture register 1 interrupt |
| IEN1.0 | ECT0 | Enable T2 Capture register 0 interrupt |

*SU00755*

**Figure 2. Timer T2 Interrupt Enable Register (IEN1)**

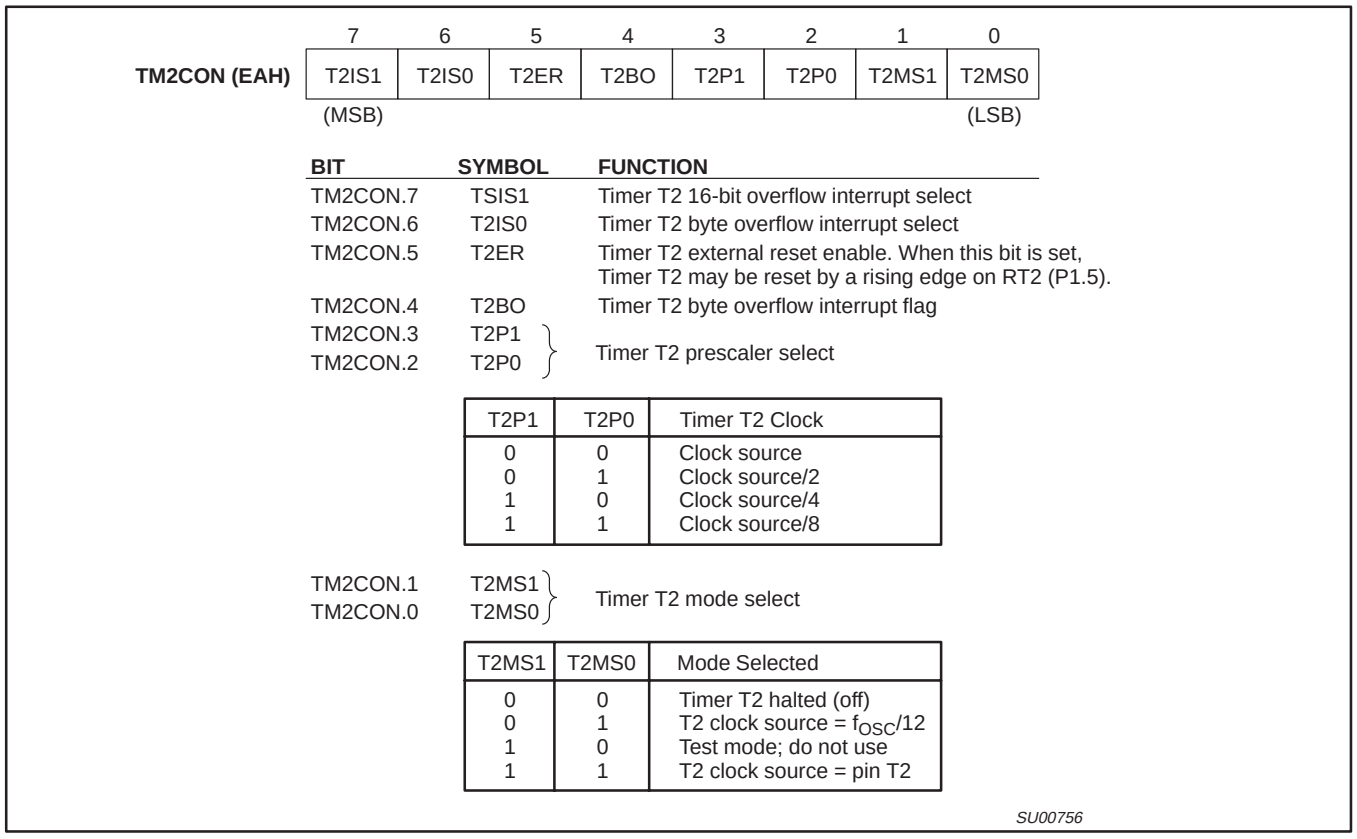



**Figure 3. T2 Control Register (TM2CON)**

**Timer T2 Extension:** When a 12MHz oscillator is used, a 16-bit overflow on Timer T2 occurs every 65.5, 131, 262, or 524 ms, depending on the prescaler division ratio; i.e., the maximum cycle time is approximately 0.5 seconds. In applications where cycle times are greater than 0.5 seconds, it is necessary to extend Timer T2. This is achieved by selecting fosc/12 as the clock source (set T2MS0, reset T2MS1), setting the prescaler division ration to 1/8 (set T2P0, set T2P1), disabling the byte overflow interrupt (reset T2IS0) and enabling the 16-bit overflow interrupt (set T2IS1). The following software routine is written for a three-byte extension which gives a maximum cycle time of approximately 2400 hours.

```
OVINT:  PUSH    ACC     ;save accumulator
        PUSH    PSW     ;save status
        INC     TIMEX1  ;increment first byte (low order)
                        ;of extended timer
        MOV     A,TIMEX1
        JNZ     INTEX   ;jump to INTEX if ;there is no overflow

        INC     TIMEX2  ;increment second byte
        MOV     A,TIMEX2
        JNZ     INTEX   ;jump to INTEX if there is no overflow
        INC     TIMEX3  ;increment third byte (high order)

INTEX:  CLR     T2OV    ;reset interrupt flag
        POP     PSW     ;restore status
        POP     ACC     ;restore accumulator
        RETI            ;return from interrupt
```

**Timer T2, Capture and Compare Logic:** Timer T2 is connected to four 16-bit capture registers and three 16-bit compare registers. A capture register may be used to capture the contents of Timer T2 when a transition occurs on its corresponding input pin. A compare register may be used to set, reset, or toggle port 4 output pins at certain pre-programmable time intervals.

The combination of Timer T2 and the capture and compare logic is very powerful in applications involving rotating machinery, automotive injection systems, etc. Timer T2 and the capture and compare logic are shown in Figure 4.

**Capture Logic:** The four 16-bit capture registers that Timer T2 is connected to are: CT0, CT1, CT2, and CT3. These registers are loaded with the contents of Timer T2, and an interrupt is requested upon receipt of the input signals CT0I, CT1I, CT2I, or CT3I. These input signals are shared with port 1. The four interrupt flags are in the Timer T2 interrupt register (TM2IR special function register). If the capture facility is not required, these inputs can be regarded as additional external interrupt inputs.

Using the capture control register CTCON (see Figure 5), these inputs may capture on a rising edge, a falling edge, or on either a rising or falling edge. The inputs are sampled during S1P1 of each cycle. When a selected edge is detected, the contents of Timer T2 are captured at the end of the cycle.

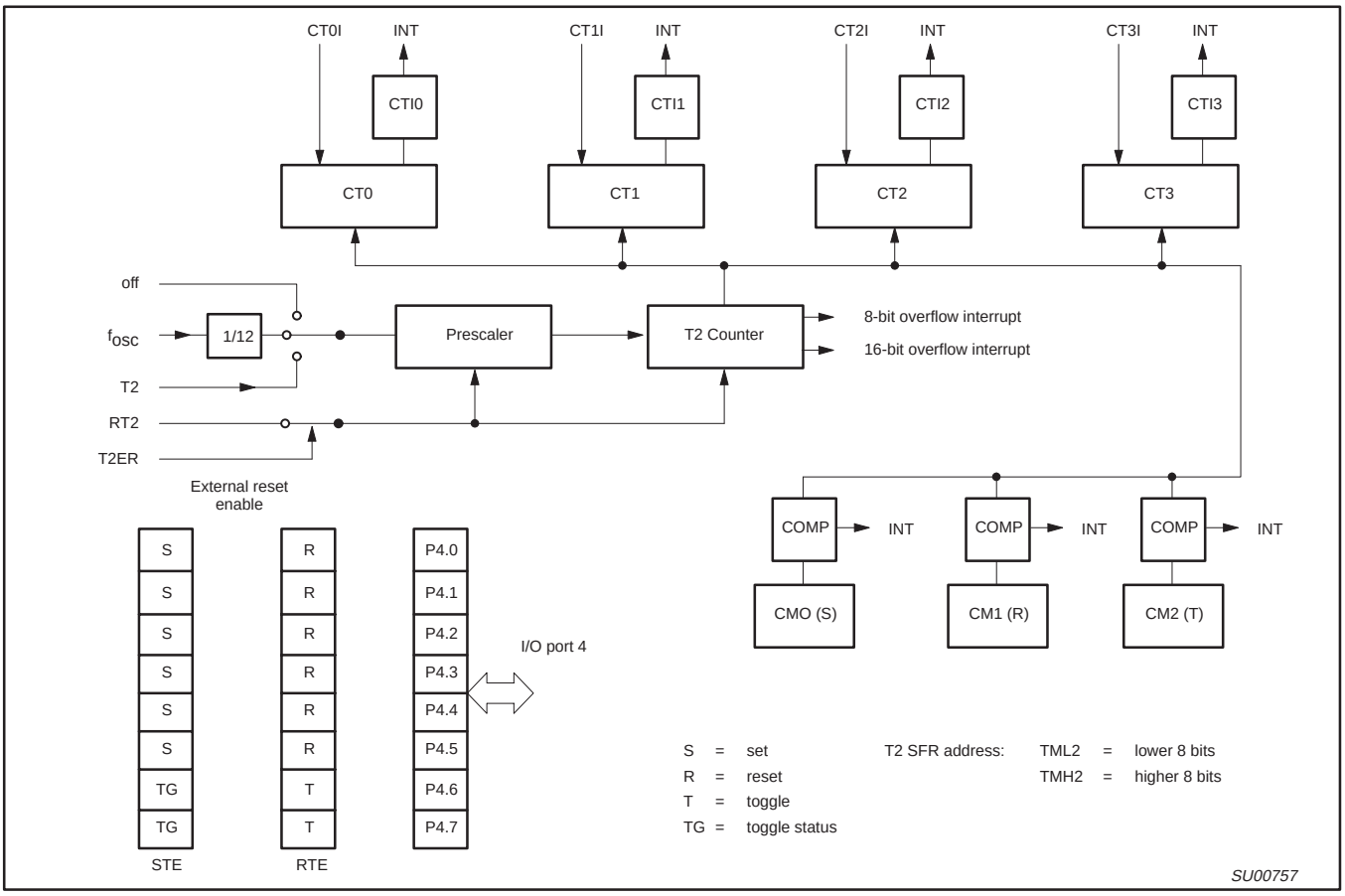



**Figure 4. Block Diagram of Timer 2**

**Measuring Time Intervals Using Capture Registers:** When a recurring external event is represented in the form of rising or falling edges on one of the four capture pins, the time between two events can be measured using Timer T2 and a capture register. When an event occurs, the contents of Timer T2 are copied into the relevant capture register and an interrupt request is generated. The interrupt service routine may then compute the interval time if it knows the previous contents of Timer T2 when the last event occurred. With a 12MHz oscillator, Timer T2 can be programmed to overflow every 524ms. When event interval times are shorter than this, computing the interval time is simple, and the interrupt service routine is short. For longer interval times, the Timer T2 extension routine may be used.

**Compare Logic:** Each time Timer T2 is incremented, the contents of the three 16-bit compare registers CM0, CM1, and CM2 are compared with the new counter value of Timer T2. When a match is found, the corresponding interrupt flag in TM2IR is set at the end of the following cycle. When a match with CM0 occurs, the controller sets bits 0-5 of port 4 if the corresponding bits of the set enable register STE are at logic 1.

When a match with CM1 occurs, the controller resets bits 0-5 of port 4 if the corresponding bits of the reset/toggle enable register RTE are at logic 1 (see Figure 6 for RTE register function). If RTE is “0”, then P4.n is not affected by a match between CM1 or CM2 and Timer 2. When a match with CM2 occurs, the controller “toggles” bits 6 and 7 of port 4 if the corresponding bits of the RTE are at logic 1. The port latches of bits 6 and 7 are not toggled.

Two additional flip-flops store the last operation, and it is these flip-flops that are toggled.

Thus, if the current operation is “set,” the next operation will be “reset” even if the port latch is reset by software before the “reset” operation occurs. The first “toggle” after a chip RESET will set the port latch. The contents of these two flip-flops can be read at STE.6 and STE.7 (corresponding to P4.6 and P4.7, respectively). Bits STE.6 and STE.7 are read only (see Figure 7 for STE register function). A logic 1 indicates that the next toggle will set the port latch; a logic 0 indicates that the next toggle will reset the port latch. CM0, CM1, and CM2 are reset by the RST signal.

The modified port latch information appears at the port pin during S5P1 of the cycle following the cycle in which a match occurred. If the port is modified by software, the outputs change during S1P1 of the following cycle. Each port 4 bit can be set or reset by software at any time. A hardware modification resulting from a comparator match takes precedence over a software modification in the same cycle. When the comparator results require a “set” and a “reset” at the same time, the port latch will be reset.

**Timer T2 Interrupt Flag Register TM2IR:** Eight of the nine Timer T2 interrupt flags are located in special function register TM2IR (see Figure 8). The ninth flag is TM2CON.4.

The CT0I and CT1I flags are set during S4 of the cycle in which the contents of Timer T2 are captured. CT0I is scanned by the interrupt logic during S2, and CT1I is scanned during S3. CT2I and CT3I are set during S6 and are scanned during S4 and S5. The associated

interrupt requests are recognized during the following cycle. If these flags are polled, a transition at CT0I or CT1I will be recognized one cycle before a transition on CT2I or CT3I since registers are read during S5. The CMI0, CMI1, and CMI2 flags are set during S6 of the cycle following a match. CMI0 is scanned by the interrupt logic during S2; CMI1 and CMI2 are scanned during S3 and S4. A match will be recognized by the interrupt logic (or by polling the flags) two cycles after the match takes place.

The 16-bit overflow flag (T2OV) and the byte overflow flag (T2BO) are set during S6 of the cycle in which the overflow occurs. These flags are recognized by the interrupt logic during the next cycle.

Special function register IP1 (Figure 8) is used to determine the Timer T2 interrupt priority. Setting a bit high gives that function a high priority, and setting a bit low gives the function a low priority. The functions controlled by the various bits of the IP1 register are shown in Figure 8.

**CTCON (EBH)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CTN3 | CTP3 | CTN2 | CTP2 | CTN1 | CTP1 | CTN1 | CTP0 |
| (MSB) | | | | | | | (LSB) |

| BIT | SYMBOL | CAPTURE/INTERRUPT ON: |
|---|---|---|
| CTCON.7 | CTN3 | Capture Register 3 triggered by a falling edge on CT3I |
| CTCON.6 | CTP3 | Capture Register 3 triggered by a rising edge on CT3I |
| CTCON.5 | CTN2 | Capture Register 2 triggered by a falling edge on CT2I |
| CTCON.4 | CTP2 | Capture Register 2 triggered by a rising edge on CT2I |
| CTCON.3 | CTN1 | Capture Register 1 triggered by a falling edge on CT1I |
| CTCON.2 | CTP1 | Capture Register 1 triggered by a rising edge on CT1I |
| CTCON.1 | CTN0 | Capture Register 0 triggered by a falling edge on CT0I |
| CTCON.0 | CTP0 | Capture Register 0 triggered by a rising edge on CT0I |

SU00758

**Figure 5. Capture Control Register (CTCON)**

**RTE (EFH)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TP47 | TP46 | RP45 | RP44 | RP43 | RP42 | RO41 | RP40 |
| (MSB) | | | | | | | (LSB) |

| BIT | SYMBOL | FUNCTION |
|---|---|---|
| RTE.7 | TP47 | If “1” then P4.7 toggles on a match between CM1 and Timer T2 |
| RTE.6 | TP46 | If “1” then P4.6 toggles on a match between CM1 and Timer T2 |
| RTE.5 | RP45 | If “1” then P4.5 is reset on a match between CM1 and Timer T2 |
| RTE.4 | RP44 | If “1” then P4.4 is reset on a match between CM1 and Timer T2 |
| RTE.3 | RP43 | If “1” then P4.3 is reset on a match between CM1 and Timer T2 |
| RTE.2 | RP42 | If “1” then P4.2 is reset on a match between CM1 and Timer T2 |
| RTE.1 | RP41 | If “1” then P4.1 is reset on a match between CM1 and Timer T2 |
| RTE.0 | RP40 | If “1” then P4.0 is reset on a match between CM1 and Timer T2 |

SU00759

**Figure 6. Reset/Toggle Enable Register (RTE)**

**STE (EEH)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TG47 | TG46 | SP45 | SP44 | SP43 | SP42 | SP41 | SP40 |
| (MSB) | | | | | | | (LSB) |

| BIT | SYMBOL | FUNCTION |
|---|---|---|
| STE.7 | TG47 | Toggle flip-flops |
| STE.6 | TG46 | Toggle flip-flops |
| STE.5 | SP45 | If “1” then P4.5 is set on a match between CM0 and Timer T2 |
| STE.4 | SP44 | If “1” then P4.4 is set on a match between CM0 and Timer T2 |
| STE.3 | SP43 | If “1” then P4.3 is set on a match between CM0 and Timer T2 |
| STE.2 | SP42 | If “1” then P4.2 is set on a match between CM0 and Timer T2 |
| STE.1 | SP41 | If “1” then P4.1 is set on a match between CM0 and Timer T2 |
| STE.0 | SP40 | If “1” then P4.0 is set on a match between CM0 and Timer T2 |

SU00760

**Figure 7. Set Enable Register (STE)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **TM2IR (C8H)** | T2OV | CMI2 | CMI1 | CMI0 | CTI3 | CTI2 | CTI1 | CTI0 |
| | (MSB) | | | | | | | (LSB) |

| BIT | SYMBOL | FUNCTION |
|---|---|---|
| TM2IR.7 | T2OV | Timer T2 16-bit overflow interrupt flag |
| TM2IR.6 | CMI2 | CM2 interrupt flag |
| TM2IR.5 | CMI1 | CM1 interrupt flag |
| TM2IR.4 | CMI0 | CM0 interrupt flag |
| TM2IR.3 | CTI3 | CT3 interrupt flag |
| TM2IR.2 | CTI2 | CT2 interrupt flag |
| TM2IR.1 | CTI1 | CT1 interrupt flag |
| TM2IR.0 | CTI0 | CT0 interrupt flag |

**Interrupt Flag Register (TM2IR)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **IP1 (F8H)** | PT2 | PCM2 | PCM1 | PCM0 | PCT3 | PCT2 | PCT1 | PCT0 |
| | (MSB) | | | | | | | (LSB) |

| BIT | SYMBOL | FUNCTION |
|---|---|---|
| IP1.7 | PT2 | Timer T2 overflow interrupt(s) priority level |
| IP1.6 | PCM2 | Timer T2 comparator 2 interrupt priority level |
| IP1.5 | PCM1 | Timer T2 comparator 1 interrupt priority level |
| IP1.4 | PCM0 | Timer T2 comparator 0 interrupt priority level |
| IP1.3 | PCT3 | Timer T2 capture register 3 interrupt priority level |
| IP1.2 | PCT2 | Timer T2 capture register 2 interrupt priority level |
| IP1.1 | PCT1 | Timer T2 capture register 1 interrupt priority level |
| IP1.0 | PCT0 | Timer T2 capture register 0 interrupt priority level |

**Timer 2 Interrupt Priority Register (IP1)**

SU00761

**Figure 8. Interrupt Flag Register (TM2IR) and Timer T2 Interrupt Priority Register (IP1)**

**Timer T3, The Watchdog Timer**

In addition to Timer T2 and the standard timers, a watchdog timer is also incorporated on the 8XC552. The purpose of a watchdog timer is to reset the microcontroller if it enters erroneous processor states (possibly caused by electrical noise or RFI) within a reasonable period of time. An analogy is the “dead man’s handle” in railway locomotives. When enabled, the watchdog circuitry will generate a system reset if the user program fails to reload the watchdog timer within a specified length of time known as the “watchdog interval.”

**Watchdog Circuit Description:** The watchdog timer (Timer T3) consists of an 8-bit timer with an 11-bit prescaler as shown in Figure 9. The prescaler is fed with a signal whose frequency is 1/12 the oscillator frequency (1MHz with a 12MHz oscillator). The 8-bit timer is incremented every “t” seconds, where:

$t = 12 \times 2048 \times 1/f_{OSC}$
(= 1.5ms at $f_{OSC}$ = 16MHz; = 1ms at $f_{OSC}$ = 24MHz)

If the 8-bit timer overflows, a short internal reset pulse is generated which will reset the 8XC552. A short output reset pulse is also generated at the RST pin. This short output pulse (3 machine cycles) may be destroyed if the RST pin is connected to a capacitor. This would not, however, affect the internal reset operation.

Watchdog operation is activated when external pin $\overline{EW}$ is tied low. When $\overline{EW}$ is tied low, it is impossible to disable the watchdog operation by software.

**How to Operate the Watchdog Timer:** The watchdog timer has to be reloaded within periods that are shorter than the programmed watchdog interval; otherwise the watchdog timer will overflow and a system reset will be generated. The user program must therefore continually execute sections of code which reload the watchdog timer. The period of time elapsed between execution of these sections of code must never exceed the watchdog interval. When using a 16MHz oscillator, the watchdog interval is programmable between 1.5ms and 392ms. When using a 24MHz oscillator, the watchdog interval is programmable between 1ms and 255ms.

In order to prepare software for watchdog operation, a programmer should first determine how long his system can sustain an erroneous processor state. The result will be the maximum watchdog interval. As the maximum watchdog interval becomes shorter, it becomes more difficult for the programmer to ensure that the user program always reloads the watchdog timer within the watchdog interval, and thus it becomes more difficult to implement watchdog operation.

The programmer must now partition the software in such a way that reloading of the watchdog is carried out in accordance with the above requirements. The programmer must determine the execution times of all software modules. The effect of possible conditional branches, subroutines, external and internal interrupts must all be taken into account. Since it may be very difficult to evaluate the execution times of some sections of code, the programmer should use worst case estimations. In any event, the programmer must make sure that the watchdog is not activated during normal operation.

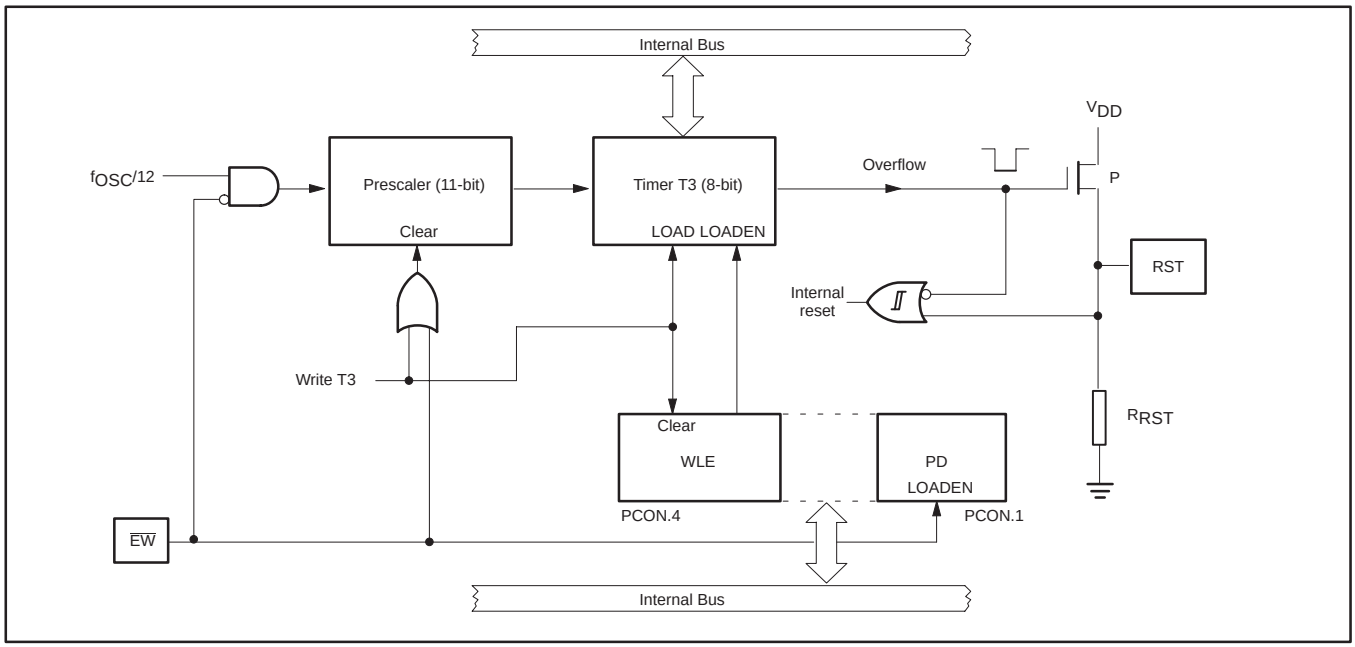



**Figure 9. Watchdog Timer**

The watchdog timer is reloaded in two stages in order to prevent erroneous software from reloading the watchdog. First PCON.4 (WLE) must be set. The T3 may be loaded. When T3 is loaded, PCON.4 (WLE) is automatically reset. T3 cannot be loaded if PCON.4 (WLE) is reset. Reload code may be put in a subroutine as it is called frequently. Since Timer T3 is an up-counter, a reload value of 00H gives the maximum watchdog interval (510ms with a 12MHz oscillator), and a reload value of 0FFH gives the minimum watchdog interval (2ms with a 12MHz oscillator).

In the idle mode, the watchdog circuitry remains active. When watchdog operation is implemented, the power-down mode cannot be used since both states are contradictory. Thus, when watchdog operation is enabled by tying external pin $\overline{EW}$ low, it is impossible to enter the power-down mode, and an attempt to set the power-down bit (PCON.1) will have no effect. PCON.1 will remain at logic 0.

During the early stages of software development/debugging, the watchdog may be disabled by tying the $\overline{EW}$ pin high. At a later stage, $\overline{EW}$ may be tied low to complete the debugging process.

**Watchdog Software Example:** The following example shows how watchdog operation might be handled in a user program.

```
;at the program start:

T3              EQU  0FFH ;address of watchdog timer T3
PCON            EQU  087H ;address of PCON SFR
WATCH-INTV      EQU  156  ;watchdog interval (e.g., 2x100ms)

;to be inserted at each watchdog reload location within
;the user program:

LCALL WATCHDOG

;watchdog service routine:

WATCHDOG:  ORL   PCON,#10H  ;set condition flag (PCON.4)
           MOV T3,WATCH-INV      ;load T3 with  watchdog interval
           RET
```

If it is possible for this subroutine to be called in an erroneous state, then the condition flag WLE should be set at different parts of the main program.

**Serial I/O**

The 8XC552 is equipped with two independent serial ports: SIO0 and SIO1. SIO0 is a full duplex UART port and is identical to the 80C51 serial port. SIO1 accommodates the $I^2C$ bus.

**SIO0:** SIO0 is a full duplex serial I/O port identical to that on the 80C51. Its operation is the same, including the use of timer 1 as a baud rate generator.

**SIO1, $I^2C$ Serial I/O:** The $I^2C$ bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the bus are:

- Bidirectional data transfer between masters and slaves
- Multimaster bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
- The $I^2C$ bus may be used for test and diagnostic purposes

The output latches of P1.6 and P1.7 must be set to logic 1 in order to enable SIO1.

The 8XC552 on-chip $I^2C$ logic provides a serial interface that meets the $I^2C$ bus specification and supports all transfer modes (other than the low-speed mode) from and to the $I^2C$ bus. The SIO1 logic handles bytes transfer autonomously. It also keeps track of serial transfers, and a status register (S1STA) reflects the status of SIO1 and the $I^2C$ bus.

The CPU interfaces to the $I^2C$ logic via the following four special function registers: S1CON (SIO1 control register), S1STA (SIO1 status register), S1DAT (SIO1 data register), and S1ADR (SIO1 slave address register). The SIO1 logic interfaces to the external $I^2C$ bus via two port 1 pins: P1.6/SCL (serial clock line) and P1.7/SDA (serial data line).

A typical $I^2C$ bus configuration is shown in Figure 10, and Figure 11 shows how a data transfer is accomplished on the bus. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the $I^2C$ bus:

1. Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte.
2. Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “not acknowledge” is returned.

The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the $I^2C$ bus will not be released.

**Modes of Operation:** The on-chip SIO1 logic may operate in the following four modes:

1. Master Transmitter Mode:

   Serial data output through P1.7/SDA while P1.6/SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this case the data direction bit (R/$\overline{W}$) will be logic 0, and we say that a “W” is transmitted. Thus the first byte transmitted is SLA+W. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

2. Master Receiver Mode:

   The first byte transmitted contains the slave address of the transmitting device (7 bits) and the data direction bit. In this case the data direction bit (R/$\overline{W}$) will be logic 1, and we say that an “R” is transmitted. Thus the first byte transmitted is SLA+R. Serial data is received via P1.7/SDA while P1.6/SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are output to indicate the beginning and end of a serial transfer.

3. Slave Receiver Mode:

   Serial data and the serial clock are received through P1.7/SDA and P1.6/SCL. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit.

4. Slave Transmitter Mode:

   The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted via P1.7/SDA while the serial clock is input through P1.6/SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer.

In a given application, SIO1 may operate as a master and as a slave. In the slave mode, the SIO1 hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. When the microcontroller wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, SIO1 switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

**SIO1 Implementation and Operation:** Figure 12 shows how the on-chip $I^2C$ bus interface is implemented, and the following text describes the individual blocks.

INPUT FILTERS AND OUTPUT STAGES

The input filters have $I^2C$ compatible input levels. If the input voltage is less than 1.5V, the input logic level is interpreted as 0; if the input voltage is greater than 3.0V, the input logic level is interpreted as 1. Input signals are synchronized with the internal clock ($f_{OSC}/4$), and spikes shorter than three oscillator periods are filtered out.

The output stages consist of open drain transistors that can sink 3mA at $V_{OUT} < 0.4V$. These open drain outputs do not have clamping diodes to $V_{DD}$. Thus, if the device is connected to the $I^2C$ bus and $V_{DD}$ is switched off, the $I^2C$ bus is not affected.

ADDRESS REGISTER, S1ADR

This 8-bit special function register may be loaded with the 7-bit slave address (7 most significant bits) to which SIO1 will respond when programmed as a slave transmitter or receiver. The LSB (GC) is used to enable general call address (00H) recognition.

COMPARATOR

The comparator compares the received 7-bit slave address with its own slave address (7 most significant bits in S1ADR). It also compares the first received 8-bit byte with the general call address (00H). If an equality is found, the appropriate status bits are set and an interrupt is requested.

SHIFT REGISTER, S1DAT

This 8-bit special function register contains a byte of serial data to be transmitted or a byte which has just been received. Data in S1DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of S1DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; S1DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in S1DAT.

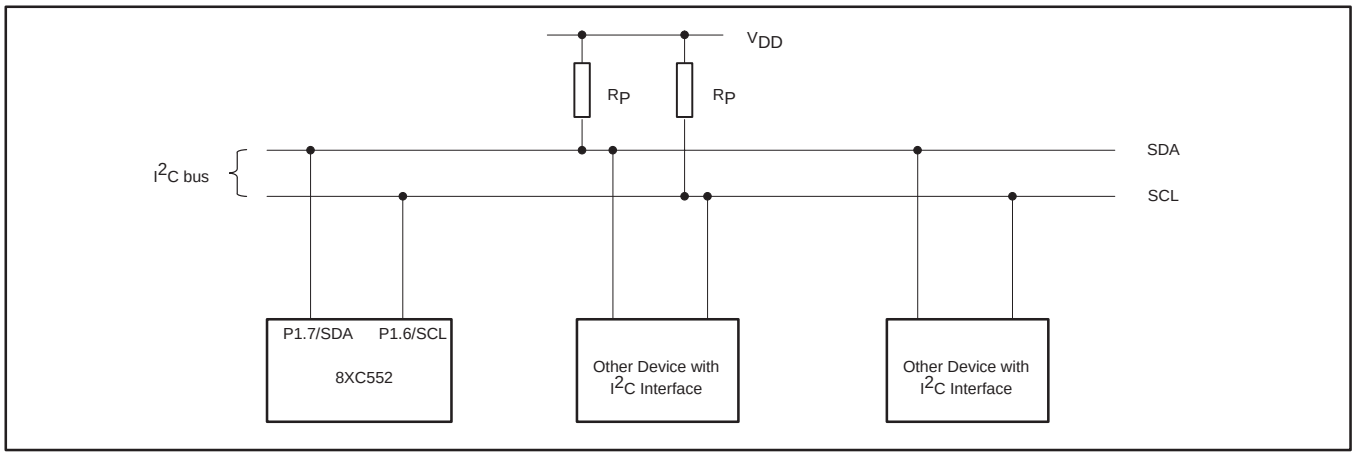



**Figure 10. Typical I²C Bus Configuration**

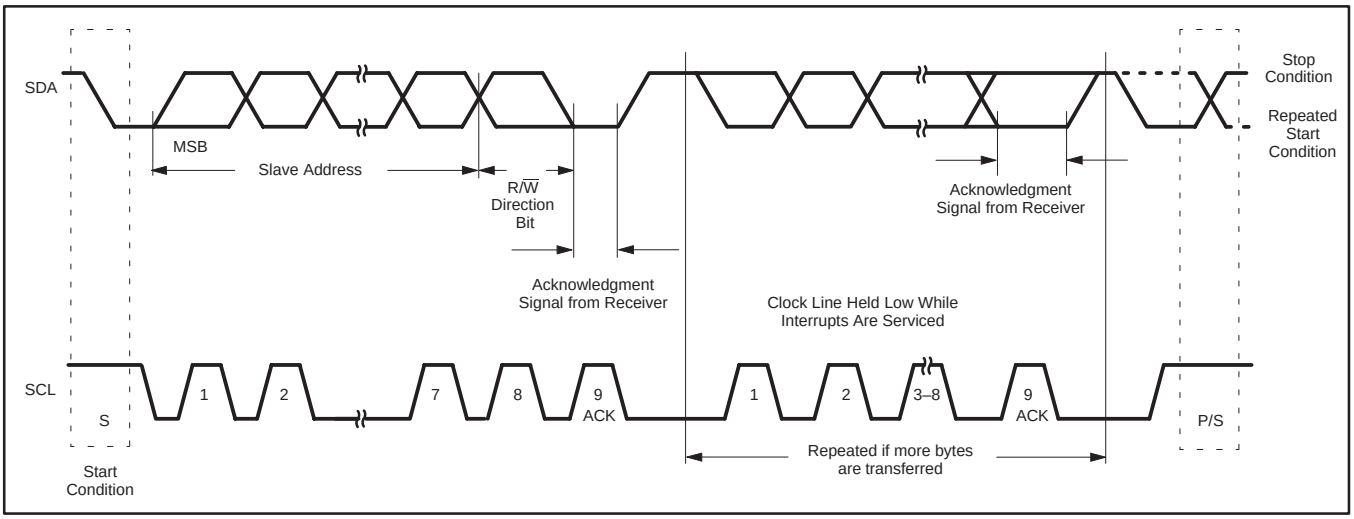



**Figure 11. Data Transfer on the I²C Bus**

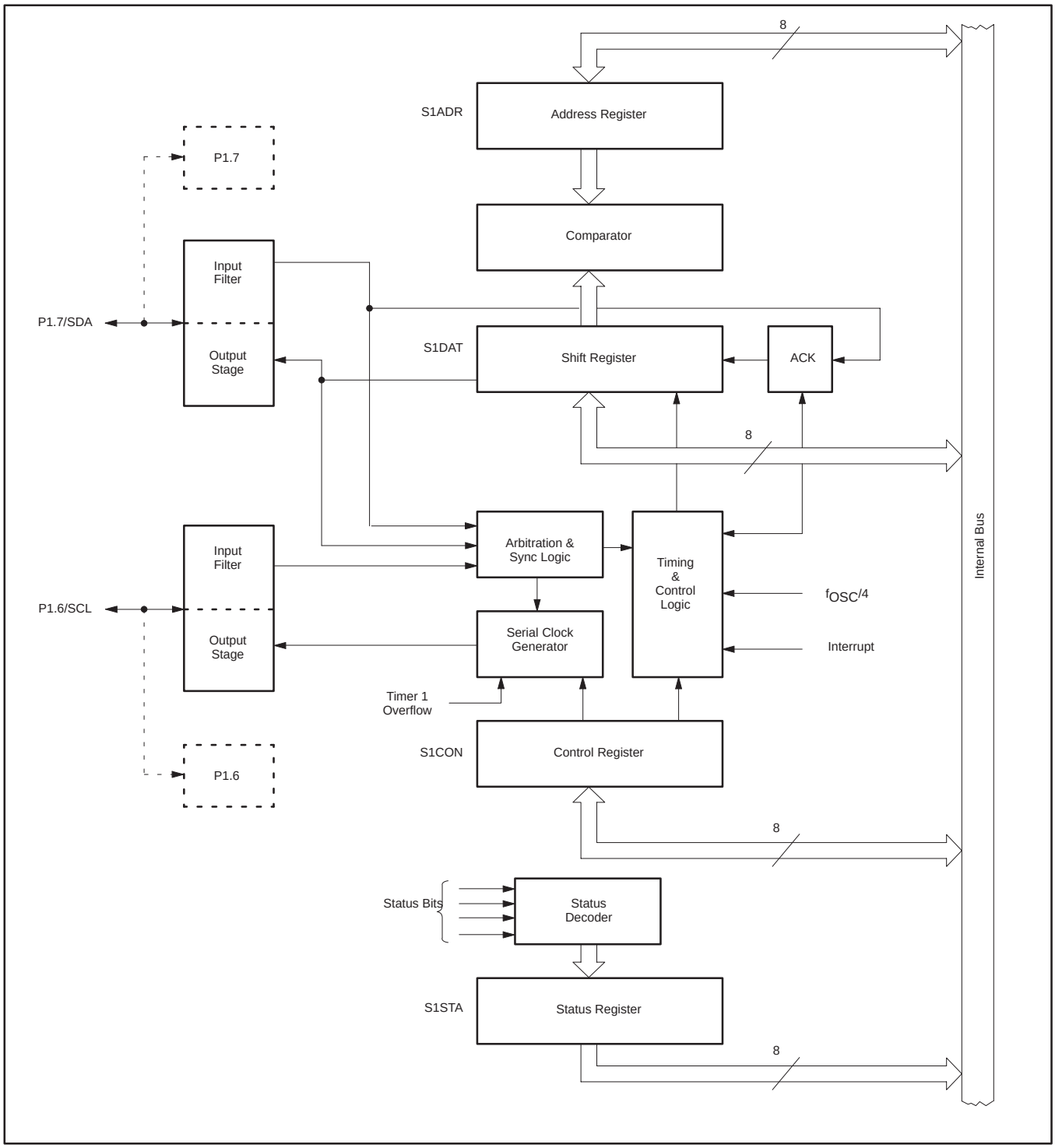
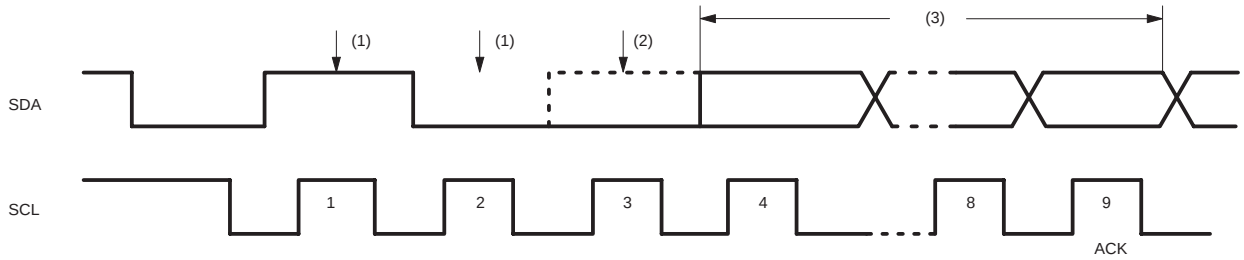
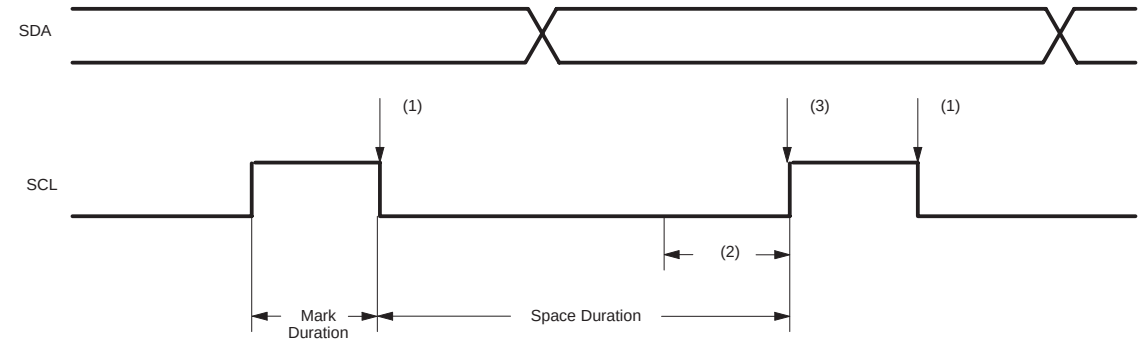



**Figure 12. I²C Bus Serial Interface Block Diagram**




1. Another device transmits identical serial data.
2. Another device overrules a logic 1 (dotted line) transmitted by SIO1 (master) by pulling the SDA line low. Arbitration is lost, and SIO1 enters the slave receiver mode.
3. SIO1 is in the slave receiver mode but still generates clock pulses until the current byte has been transmitted. SIO1 will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

**Figure 13. Arbitration Procedure**




1. Another service pulls the SCL line low before the SIO1 “mark” duration is complete. The serial clock generator is immediately reset and commences with the “space” duration by pulling SCL low.
2. Another device still pulls the SCL line low after SIO1 releases SCL. The serial clock generator is forced into the wait state until the SCL line is released.
3. The SCL line is released, and the serial clock generator commences with the mark duration.

**Figure 14. Serial Clock Synchronization**

### Arbitration and Synchronization Logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic 1 actually appears as a logic 1 on the $I^2C$ bus. If another device on the bus overrules a logic 1 and pulls the SDA line low, arbitration is lost, and SIO1 immediately changes from master transmitter to slave receiver. SIO1 will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete.

Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while SIO1 is returning a “not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal LOW. Since this can occur only at the end of a serial byte, SIO1 generates no further clock pulses. Figure 13 shows the arbitration procedure.

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the “mark” duration is determined by the device that generates the shortest “marks,” and the “space” duration is determined by the device that generates the longest “spaces.” Figure 14 shows the synchronization procedure.

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. SIO1 will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. The serial interrupt flag (SI) is set, and the stretching continues until the serial interrupt flag is cleared.

### Serial Clock Generator

This programmable clock pulse generator provides the SCL clock pulses when SIO1 is in the master transmitter or master receiver mode. It is switched off when SIO1 is in a slave mode. The programmable output clock frequencies are: $f_{OSC}/120$, $f_{OSC}/9600$, and the Timer 1 overflow rate divided by eight. The output clock

pulses have a 50% duty cycle unless the clock generator is synchronized with other SCL clock sources as described above.

TIMING AND CONTROL

The timing and control logic generates the timing and control signals for serial byte handling. This logic block provides the shift pulses for S1DAT, enables the comparator, generates and detects start and stop conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic, and monitors the $I^2C$ bus status.

CONTROL REGISTER, S1CON

This 7-bit special function register is used by the microcontroller to control the following SIO1 functions: start and restart of a serial transfer, termination of a serial transfer, bit rate, address recognition, and acknowledgment.

STATUS DECODER AND STATUS REGISTER

The status decoder takes all of the internal status bits and compresses them into a 5-bit code. This code is unique for each $I^2C$ bus status. The 5-bit code may be used to generate vector addresses for fast processing of the various service routines. Each service routine processes a particular bus status. There are 26 possible bus states if all four modes of SIO1 are used. The 5-bit status code is latched into the five most significant bits of the status register when the serial interrupt flag is set (by hardware) and remains stable until the interrupt flag is cleared by software. The three least significant bits of the status register are always zero. If the status code is used as a vector to service routines, then the routines are displaced by eight address locations. Eight bytes of code is sufficient for most of the service routines (see the software example in this section).

**The Four SIO1 Special Function Registers:** The microcontroller interfaces to SIO1 via four special function registers. These four SFRs (S1ADR, S1DAT, S1CON, and S1STA) are described individually in the following sections.

**The Address Register, S1ADR:** The CPU can read from and write to this 8-bit, directly addressable SFR. S1ADR is not affected by the SIO1 hardware. The contents of this register are irrelevant when SIO1 is in a master mode. In the slave modes, the seven most significant bits must be loaded with the microcontroller's own slave address, and, if the least significant bit is set, the general call address (00H) is recognized; otherwise it is ignored.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| S1ADR (DBH) | X | X | X | X | X | X | X | GC |

own slave address

The most significant bit corresponds to the first bit received from the $I^2C$ bus after a start condition. A logic 1 in S1ADR corresponds to a high level on the $I^2C$ bus, and a logic 0 corresponds to a low level on the bus.

**The Data Register, S1DAT:** S1DAT contains a byte of serial data to be transmitted or a byte which has just been received. The CPU can read from and write to this 8-bit, directly addressable SFR while it is not in the process of shifting a byte. This occurs when SIO1 is in a defined state and the serial interrupt flag is set. Data in S1DAT remains stable as long as SI is set. Data in S1DAT is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and, after a byte has been received, the first bit of received data is located at the MSB of S1DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; S1DAT always contains the last data byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in S1DAT.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| S1DAT (DAH) | SD7 | SD6 | SD5 | SD4 | SD3 | SD2 | SD1 | SD0 |

shift direction

SD7 - SD0:

Eight bits to be transmitted or just received. A logic 1 in S1DAT corresponds to a high level on the $I^2C$ bus, and a logic 0 corresponds to a low level on the bus. Serial data shifts through S1DAT from right to left. Figure 15 shows how data in S1DAT is serially transferred to and from the SDA line.

S1DAT and the ACK flag form a 9-bit shift register which shifts in or shifts out an 8-bit byte, followed by an acknowledge bit. The ACK flag is controlled by the SIO1 hardware and cannot be accessed by the CPU. Serial data is shifted through the ACK flag into S1DAT on the rising edges of serial clock pulses on the SCL line. When a byte has been shifted into S1DAT, the serial data is available in S1DAT, and the acknowledge bit is returned by the control logic during the ninth clock pulse. Serial data is shifted out from S1DAT via a buffer (BSD7) on the falling edges of clock pulses on the SCL line.

When the CPU writes to S1DAT, BSD7 is loaded with the content of S1DAT.7, which is the first bit to be transmitted to the SDA line (see Figure 16). After nine serial clock pulses, the eight bits in S1DAT will have been transmitted to the SDA line, and the acknowledge bit will be present in ACK. Note that the eight transmitted bits are shifted back into S1DAT.

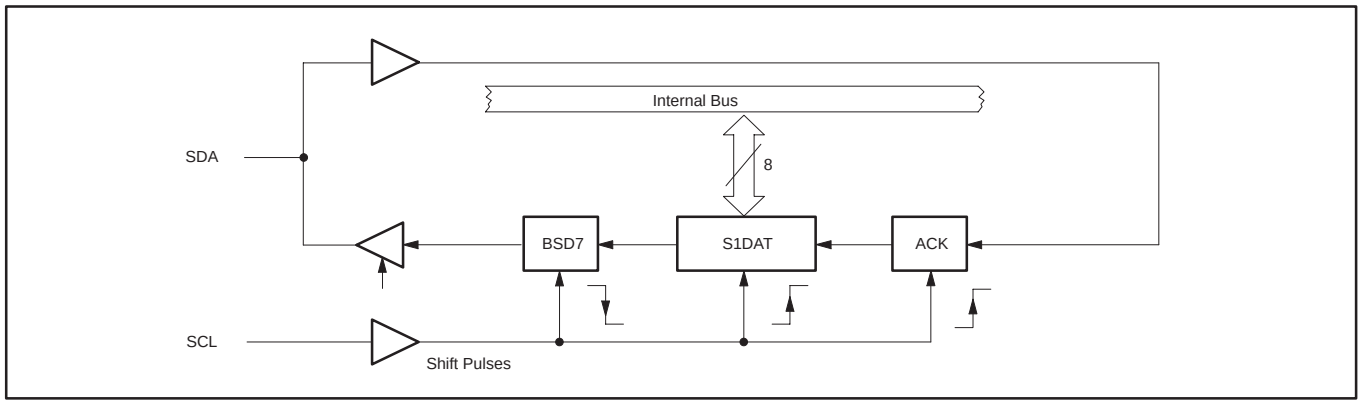



**Figure 15. Serial Input/Output Configuration**

**The Control Register, S1CON:** The CPU can read from and write to this 8-bit, directly addressable SFR. Two bits are affected by the SIO1 hardware: the SI bit is set when a serial interrupt is requested, and the STO bit is cleared when a STOP condition is present on the $I^2C$ bus. The STO bit is also cleared when ENS1 = "0".

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| S1CON (D8H) | CR2 | ENS1 | STA | STO | SI | AA | CR1 | CR0 |

ENS1, THE SIO1 ENABLE BIT

ENS1 = "0": When ENS1 is "0", the SDA and SCL outputs are in a high impedance state. SDA and SCL input signals are ignored, SIO1 is in the "not addressed" slave state, and the STO bit in S1CON is forced to "0". No other bits are affected. P1.6 and P1.7 may be used as open drain I/O ports.

ENS1 = "1": When ENS1 is "1", SIO1 is enabled. The P1.6 and P1.7 port latches must be set to logic 1.

ENS1 should not be used to temporarily release SIO1 from the I2C bus since, when ENS1 is reset, the I2C bus status is lost. The AA flag should be used instead (see description of the AA flag in the following text).

In the following text, it is assumed that ENS1 = "1".

STA, THE START FLAG

STA = "1": When the STA bit is set to enter a master mode, the SIO1 hardware checks the status of the I2C bus and generates a START condition if the bus is free. If the bus is not free, then SIO1 waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a half clock period of the internal serial clock generator.

If STA is set while SIO1 is already in a master mode and one or more bytes are transmitted or received, SIO1 transmits a repeated START condition. STA may be set at any time. STA may also be set when SIO1 is an addressed slave.

STA = "0": When the STA bit is reset, no START condition or repeated START condition will be generated.

STO, THE STOP FLAG

STO = "1": When the STO bit is set while SIO1 is in a master mode, a STOP condition is transmitted to the $I^2C$ bus. When the STOP condition is detected on the bus, the SIO1 hardware clears the STO flag. In a slave mode, the STO flag may be set to recover from an error condition. In this case, no STOP condition is transmitted to the $I^2C$ bus. However, the SIO1 hardware behaves as if a STOP condition has been received and switches to the defined "not addressed" slave receiver mode. The STO flag is automatically cleared by hardware.

If the STA and STO bits are both set, the a STOP condition is transmitted to the $I^2C$ bus if SIO1 is in a master mode (in a slave mode, SIO1 generates an internal STOP condition which is not transmitted). SIO1 then transmits a START condition.

STO = "0": When the STO bit is reset, no STOP condition will be generated.

SI, THE SERIAL INTERRUPT FLAG

SI = "1": When the SI flag is set, then, if the EA and ES1 (interrupt enable register) bits are also set, a serial interrupt is requested. SI is set by hardware when one of 25 of the 26 possible SIO1 states is entered. The only state that does not cause SI to be set is state F8H, which indicates that no relevant state information is available.

While SI is set, the low period of the serial clock on the SCL line is stretched, and the serial transfer is suspended. A high level on the SCL line is unaffected by the serial interrupt flag. SI must be reset by software.

SI = "0": When the SI flag is reset, no serial interrupt is requested, and there is no stretching of the serial clock on the SCL line.

AA, THE ASSERT ACKNOWLEDGE FLAG

AA = "1": If the AA flag is set, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when:

– The "own slave address" has been received
– The general call address has been received while the general call bit (GC) in S1ADR is set
– A data byte has been received while SIO1 is in the master receiver mode
– A data byte has been received while SIO1 is in the addressed slave receiver mode

AA = "0": if the AA flag is reset, a not acknowledge (high level to SDA) will be returned during the acknowledge clock pulse on SCL when:

– A data has been received while SIO1 is in the master receiver mode
– A data byte has been received while SIO1 is in the addressed slave receiver mode

When SIO1 is in the addressed slave transmitter mode, state C8H will be entered after the last serial is transmitted (see Figure 20). When SI is cleared, SIO1 leaves state C8H, enters the not addressed slave receiver mode, and the SDA line remains at a high level. In state C8H, the AA flag can be set again for future address recognition.

When SIO1 is in the not addressed slave mode, its own slave address and the general call address are ignored. Consequently, no acknowledge is returned, and a serial interrupt is not requested. Thus, SIO1 can be temporarily released from the $I^2C$ bus while the bus status is monitored. While SIO1 is released from the bus, START and STOP conditions are detected, and serial data is shifted in. Address recognition can be resumed at any time by setting the AA flag. If the AA flag is set when the part's own slave address or the general call address has been partly received, the address will be recognized at the end of the byte transmission.

CR0, CR1, AND CR2, THE CLOCK RATE BITS

These three bits determine the serial clock frequency when SIO1 is in a master mode. The various serial rates are shown in Table 2.

A 12.5kHz bit rate may be used by devices that interface to the $I^2C$ bus via standard I/O port lines which are software driven and slow. 100kHz is usually the maximum bit rate and can be derived from a 16MHz, 12MHz, or a 6MHz oscillator. A variable bit rate (0.5kHz to 62.5kHz) may also be used if Timer 1 is not required for any other purpose while SIO1 is in a master mode.

The frequencies shown in Table 2 are unimportant when SIO1 is in a slave mode. In the slave modes, SIO1 will automatically synchronize with any clock frequency up to 100kHz.

**The Status Register, S1STA:** S1STA is an 8-bit read-only special function register. The three least significant bits are always zero. The five most significant bits contain the status code. There are 26 possible status codes. When S1STA contains F8H, no relevant state information is available and no serial interrupt is requested. All other S1STA values correspond to defined SIO1 states. When each of these states is entered, a serial interrupt is requested (SI = "1"). A valid status code is present in S1STA one machine cycle after SI is set by hardware and is still present one machine cycle after SI has been reset by software.

**More Information on SIO1 Operating Modes:** The four operating modes are:

– Master Transmitter
– Master Receiver
– Slave Receiver
– Slave Transmitter

Data transfers in each mode of operation are shown in Figures 17–37. These figures contain the following abbreviations:

| Abbreviation | Explanation |
|---|---|
| S | Start condition |
| SLA | 7-bit slave address |
| R | Read bit (high level at SDA) |
| W | Write bit (low level at SDA) |
| A | Acknowledge bit (low level at SDA) |
| $\overline{A}$ | Not acknowledge bit (high level at SDA) |
| Data | 8-bit data byte |
| P | Stop condition |

In Figures 17-37, circles are used to indicate when the serial interrupt flag is set. The numbers in the circles show the status code held in the S1STA register. At these points, a service routine must be executed to continue or complete the serial transfer. These service routines are not critical since the serial transfer is suspended until the serial interrupt flag is cleared by software.

When a serial interrupt routine is entered, the status code in S1STA is used to branch to the appropriate service routine. For each status code, the required software action and details of the following serial transfer are given in Tables 3-7.

**Master Transmitter Mode:** In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 17). Before the master transmitter mode can be entered, S1CON must be initialized as follows:

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| S1CON (D8H) | CR2 | ENS1 | STA | STO | SI | AA | CR1 | CR0 |
| | bit rate | 1 | 0 | 0 | 0 | X | bit rate | |

CR0, CR1, and CR2 define the serial bit rate. ENS1 must be set to logic 1 to enable SIO1. If the AA bit is reset, SIO1 will not acknowledge its own slave address or the general call address in the event of another device becoming master of the bus. In other words, if AA is reset, SIO0 cannot enter a slave mode. STA, STO, and SI must be reset.

The master transmitter mode may now be entered by setting the STA bit using the SETB instruction. The SIO1 logic will now test the $I^2C$ bus and generate a start condition as soon as the bus becomes free. When a START condition is transmitted, the serial interrupt flag (SI) is set, and the status code in the status register (S1STA) will be 08H. This status code must be used to vector to an interrupt service routine that loads S1DAT with the slave address and the data direction bit (SLA+W). The SI bit in S1CON must then be reset before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in S1STA are possible. There are 18H, 20H, or 38H for the master mode and also 68H, 78H, or B0H if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 3. After a repeated start condition (state 10H). SIO1 may switch to the master receiver mode by loading S1DAT with SLA+R).

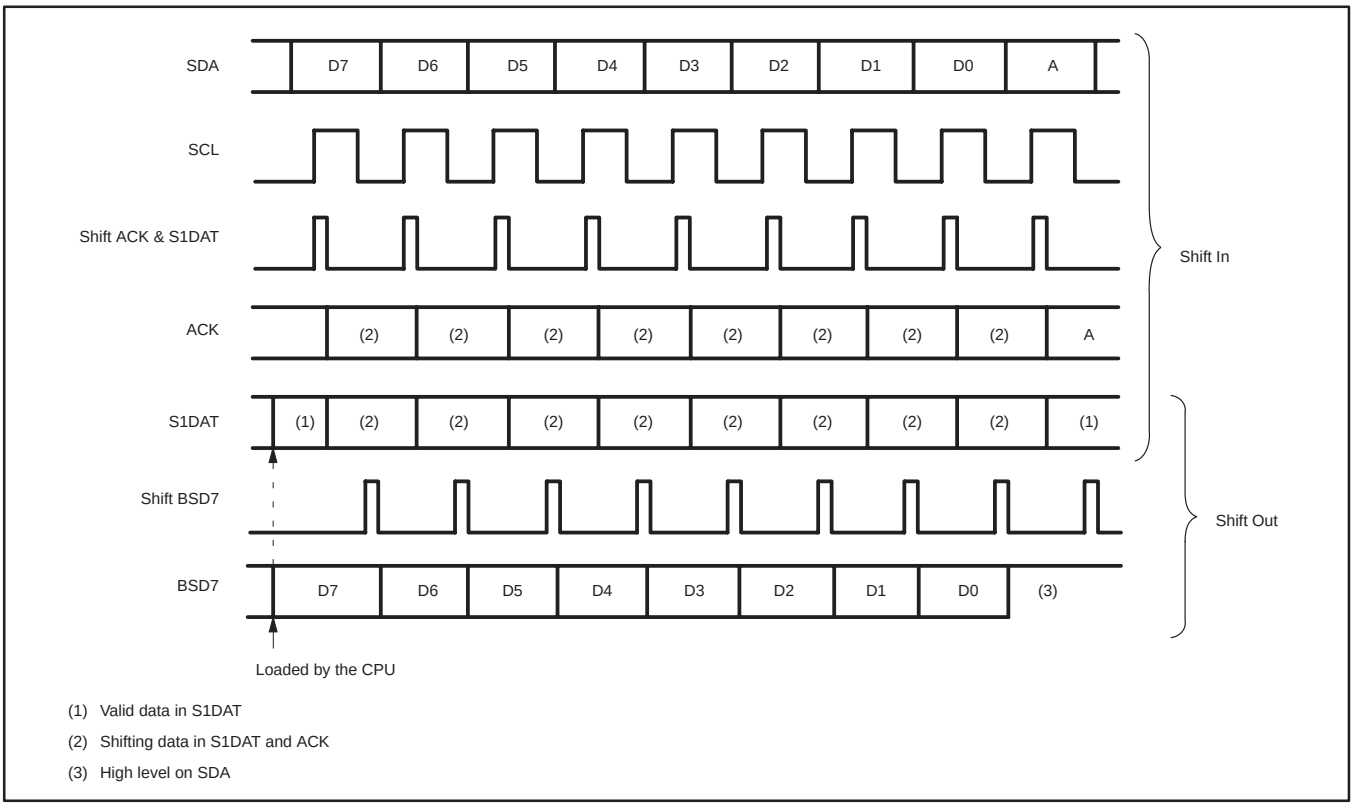



**Figure 16. Shift-in and Shift-out Timing**

## Table 2. Serial Clock Rates

| CR2 | CR1 | CR0 | BIT FREQUENCY (kHz) AT $f_{OSC}$ | | | $f_{OSC}$ DIVIDED BY |
|---|---|---|---|---|---|---|
| | | | 6MHz | 12MHz | 16MHz | |
| 0 | 0 | 0 | 23 | 47 | 63 | 256 |
| 0 | 0 | 1 | 27 | 54 | 71 | 224 |
| 0 | 1 | 0 | 31 | 63 | 83 | 192 |
| 0 | 1 | 1 | 37 | 75 | 100 | 160 |
| 1 | 0 | 0 | 6.25 | 12.5 | 17 | 960 |
| 1 | 0 | 1 | 50 | 100 | – | 120 |
| 1 | 1 | 0 | 100 | – | – | 60 |
| 1 | 1 | 1 | 0.25 < 62.5 | 0.5 < 62.5 | 0.67 < 56 | 96 × (256 – reload value Timer 1)<br>(Reload value range: 0 – 254 in mode 2) |

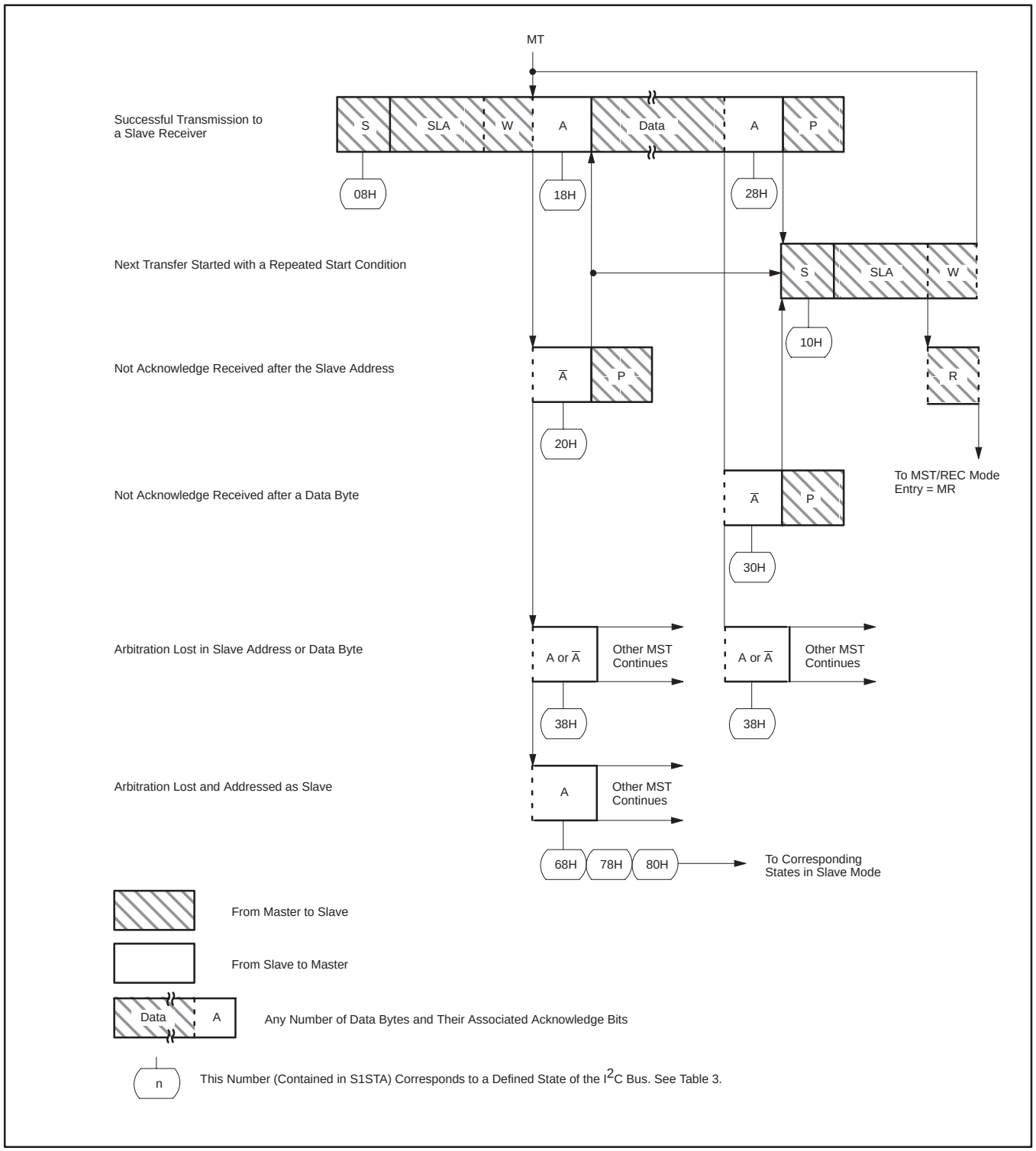



**Figure 17. Format and States in the Master Transmitter Mode**

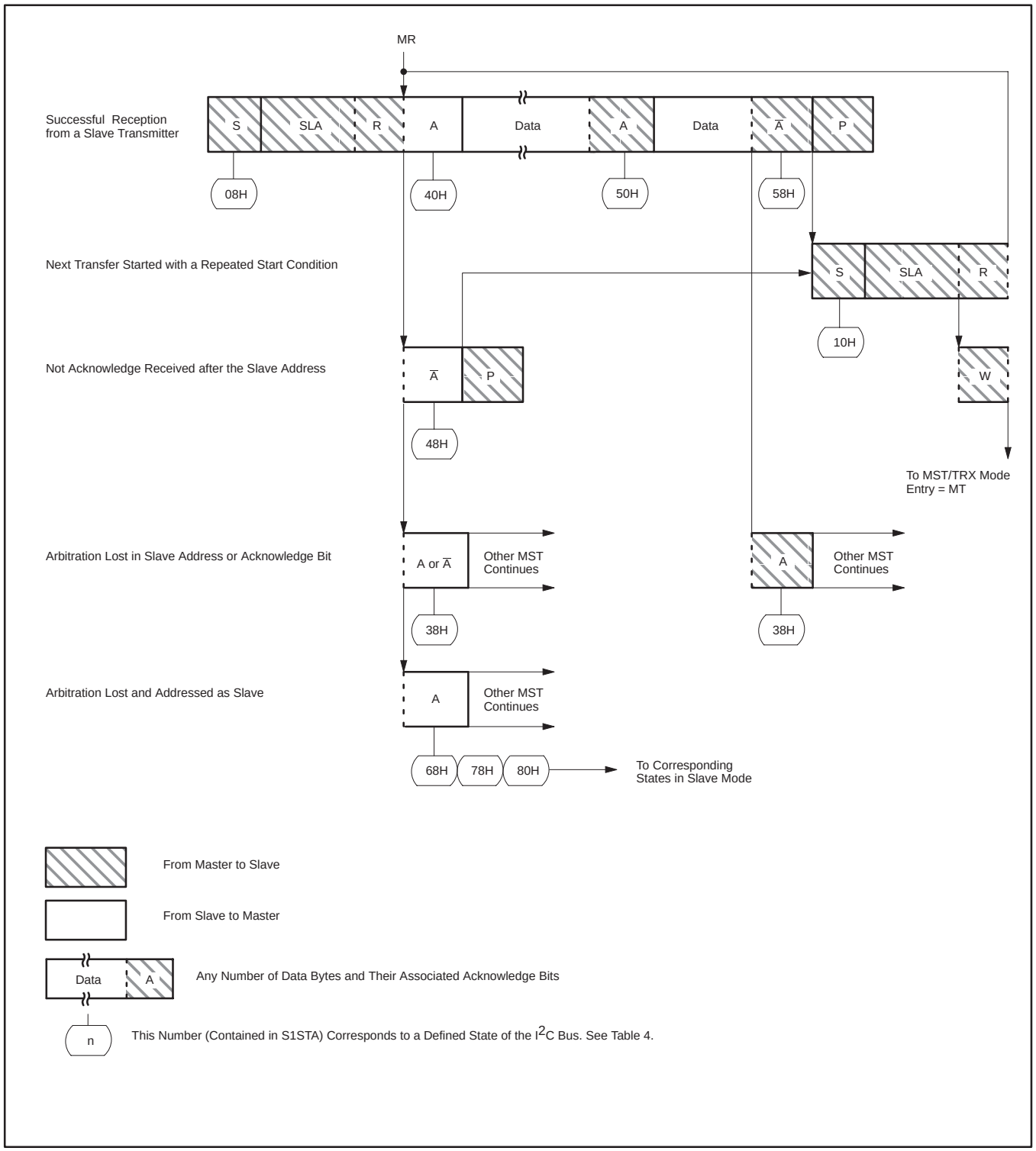



**Figure 18. Format and States in the Master Receiver Mode**

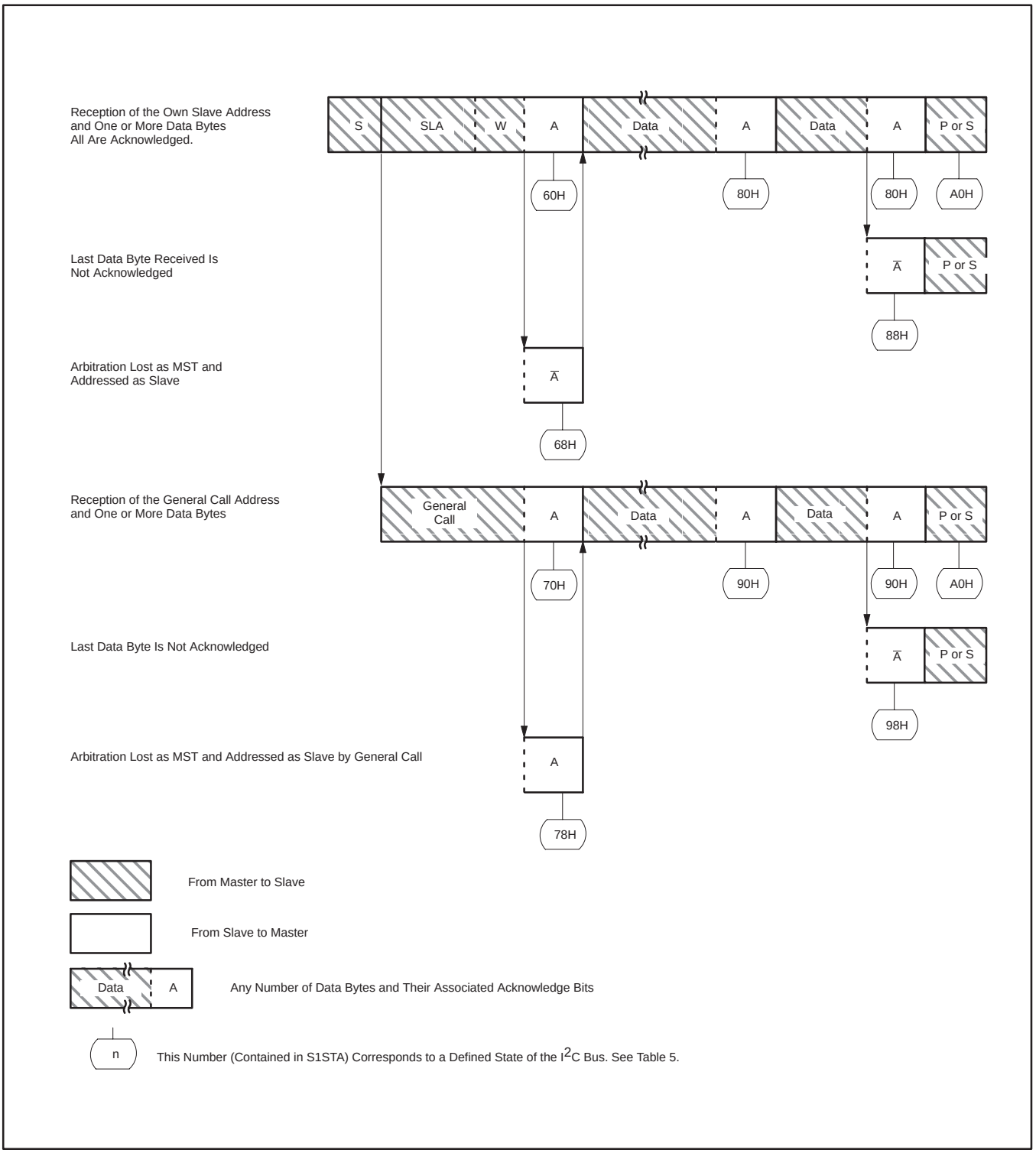



**Figure 19. Format and States in the Slave Receiver Mode**

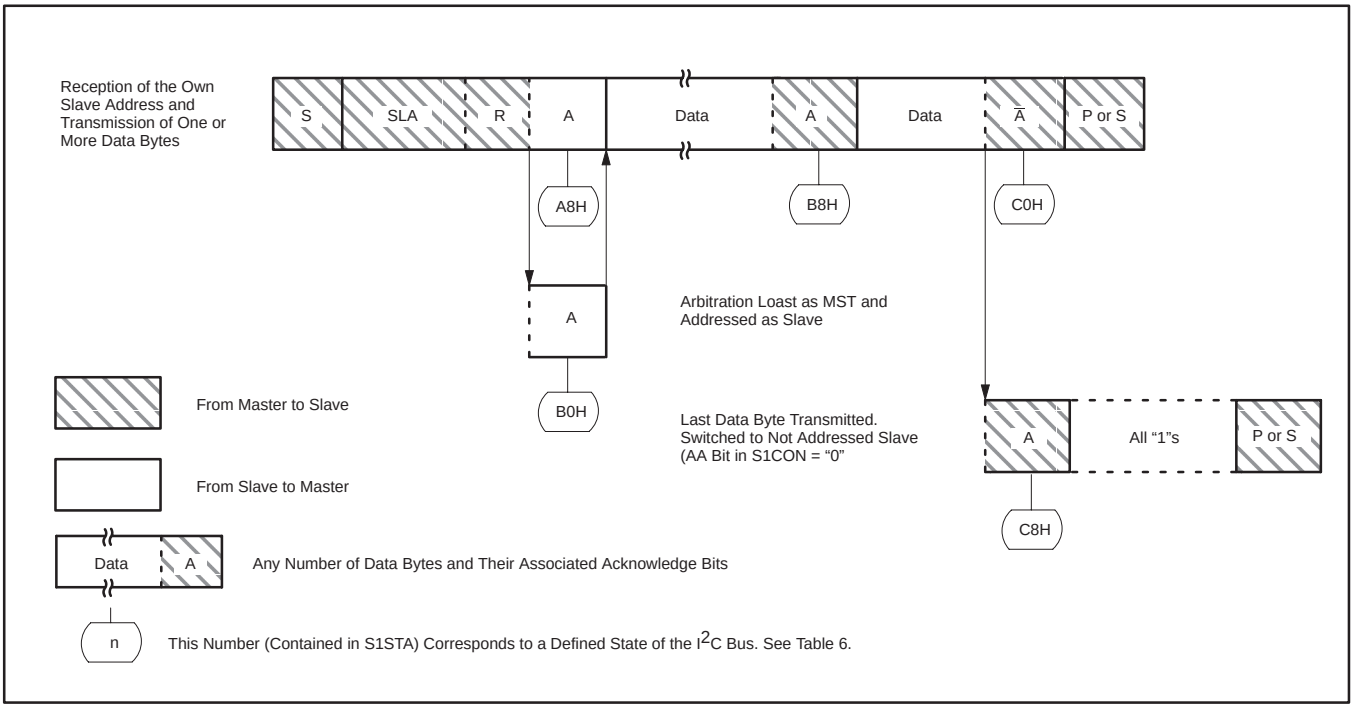



**Figure 20. Format and States of the Slave Transmitter Mode**

**Master Receiver Mode:** In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 18). The transfer is initialized as in the master transmitter mode. When the start condition has been transmitted, the interrupt service routine must load S1DAT with the 7-bit slave address and the data direction bit (SLA+R). The SI bit in S1CON must then be cleared before the serial transfer can continue.

When the slave address and the data direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in S1STA are possible. These are 40H, 48H, or 38H for the master mode and also 68H, 78H, or B0H if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 4. ENS1, CR1, and CR0 are not affected by the serial transfer and are not referred to in Table 4. After a repeated start condition (state 10H), SIO1 may switch to the master transmitter mode by loading S1DAT with SLA+W.

**Slave Receiver Mode:** In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 19). To initiate the slave receiver mode, S1ADR and S1CON must be loaded as follows:

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| S1ADR (DBH) | X | X | X | X | X | X | X | GC |
| | own slave address | | | | | | | |

The upper 7 bits are the address to which SIO1 will respond when addressed by a master. If the LSB (GC) is set, SIO1 will respond to the general call address (00H); otherwise it ignores the general call address.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| S1CON (D8H) | CR2 | ENS1 | STA | STO | SI | AA | CR1 | CR0 |
| | X | 1 | 0 | 0 | 0 | 1 | X | X |

CR0, CR1, and CR2 do not affect SIO1 in the slave mode. ENS1 must be set to logic 1 to enable SIO1. The AA bit must be set to enable SIO1 to acknowledge its own slave address or the general call address. STA, STO, and SI must be reset.

When S1ADR and S1CON have been initialized, SIO1 waits until it is addressed by its own slave address followed by the data direction bit which must be "0" (W) for SIO1 to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (I) is set and a valid status code can be read from S1STA. This status code is used to vector to an interrupt service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 5. The slave receiver mode may also be entered if arbitration is lost while SIO1 is in the master mode (see status 68H and 78H).

If the AA bit is reset during a transfer, SIO1 will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, SIO1 does not respond to its own slave address or a general call address. However, the $I^2C$ bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate SIO1 from the $I^2C$ bus.

**Table 3. Master Transmitter Mode**

| STATUS CODE (S1STA) | STATUS OF THE $I^2C$ BUS AND SIO1 HARDWARE | APPLICATION SOFTWARE RESPONSE | | | | | NEXT ACTION TAKEN BY SIO1 HARDWARE |
|---|---|---|---|---|---|---|---|
| | | TO/FROM S1DAT | TO S1CON | | | | |
| | | | STA | STO | SI | AA | |
| 08H | A START condition has been transmitted | Load SLA+W | X | 0 | 0 | X | SLA+W will be transmitted;<br>ACK bit will be received |
| 10H | A repeated START condition has been transmitted | Load SLA+W or<br>Load SLA+R | X<br>X | 0<br>0 | 0<br>0 | X<br>X | As above<br>SLA+W will be transmitted;<br>SIO1 will be switched to MST/REC mode |
| 18H | SLA+W has been transmitted; ACK has been received | Load data byte or<br><br>no S1DAT action or<br>no S1DAT action or<br><br>no S1DAT action | 0<br><br>1<br>0<br><br>1 | 0<br><br>0<br>1<br><br>1 | 0<br><br>0<br>0<br><br>0 | X<br><br>X<br>X<br><br>X | Data byte will be transmitted;<br>ACK bit will be received<br>Repeated START will be transmitted;<br>STOP condition will be transmitted;<br>STO flag will be reset<br>STOP condition followed by a<br>START condition will be transmitted;<br>STO flag will be reset |
| 20H | SLA+W has been transmitted; NOT ACK has been received | Load data byte or<br><br>no S1DAT action or<br>no S1DAT action or<br><br>no S1DAT action | 0<br><br>1<br>0<br><br>1 | 0<br><br>0<br>1<br><br>1 | 0<br><br>0<br>0<br><br>0 | X<br><br>X<br>X<br><br>X | Data byte will be transmitted;<br>ACK bit will be received<br>Repeated START will be transmitted;<br>STOP condition will be transmitted;<br>STO flag will be reset<br>STOP condition followed by a<br>START condition will be transmitted;<br>STO flag will be reset |
| 28H | Data byte in S1DAT has been transmitted; ACK has been received | Load data byte or<br><br>no S1DAT action or<br>no S1DAT action or<br><br>no S1DAT action | 0<br><br>1<br>0<br><br>1 | 0<br><br>0<br>1<br><br>1 | 0<br><br>0<br>0<br><br>0 | X<br><br>X<br>X<br><br>X | Data byte will be transmitted;<br>ACK bit will be received<br>Repeated START will be transmitted;<br>STOP condition will be transmitted;<br>STO flag will be reset<br>STOP condition followed by a<br>START condition will be transmitted;<br>STO flag will be reset |
| 30H | Data byte in S1DAT has been transmitted; NOT ACK has been received | Load data byte or<br><br>no S1DAT action or<br>no S1DAT action or<br><br>no S1DAT action | 0<br><br>1<br>0<br><br>1 | 0<br><br>0<br>1<br><br>1 | 0<br><br>0<br>0<br><br>0 | X<br><br>X<br>X<br><br>X | Data byte will be transmitted;<br>ACK bit will be received<br>Repeated START will be transmitted;<br>STOP condition will be transmitted;<br>STO flag will be reset<br>STOP condition followed by a<br>START condition will be transmitted;<br>STO flag will be reset |
| 38H | Arbitration lost in SLA+R/$\overline{W}$ or<br>Data bytes | No S1DAT action or<br><br>No S1DAT action | 0<br><br>1 | 0<br><br>0 | 0<br><br>0 | X<br><br>X | $I^2C$ bus will be released;<br>not addressed slave will be entered<br>A START condition will be transmitted when the bus becomes free |

**Table 4. Master Receiver Mode**

| STATUS CODE (S1STA) | STATUS OF THE $I^2C$ BUS AND SIO1 HARDWARE | APPLICATION SOFTWARE RESPONSE | | | | | NEXT ACTION TAKEN BY SIO1 HARDWARE |
|---|---|---|---|---|---|---|---|
| | | TO/FROM S1DAT | TO S1CON | | | | |
| | | | STA | STO | SI | AA | |
| 08H | A START condition has been transmitted | Load SLA+R | X | 0 | 0 | X | SLA+R will be transmitted;<br>ACK bit will be received |
| 10H | A repeated START condition has been transmitted | Load SLA+R or<br>Load SLA+W | X<br>X | 0<br>0 | 0<br>0 | X<br>X | As above<br>SLA+W will be transmitted;<br>SIO1 will be switched to MST/TRX mode |
| 38H | Arbitration lost in NOT ACK bit | No S1DAT action or<br><br>No S1DAT action | 0<br><br>1 | 0<br><br>0 | 0<br><br>0 | X<br><br>X | $I^2C$ bus will be released;<br>SIO1 will enter a slave mode<br>A START condition will be transmitted when the bus becomes free |
| 40H | SLA+R has been transmitted; ACK has been received | No S1DAT action or<br><br>no S1DAT action | 0<br><br>0 | 0<br><br>0 | 0<br><br>0 | 0<br><br>1 | Data byte will be received;<br>NOT ACK bit will be returned<br>Data byte will be received;<br>ACK bit will be returned |
| 48H | SLA+R has been transmitted; NOT ACK has been received | No S1DAT action or<br><br>no S1DAT action or<br><br>no S1DAT action | 1<br><br>0<br><br>1 | 0<br><br>1<br><br>1 | 0<br><br>0<br><br>0 | X<br><br>X<br><br>X | Repeated START condition will be transmitted<br><br>STOP condition will be transmitted;<br>STO flag will be reset<br>STOP condition followed by a<br>START condition will be transmitted;<br>STO flag will be reset |
| 50H | Data byte has been received; ACK has been returned | Read data byte or<br><br>read data byte | 0<br><br>0 | 0<br><br>0 | 0<br><br>0 | 0<br><br>1 | Data byte will be received;<br>NOT ACK bit will be returned<br>Data byte will be received;<br>ACK bit will be returned |
| 58H | Data byte has been received; NOT ACK has been returned | Read data byte or<br><br>read data byte or<br><br>read data byte | 1<br><br>0<br><br>1 | 0<br><br>1<br><br>1 | 0<br><br>0<br><br>0 | X<br><br>X<br><br>X | Repeated START condition will be transmitted<br><br>STOP condition will be transmitted;<br>STO flag will be reset<br>STOP condition followed by a<br>START condition will be transmitted;<br>STO flag will be reset |

**Table 5. Slave Receiver Mode**

| STATUS CODE (S1STA) | STATUS OF THE $I^2C$ BUS AND SIO1 HARDWARE | APPLICATION SOFTWARE RESPONSE | | | | | NEXT ACTION TAKEN BY SIO1 HARDWARE |
|---|---|---|---|---|---|---|---|
| | | TO/FROM S1DAT | TO S1CON | | | | |
| | | | STA | STO | SI | AA | |
| 60H | Own SLA+W has been received; ACK has been returned | No S1DAT action or | X | 0 | 0 | 0 | Data byte will be received and NOT ACK will be returned |
| | | no S1DAT action | X | 0 | 0 | 1 | Data byte will be received and ACK will be returned |
| 68H | Arbitration lost in SLA+R/$\overline{W}$ as master; Own SLA+W has been received, ACK returned | No S1DAT action or | X | 0 | 0 | 0 | Data byte will be received and NOT ACK will be returned |
| | | no S1DAT action | X | 0 | 0 | 1 | Data byte will be received and ACK will be returned |
| 70H | General call address (00H) has been received; ACK has been returned | No S1DAT action or | X | 0 | 0 | 0 | Data byte will be received and NOT ACK will be returned |
| | | no S1DAT action | X | 0 | 0 | 1 | Data byte will be received and ACK will be returned |
| 78H | Arbitration lost in SLA+R/$\overline{W}$ as master; General call address has been received, ACK has been returned | No S1DAT action or | X | 0 | 0 | 0 | Data byte will be received and NOT ACK will be returned |
| | | no S1DAT action | X | 0 | 0 | 1 | Data byte will be received and ACK will be returned |
| 80H | Previously addressed with own SLV address; DATA has been received; ACK has been returned | Read data byte or | X | 0 | 0 | 0 | Data byte will be received and NOT ACK will be returned |
| | | read data byte | X | 0 | 0 | 1 | Data byte will be received and ACK will be returned |
| 88H | Previously addressed with own SLA; DATA byte has been received; NOT ACK has been returned | Read data byte or | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address |
| | | read data byte or | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1 |
| | | read data byte or | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free |
| | | read data byte | 1 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free. |
| 90H | Previously addressed with General Call; DATA byte has been received; ACK has been returned | Read data byte or | X | 0 | 0 | 0 | Data byte will be received and NOT ACK will be returned |
| | | read data byte | X | 0 | 0 | 1 | Data byte will be received and ACK will be returned |
| 98H | Previously addressed with General Call; DATA byte has been received; NOT ACK has been returned | Read data byte or | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address |
| | | read data byte or | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1 |
| | | read data byte or | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free |
| | | read data byte | 1 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free. |

**Table 5. Slave Receiver Mode** (Continued)

| STATUS CODE (S1STA) | STATUS OF THE $I^2C$ BUS AND SIO1 HARDWARE | APPLICATION SOFTWARE RESPONSE | | | | | NEXT ACTION TAKEN BY SIO1 HARDWARE |
|---|---|---|---|---|---|---|---|
| | | TO/FROM S1DAT | TO S1CON | | | | |
| | | | STA | STO | SI | AA | |
| A0H | A STOP condition or repeated START condition has been received while still addressed as SLV/REC or SLV/TRX | No STDAT action or | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address |
| | | No STDAT action or | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1 |
| | | No STDAT action or | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free |
| | | No STDAT action | 1 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free. |

**Table 6. Slave Transmitter Mode**

| STATUS CODE (S1STA) | STATUS OF THE $I^2C$ BUS AND SIO1 HARDWARE | APPLICATION SOFTWARE RESPONSE | | | | | NEXT ACTION TAKEN BY SIO1 HARDWARE |
|---|---|---|---|---|---|---|---|
| | | TO/FROM S1DAT | TO S1CON | | | | |
| | | | STA | STO | SI | AA | |
| A8H | Own SLA+R has been received; ACK has been returned | Load data byte or | X | 0 | 0 | 0 | Last data byte will be transmitted and ACK bit will be received |
| | | load data byte | X | 0 | 0 | 1 | Data byte will be transmitted; ACK will be received |
| B0H | Arbitration lost in SLA+R/$\overline{W}$ as master; Own SLA+R has been received, ACK has been returned | Load data byte or | X | 0 | 0 | 0 | Last data byte will be transmitted and ACK bit will be received |
| | | load data byte | X | 0 | 0 | 1 | Data byte will be transmitted; ACK bit will be received |
| B8H | Data byte in S1DAT has been transmitted; ACK has been received | Load data byte or | X | 0 | 0 | 0 | Last data byte will be transmitted and ACK bit will be received |
| | | load data byte | X | 0 | 0 | 1 | Data byte will be transmitted; ACK bit will be received |
| C0H | Data byte in S1DAT has been transmitted; NOT ACK has been received | No S1DAT action or | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address |
| | | no S1DAT action or | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1 |
| | | no S1DAT action or | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free |
| | | no S1DAT action | 1 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free. |
| C8H | Last data byte in S1DAT has been transmitted (AA = 0); ACK has been received | No S1DAT action or | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address |
| | | no S1DAT action or | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1 |
| | | no S1DAT action or | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free |
| | | no S1DAT action | 1 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free. |

**Slave Transmitter Mode:** In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 20). Data transfer is initialized as in the slave receiver mode. When S1ADR and S1CON have been initialized, SIO1 waits until it is addressed by its own slave address followed by the data direction bit which must be "1" (R) for SIO1 to operate in the slave transmitter mode. After its own slave address and the R bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from S1STA. This status code is used to vector to an interrupt service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 6. The slave transmitter mode may also be entered if arbitration is lost while SIO1 is in the master mode (see state B0H).

If the AA bit is reset during a transfer, SIO1 will transmit the last byte of the transfer and enter state C0H or C8H. SIO1 is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, SIO1 does not respond to its own slave address or a general call address. However, the $I^2C$ bus is still monitored, and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate SIO1 from the $I^2C$ bus.

**Miscellaneous States:** There are two S1STA codes that do not correspond to a defined SIO1 hardware state (see Table 7). These are discussed below.

**S1STA = F8H:**
This status code indicates that no relevant information is available because the serial interrupt flag, SI, is not yet set. This occurs between other states and when SIO1 is not involved in a serial transfer.

**S1STA = 00H:**
This status code indicates that a bus error has occurred during an SIO1 serial transfer. A bus error is caused when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. A bus error may also be caused when external interference disturbs the internal SIO1 signals. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared. This causes SIO1 to enter the "not addressed" slave mode (a defined state) and to clear the STO flag (no other bits in S1CON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

**Some Special Cases:** The SIO1 hardware has facilities to handle the following special cases that may occur during a serial transfer:

Simultaneous Repeated START Conditions from Two Masters

A repeated START condition may be generated in the master transmitter or master receiver modes. A special case occurs if another master simultaneously generates a repeated START condition (see Figure 21). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

If the SIO1 hardware detects a repeated START condition on the $I^2C$ bus before generating a repeated START condition itself, it will release the bus, and no interrupt request is generated. If another master frees the bus by generating a STOP condition, SIO1 will transmit a normal START condition (state 08H), and a retry of the total serial data transfer can commence.

Data Transfer After Loss of Arbitration
Arbitration may be lost in the master transmitter and master receiver modes (see Figure 13). Loss of arbitration is indicated by the following states in S1STA; 38H, 68H, 78H, and B0H (see Figures 17 and 18).

If the STA flag in S1CON is set by the routines which service these states, then, if the bus is free again, a START condition (state 08H) is transmitted without intervention by the CPU, and a retry of the total serial transfer can commence.

Forced Access to the $I^2C$ Bus
In some applications, it may be possible for an uncontrolled source to cause a bus hang-up. In such situations, the problem may be caused by interference, temporary interruption of the bus or a temporary short-circuit between SDA and SCL.

If an uncontrolled source generates a superfluous START or masks a STOP condition, then the $I^2C$ bus stays busy indefinitely. If the STA flag is set and bus access is not obtained within a reasonable amount of time, then a forced access to the $I^2C$ bus is possible. This is achieved by setting the STO flag while the STA flag is still set. No STOP condition is transmitted. The SIO1 hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware (see Figure 22).

$I^2C$ Bus Obstructed by a Low Level on SCL or SDA
An $I^2C$ bus hang-up occurs if SDA or SCL is pulled LOW by an uncontrolled source. If the SCL line is obstructed (pulled LOW) by a device on the bus, no further serial transfer is possible, and the SIO1 hardware cannot resolve this type of problem. When this occurs, the problem must be resolved by the device that is pulling the SCL bus line LOW.

If the SDA line is obstructed by another device on the bus (e.g., a slave device out of bit synchronization), the problem can be solved by transmitting additional clock pulses on the SCL line (see Figure 23). The SIO1 hardware transmits additional clock pulses when the STA flag is set, but no START condition can be generated because the SDA line is pulled LOW while the $I^2C$ bus is considered free. The SIO1 hardware attempts to generate a START condition after every two additional clock pulses on the SCL line. When the SDA line is eventually released, a normal START condition is transmitted, state 08H is entered, and the serial transfer continues.

If a forced bus access occurs or a repeated START condition is transmitted while SDA is obstructed (pulled LOW), the SIO1 hardware performs the same action as described above. In each case, state 08H is entered after a successful START condition is transmitted and normal serial transfer continues. Note that the CPU is not involved in solving these bus hang-up problems.

Bus Error
A bus error occurs when a START or STOP condition is present at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data or an acknowledge bit.

The SIO1 hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, SIO1 immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 00H. This status code may be used to vector to a service routine which either attempts the aborted serial transfer again or simply recovers from the error condition as shown in Table 7.

**Table 7. Miscellaneous States**

| STATUS CODE (S1STA) | STATUS OF THE $I^2C$ BUS AND SIO1 HARDWARE | APPLICATION SOFTWARE RESPONSE | | | | | NEXT ACTION TAKEN BY SIO1 HARDWARE |
|---|---|---|---|---|---|---|---|
| | | TO/FROM S1DAT | TO S1CON | | | | |
| | | | STA | STO | SI | AA | |
| F8H | No relevant state information available; SI = 0 | No S1DAT action | No S1CON action | | | | Wait or proceed current transfer |
| 00H | Bus error during MST or selected slave modes, due to an illegal START or STOP condition. State 00H can also occur when interference causes SIO1 to enter an undefined state. | No S1DAT action | 0 | 1 | 0 | X | Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and SIO1 is switched to the not addressed SLV mode. STO is reset. |

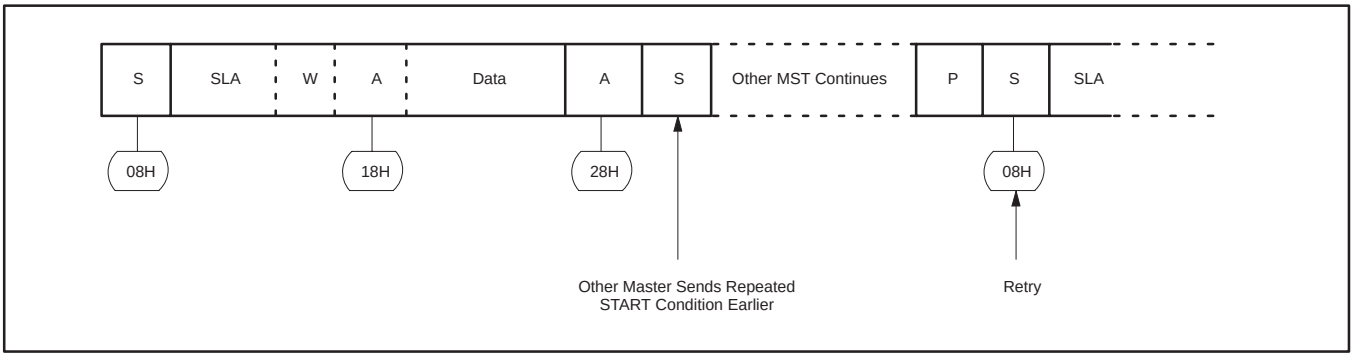



**Figure 21. Simultaneous Repeated START Conditions from 2 Masters**

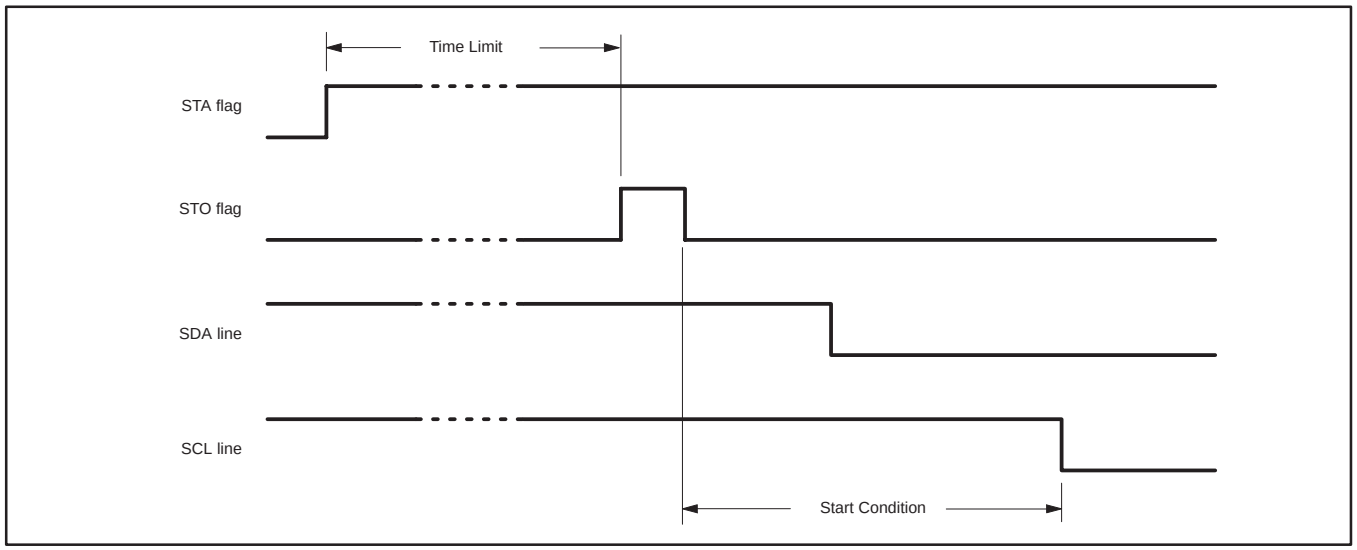



**Figure 22. Forced Access to a Busy $I^2C$ Bus**

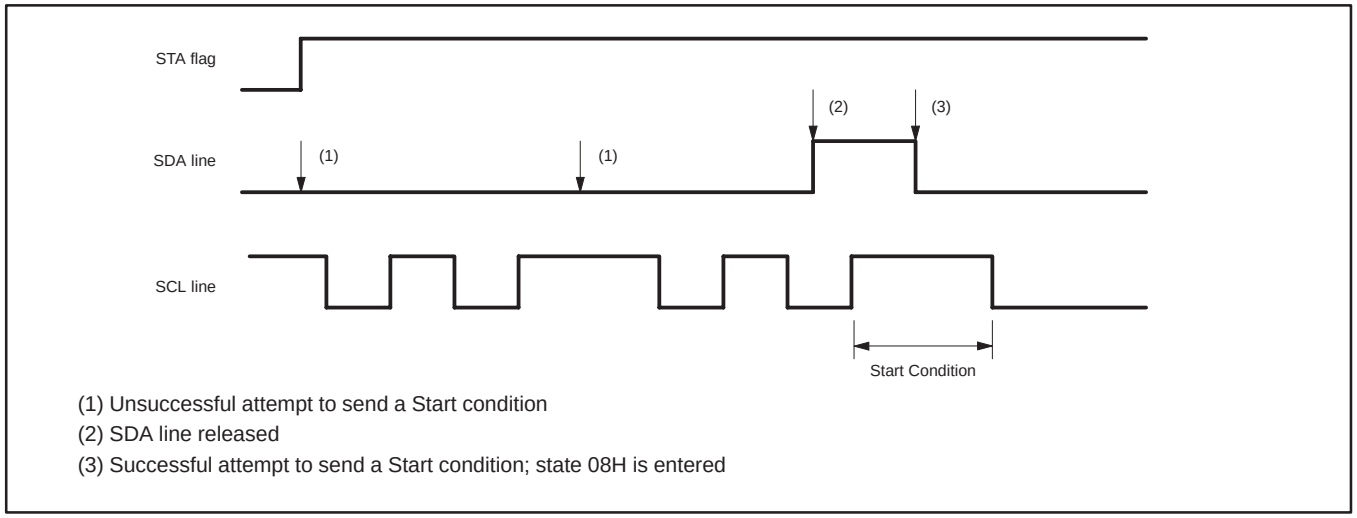



**Figure 23. Recovering from a Bus Obstruction Caused by a Low Level on SDA**

**Software Examples of SIO1 Service Routines:** This section consists of a software example for:

– Initialization of SIO1 after a RESET
– Entering the SIO1 interrupt routine
– The 26 state service routines for the
  – Master transmitter mode
  – Master receiver mode
  – Slave receiver mode
  – Slave transmitter mode

INITIALIZATION

In the initialization routine, SIO1 is enabled for both master and slave modes. For each mode, a number of bytes of internal data RAM are allocated to the SIO to act as either a transmission or reception buffer. In this example, 8 bytes of internal data RAM are reserved for different purposes. The data memory map is shown in Figure 24. The initialization routine performs the following functions:

– S1ADR is loaded with the part's own slave address and the general call bit (GC)
– P1.6 and P1.7 bit latches are loaded with logic 1s
– RAM location HADD is loaded with the high-order address byte of the service routines
– The SIO1 interrupt enable and interrupt priority bits are set
– The slave mode is enabled by simultaneously setting the ENS1 and AA bits in S1CON and the serial clock frequency (for master modes) is defined by loading CR0 and CR1 in S1CON. The master routines must be started in the main program.

The SIO1 hardware now begins checking the $I^2C$ bus for its own slave address and general call. If the general call or the own slave address is detected, an interrupt is requested and S1STA is loaded with the appropriate state information. The following text describes a fast method of branching to the appropriate service routine.

SIO1 INTERRUPT ROUTINE

When the SIO1 interrupt is entered, the PSW is first pushed on the stack. Then S1STA and HADD (loaded with the high-order address byte of the 26 service routines by the initialization routine) are pushed on to the stack. S1STA contains a status code which is the lower byte of one of the 26 service routines. The next instruction is RET, which is the return from subroutine instruction. When this instruction is executed, the high and low order address bytes are popped from stack and loaded into the program counter.

The next instruction to be executed is the first instruction of the state service routine. Seven bytes of program code (which execute in eight machine cycles) are required to branch to one of the 26 state service routines.

```
SI      PUSH  PSW     Save PSW
        PUSH  S1STA   Push status code
                      (low order address byte)
        PUSH  HADD    Push high order address byte
        RET           Jump to state service routine
```

The state service routines are located in a 256-byte page of program memory. The location of this page is defined in the initialization routine. The page can be located anywhere in program memory by loading data RAM register HADD with the page number. Page 01 is chosen in this example, and the service routines are located between addresses 0100H and 01FFH.

THE STATE SERVICE ROUTINES

The state service routines are located 8 bytes from each other. Eight bytes of code are sufficient for most of the service routines. A few of the routines require more than 8 bytes and have to jump to other locations to obtain more bytes of code. Each state routine is part of the SIO1 interrupt routine and handles one of the 26 states. It ends with a RETI instruction which causes a return to the main program.

MASTER TRANSMITTER AND MASTER RECEIVER MODES

The master mode is entered in the main program. To enter the master transmitter mode, the main program must first load the internal data RAM with the slave address, data bytes, and the number of data bytes to be transmitted. To enter the master receiver mode, the main program must first load the internal data RAM with the slave address and the number of data bytes to be received. The R/$\overline{W}$ bit determines whether SIO1 operates in the master transmitter or master receiver mode.

Master mode operation commences when the STA bit in S1CION is set by the SETB instruction and data transfer is controlled by the master state service routines in accordance with Table 3, Table 4, Figure 17, and Figure 18. In the example below, 4 bytes are transferred. There is no repeated START condition. In the event of lost arbitration, the transfer is restarted when the bus becomes free. If a bus error occurs, the $I^2C$ bus is released and SIO1 enters the not selected slave receiver mode. If a slave device returns a not acknowledge, a STOP condition is generated.

A repeated START condition can be included in the serial transfer if the STA flag is set instead of the STO flag in the state service routines vectored to by status codes 28H and 58H. Additional software must be written to determine which data is transferred after a repeated START condition.

SLAVE TRANSMITTER AND SLAVE RECEIVER MODES

After initialization, SIO1 continually tests the $I^2C$ bus and branches to one of the slave state service routines if it detects its own slave address or the general call address (see Table 5, Table 6, Figure 19, and Figure 20). If arbitration was lost while in the master mode, the master mode is restarted after the current transfer. If a bus error occurs, the $I^2C$ bus is released and SIO1 enters the not selected slave receiver mode.

In the slave receiver mode, a maximum of 8 received data bytes can be stored in the internal data RAM. A maximum of 8 bytes ensures that other RAM locations are not overwritten if a master sends more bytes. If more than 8 bytes are transmitted, a not acknowledge is returned, and SIO1 enters the not addressed slave receiver mode. A maximum of one received data byte can be stored in the internal data RAM after a general call address is detected. If more than one byte is transmitted, a not acknowledge is returned and SIO1 enters the not addressed slave receiver mode.

In the slave transmitter mode, data to be transmitted is obtained from the same locations in the internal data RAM that were previously loaded by the main program. After a not acknowledge has been returned by a master receiver device, SIO1 enters the not addressed slave mode.

ADAPTING THE SOFTWARE FOR DIFFERENT APPLICATIONS

The following software example shows the typical structure of the interrupt routine including the 26 state service routines and may be used as a base for user applications. If one or more of the four modes are not used, the associated state service routines may be removed but, care should be taken that a deleted routine can never be invoked.

This example does not include any time-out routines. In the slave modes, time-out routines are not very useful since, in these modes, SIO1 behaves essentially as a passive device. In the master modes, an internal timer may be used to cause a time-out if a serial transfer is not complete after a defined period of time. This time period is defined by the system connected to the $I^2C$ bus.

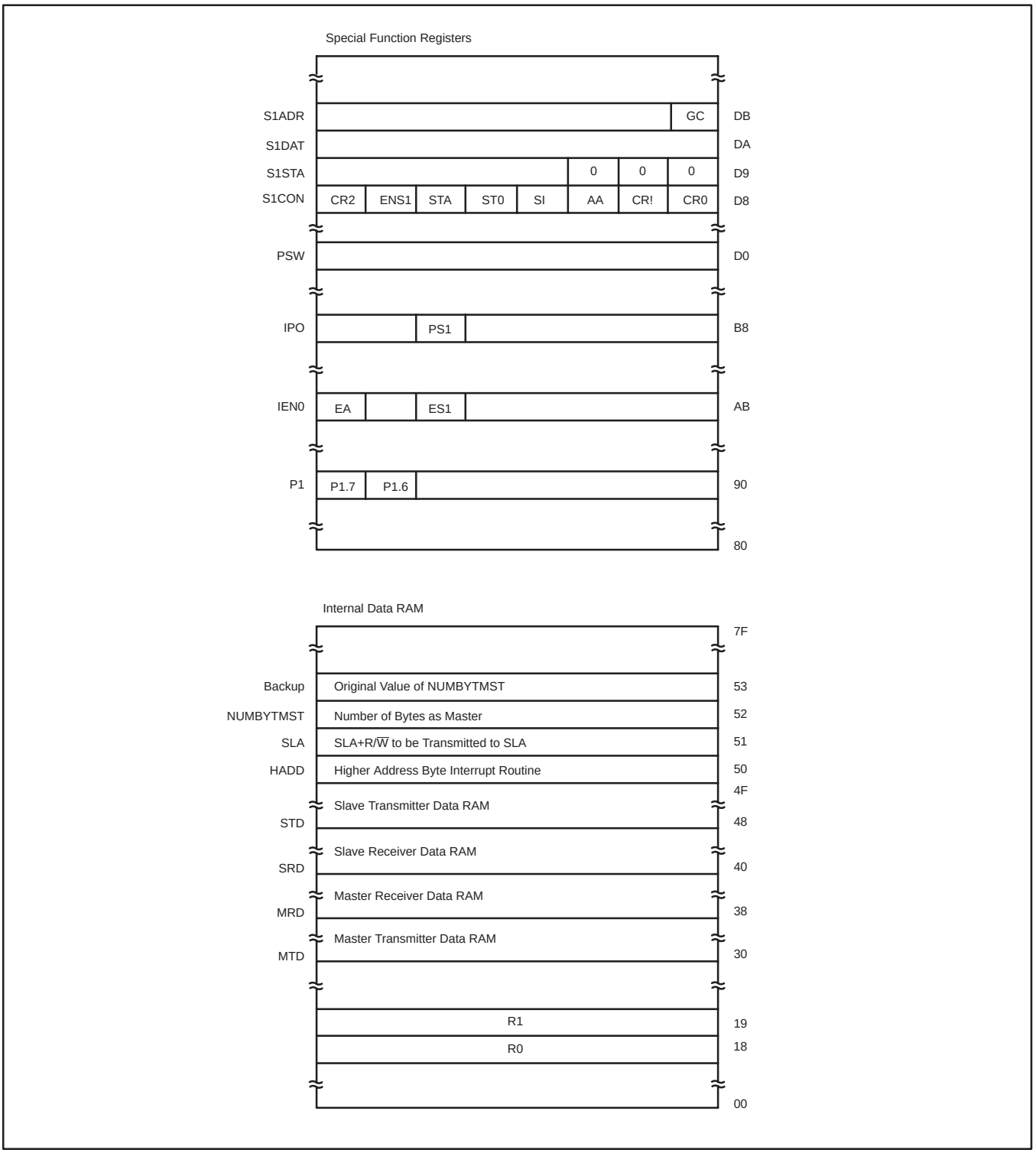



**Figure 24. SIO1 Data Memory Map**

```
                        !*****************************************************************************************************
                        ! SI01 EQUATE LIST
                        !*****************************************************************************************************
                        !*****************************************************************************************************
                        ! LOCATIONS OF THE SI01 SPECIAL FUNCTION REGISTERS
                        !*****************************************************************************************************
00D8                    S1CON       –0xd8
00D9                    S1STA       –0xd9
00DA                    S1DAT       –0xda
00DB                    S1ADR       –0xdb

00A8                    IEN0        –0xa8
00B8                    IP0         –02b8

                        !*****************************************************************************************************
                        ! BIT LOCATIONS
                        !*****************************************************************************************************

00DD                    STA         –0xdd                       ! STA bit in S1CON
00BD                    SI01HP      –0xbd                       ! IP0, SI01 Priority bit

                        !*****************************************************************************************************
                        ! IMMEDIATE DATA TO WRITE INTO REGISTER S1CON
                        !*****************************************************************************************************
00D5                    ENS1_NOTSTA_STO_NOTSI_AA_CR0            –0xd5   ! Generates STOP
                                                                        ! (CR0 = 100kHz)
00C5                    ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0         –0xc5   ! Releases BUS and
                                                                        ! ACK
00C1                    ENS1_NOTSTA_NOTSTO_NOTSI_NOTAA_CR0      –0xc1   ! Releases BUS and
                                                                        ! NOT ACK
00E5                    ENS1_STA_NOTSTO_NOTSI_AA_CR0            –0xe5   ! Releases BUS and
                                                                        ! set STA

                        !*****************************************************************************************************
                        ! GENERAL IMMEDIATE DATA
                        !*****************************************************************************************************
0031                    OWNSLA      –0x31                       ! Own SLA+General Call
                                                                ! must be written into S1ADR
00A0                    ENSI01      –0xa0                       ! EA+ES1, enable SIO1 interrupt
                                                                ! must be written into IEN0
0001                    PAG1        –0x01                       ! select PAG1 as HADD
00C0                    SLAW        –0xc0                       ! SLA+W to be transmitted
00C1                    SLAR        –0xc1                       ! SLA+R to be transmitted
0018                    SELRB3      –0x18                       ! Select Register Bank 3

                        !*****************************************************************************************************
                        ! LOCATIONS IN DATA RAM
                        !*****************************************************************************************************
0030                    MTD         –0x30                       ! MST/TRX/DATA base address
0038                    MRD         –0x38                       ! MST/REC/DATA base address
0040                    SRD         –0x40                       ! SLV/REC/DATA base address
0048                    STD         –0x48                       ! SLV/TRX/DATA base address

0053                    BACKUP          –0x53                   ! Backup from NUMBYTMST
                                                                ! To restore NUMBYTMST in case
                                                                ! of an Arbitration Loss.
0052                    NUMBYTMST       –0x52                   ! Number of bytes to transmit
                                                                ! or receive as MST.
0051                    SLA             –0x51                   ! Contains SLA+R/W to be
                                                                ! transmitted.
0050                    HADD            –0x50                   ! High Address byte for STATE 0
                                                                ! till STATE 25.
```

```
                        !*************************************************************************************************
                        ! INITIALIZATION ROUTINE
                        ! Example to initialize IIC Interface as slave receiver or slave transmitter and
                        ! start a MASTER TRANSMIT or a MASTER RECEIVE function. 4 bytes will be transmitted or received.
                        !*************************************************************************************************
                        .sect   strt
                        .base   0x00
0000    4100                            ajmp    INIT                    ! RESET

                        .sect   initial
                        .base   0x200
0200    75DB31          INIT:           mov     S1ADR,#OWNSLA           ! Load own SLA + enable
                                                                        ! general call recognition
0203    D296                            setb    P1(6)                   ! P1.6 High level.
0205    D297                            setb    P1(7)                   ! P1.7 High level.
0207    755001                          mov     HADD,#PAG1
020A    43A8A0                          orl     IEN0,#ENSI01            ! Enable SI01 interrupt
020D    C2BD                            clr     SI01HP                  ! SI01 interrupt low priority
020F    75D8C5                          mov     S1CON, #ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                        ! Initialize SLV funct.

                        !*************************************************************************************************

                        !-------------------------------------------------
                        ! START MASTER TRANSMIT FUNCTION
                        !-------------------------------------------------

0212    755204                          mov     NUMBYTMST,#0x4          ! Transmit 4 bytes.
0215    7551C0                          mov     SLA,#SLAW               ! SLA+W, Transmit funct.
0218    D2DD                            setb    STA                     ! set STA in S1CON

                        !-------------------------------------------------
                        ! START MASTER RECEIVE FUNCTION
                        !-------------------------------------------------
021A    755204                          mov     NUMBYTMST,#0x4          ! Receive 4 bytes.
021D    7551C1                          mov     SLA,#SLAR               ! SLA+R, Receive funct.
0220    D2DD                            setb    STA                     ! set STA in S1CON


                        !*************************************************************************************************
                        ! SI01 INTERRUPT ROUTINE
                        !*************************************************************************************************
                        .sect   intvec                                  ! SI01 interrupt vector
                        .base   0x00

                        ! S1STA and HADD are pushed onto the stack.
                        ! They serve as return address for the RET instruction.
                        ! The RET instruction sets the Program Counter to address HADD,
                        ! S1STA and jumps to the right subroutine.

002B    C0D0                            push    psw                     ! save  psw
002D    C0D9                            push    S1STA
002F    C050                            push    HADD
0031    22                              ret                             ! JMP to address HADD,S1STA.

                        !-------------------------------------------------
                        ! STATE    : 00, Bus error.
                        ! ACTION   : Enter not addressed SLV mode and release bus. STO reset.
                        !-------------------------------------------------
                        .sect   st0
                        .base   0x100

0100    75D8D5                          mov     S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0 ! clr SI
                                                                                    ! set  STO,AA
0103    D0D0                            pop     psw
0105    32                              reti
```

```
                        !*****************************************************************************************************
                        !*****************************************************************************************************
                        ! MASTER STATE SERVICE ROUTINES
                        !*****************************************************************************************************
                        ! State 08 and State 10 are both for MST/TRX and MST/REC.
                        ! The R/W bit decides whether the next state is within
                        ! MST/TRX mode or within MST/REC mode.
                        !*****************************************************************************************************

                        !-------------------------------------------------------------
                        ! STATE    : 08, A, START condition has been transmitted.
                        ! ACTION   : SLA+R/W are transmitted, ACK bit is received.
                        !-------------------------------------------------------------
                        .sect      mts8
                        .base      0x108

0108    8551DA                                mov   S1DAT,SLA                  ! Load  SLA+R/W
010B    75D8C5                                mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                                 ! clr  SI
010E    01A0                                  ajmp  INITBASE1

                        !-------------------------------------------------------------
                        ! STATE    : 10, A repeated START condition has been
                        !              transmitted.
                        ! ACTION   : SLA+R/W are transmitted, ACK bit is received.
                        !-------------------------------------------------------------
                        .sect      mts10
                        .base      0x110

0110    8551DA                                mov   S1DAT,SLA                  ! Load  SLA+R/W
0113    75D8C5                                mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                                 ! clr  SI
010E    01A0                                  ajmp  INITBASE1

                        .sect      ibase1
                        .base      0xa0
00A0    75D018  INITBASE1:                    mov   psw,#SELRB3
00A3    7930                                  mov   r1,#MTD
00A5    7838                                  mov   r0,#MRD
00A7    855253                                mov   BACKUP,NUMBYTMST           ! Save initial value
00AA    D0D0                                  pop   psw
00AC    32                                    reti



                        !*****************************************************************************************************
                        !*****************************************************************************************************
                        ! MASTER TRANSMITTER STATE SERVICE ROUTINES
                        !*****************************************************************************************************
                        !*****************************************************************************************************

                        !-------------------------------------------------------------
                        ! STATE    : 18, Previous state was STATE 8 or STATE 10, SLA+W have been transmitted,
                        !              ACK has been received.
                        ! ACTION   : First DATA is transmitted, ACK bit  is received.
                        !-------------------------------------------------------------
                        .sect      mts18
                        .base      0x118

0118    75D018                                mov   psw,#SELRB3
011B    87DA                                  mov   S1DAT,@r1
011D    01B5                                  ajmp  CON
```

```
                        !-------------------------------------------------------
                        ! STATE    : 20, SLA+W have been transmitted, NOT ACK has been received
                        ! ACTION   : Transmit STOP condition.
                        !-------------------------------------------------------
                        .sect       mts20
                        .base       0x120

0120    75D8D5                              mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                                    ! set  STO, clr  SI
0123    D0D0                                pop   psw
0125    32                                  reti

                        !-------------------------------------------------------
                        ! STATE    : 28, DATA of S1DAT have been transmitted, ACK received.
                        ! ACTION   : If Transmitted DATA is last DATA then transmit a STOP condition,
                        !              else transmit next DATA.
                        !-------------------------------------------------------
                        .sect       mts28
                        .base       0x128

0128    D55285                              djnz  NUMBYTMST,NOTLDAT1      ! JMP if NOT last DATA
012B    75D8D5                              mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                                    ! clr  SI, set  AA
012E    01B9                                ajmp  RETmt

                        .sect       mts28sb
                        .base       0x0b0
00B0    75D018          NOTLDAT1:           mov   psw,#SELRB3
00B3    87DA                                mov   S1DAT,@r1
00B5    75D8C5          CON:                mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                    ! clr  SI, set   AA
00B8    09                                  inc   r1
00B9    D0D0            RETmt       :       pop   psw
00BB    32                                  reti
                        !-------------------------------------------------------
                        ! STATE    : 30, DATA of S1DAT have been transmitted, NOT ACK received.
                        ! ACTION   : Transmit a STOP condition.
                        !-------------------------------------------------------
                        .sect       mts30
                        .base       0x130

0130    75D8D5                              mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                                    ! set  STO, clr  SI
0133    D0D0                                pop   psw
0135    32                                  reti

                        !-------------------------------------------------------
                        ! STATE    : 38, Arbitration lost in SLA+W or DATA.
                        ! ACTION   : Bus is released, not addressed SLV mode is entered.
                        !              A new START condition is transmitted when the IIC bus is free again.
                        !-------------------------------------------------------
                        .sect       mts38
                        .base       0x138

0138    75D8E5                              mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
013B    855352                              mov   NUMBYTMST,BACKUP
013E    01B9                                ajmp  RETmt
```

```
                        !*****************************************************************************************
                        !*****************************************************************************************
                        ! MASTER RECEIVER STATE SERVICE ROUTINES
                        !*****************************************************************************************
                        !*****************************************************************************************

                        !-----------------------------------------------------------
                        ! STATE    : 40, Previous state was STATE 08 or STATE 10,
                        !            SLA+R have been transmitted, ACK received.
                        ! ACTION   : DATA will be received, ACK returned.
                        !-----------------------------------------------------------
                        .sect      mts40
                        .base      0x140

0140    75D8C5                              mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                    ! clr  STA, STO, SI  set  AA
0143    D0D0                                pop   psw
        32                                  reti

                        !-----------------------------------------------------------
                        ! STATE    : 48, SLA+R have been transmitted, NOT ACK received.
                        ! ACTION   : STOP condition will be generated.
                        !-----------------------------------------------------------
                        .sect      mts48
                        .base      0x148

0148    75D8D5          STOP:               mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                                    ! set  STO, clr  SI
014B    D0D0                                pop   psw
014D    32                                  reti

                        !-----------------------------------------------------------
                        ! STATE    : 50, DATA have been received, ACK returned.
                        ! ACTION   : Read DATA of S1DAT.
                        !            DATA will be received, if it is last DATA
                                     then NOT ACK will be returned else ACK will be returned.
                        !-----------------------------------------------------------
                        .sect      mrs50
                        .base      0x150

0150    75D018                              mov   psw,#SELRB3
0153    A6DA                                mov   @r0,S1DAT                 ! Read received DATA
0155    01C0                                ajmp  REC1

                        .sect      mrs50s
                        .base      0xc0

00C0    D55205          REC1:               djnz  NUMBYTMST,NOTLDAT2
00C3    75D8C1                              mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_NOTAA_CR0
                                                                    ! clr  SI,AA
00C6    8003                                sjmp  RETmr
00C8    75D8C5          NOTLDAT2:           mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                    ! clr  SI, set  AA
00CB    08              RETmr:              inc   r0
00CC    D0D0                                pop   psw
00CE    32                                  reti

                        !-----------------------------------------------------------
                        ! STATE    : 58, DATA have been received, NOT ACK returned.
                        ! ACTION   : Read DATA of S1DAT and generate a STOP condition.
                        !-----------------------------------------------------------
                        .sect      mrs58
                        .base      0x158

0158    75D018                              mov   psw,#SELRB3
015B    A6DA                                mov   @R0,S1DAT
015D    80E9                                sjmp  STOP
```

```
                        !*************************************************************************************************
                        !*************************************************************************************************
                        ! SLAVE RECEIVER STATE SERVICE ROUTINES
                        !*************************************************************************************************
                        !*************************************************************************************************

                        !-------------------------------------------------------------
                        ! STATE    : 60, Own SLA+W have been received, ACK returned.
                        ! ACTION   : DATA will be received and ACK returned.
                        !-------------------------------------------------------------
                        .sect      srs60
                        .base      0x160
0160    75D8C5                              mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                          ! clr  SI, set  AA
0163    75D018                              mov   psw,#SELRB3
0166    01D0                                ajmp  INITSRD
                        .sect      insrd
                        .base      0xd0

00D0    7840            INITSRD:            mov   r0,#SRD
00D2    7908                                mov   r1,#8
00D4    D0D0                                pop   psw
00D6    32                                  reti

                        !-------------------------------------------------------------
                        ! STATE    : 68, Arbitration lost in SLA and R/W as MST
                        !            Own SLA+W have been received, ACK returned
                        ! ACTION   : DATA will be received and ACK returned.
                        !            STA is set to restart MST mode after the bus is free again.
                        !-------------------------------------------------------------
                        .sect      srs68
                        .base      0x168
0168    75D8E5                              mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
016B    75D018                              mov   psw,#SELRB3
016E    01D0                                ajmp  INITSRD

                        !-------------------------------------------------------------
                        ! STATE    : 70, General call has been received, ACK returned.
                        ! ACTION   : DATA will be received and ACK returned.
                        !-------------------------------------------------------------
                        .sect      srs70
                        .base      0x170
0170    75D8C5                              mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                          ! clr  SI, set  AA
0173    75D018                              mov   psw,#SELRB3             ! Initialize SRD counter
0176    01D0                                ajmp  initsrd

                        !-------------------------------------------------------------
                        ! STATE    : 78, Arbitration lost in SLA+R/W as MST.
                        !            General call has been received, ACK returned.
                        ! ACTION   : DATA will be received and ACK returned.
                        !            STA is set to restart MST mode after the bus is free again.
                        !-------------------------------------------------------------
                        .sect      srs78
                        .base      0x178
0178    75D8E5                              mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
017B    75D018                              mov   psw,#SELRB3             ! Initialize SRD counter
017E    01D0                                ajmp  INITSRD
```

```
                        !- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        ! STATE    : 80, Previously addressed with own SLA. DATA received, ACK returned.
                        ! ACTION   : Read DATA.
                        !            IF received DATA was the last
                        !            THEN superfluous DATA will be received  and NOT ACK returned
                                     ELSE next DATA will be received and ACK returned.
                        !- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        .sect      srs80
                        .base      0x180

0180    75D018                                 mov   psw,#SELRB3
0183    A6DA                                   mov   @r0,S1DAT                    ! Read received DATA
0185    01D8                                   ajmp  REC2

                        .sect      srs80s
                        .base      0xd8

00D8    D906            REC2:                  djnz  r1,NOTLDAT3
00DA    75D8C1          LDAT:                  mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_NOTAA_CR0
                                                                                  ! clr  SI,AA
00DD    D0D0                                   pop   psw
00DF    32                                     reti
00E0    75D8C5          NOTLDAT3:              mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                                  ! clr  SI, set  AA
00E3    08                                     inc   r0
00E4    D0D0            RETsr:                 pop   psw
00E6    32                                     reti

                        !- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        ! STATE    : 88, Previously addressed with own SLA. DATA received NOT ACK returned.
                        ! ACTION   : No save of DATA, Enter NOT addressed SLV mode.
                        !            Recognition of own SLA. General call recognized, if S1ADR. 0–1.
                        !- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        .sect      srs88
                        .base      0x188

0188    75D8C5                                 mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                                  ! clr  SI, set  AA
018B    01E4                                   ajmp  RETsr

                        !- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        ! STATE    : 90, Previously addressed with general call.
                        !            DATA has been received, ACK has been returned.
                        ! ACTION   : Read DATA.
                                     After General call only one byte will be received with ACK
                        !            the second DATA will be received with NOT ACK.
                        !            DATA will be received and NOT ACK returned.
                        !- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        .sect      srs90
                        .base      0x190

0190    75D018                                 mov   psw,#SELRB3
0193    A6DA                                   mov   @r0,S1DAT                    ! Read received DATA
0195    01DA                                   ajmp  LDAT
                        !- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        ! STATE    : 98, Previously addressed with general call.
                        !            DATA has been received, NOT ACK has been returned.
                        ! ACTION   : No save of DATA, Enter NOT addressed SLV mode.
                                     Recognition of own SLA. General call recognized, if S1ADR. 0–1.
                        !- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        .sect      srs98
                        .base      0x198

0198    75D8C5                                 mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                                  ! clr  SI, set  AA
019B    D0D0                                   pop   psw
019D    32                                     reti
```

```
                        !– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –
                        ! STATE    : A0, A STOP condition or repeated START has been received,
                        !              while still addressed as SLV/REC or SLV/TRX.
                        ! ACTION   : No save of DATA, Enter NOT addressed SLV mode.
                        !              Recognition of own SLA. General call recognized, if S1ADR. 0–1.
                        !– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –
                        .sect      srsA0
                        .base      0x1a0

01A0    75D8C5                                mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                                  ! clr  SI, set  AA
01A3    D0D0                                  pop   psw
01A5    32                                    reti



                        !**********************************************************************************************************
                        !**********************************************************************************************************
                        ! SLAVE TRANSMITTER STATE SERVICE ROUTINES
                        !**********************************************************************************************************
                        !**********************************************************************************************************

                        !– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –
                        ! STATE    : A8, Own SLA+R received, ACK returned.
                        ! ACTION   : DATA will be transmitted, A bit received.
                        !– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –
                        .sect      stsa8
                        .base      0x1a8

01A8    8548DA                                mov   S1DAT,STD                   ! load DATA in S1DAT
01AB    75D8C5                                mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                                  ! clr  SI, set  AA
01AE    01E8                                  ajmp  INITBASE2

                        .sect      ibase2
                        .base      0xe8
00E8    75D018          INITBASE2:            mov   psw,#SELRB3
00EB    7948                                  mov   r1, #STD
00ED    09                                    inc   r1
00EE    D0D0                                  pop   psw
00F0    32                                    reti

                        !– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –
                        ! STATE    : B0, Arbitration lost in SLA and R/W as MST.  Own SLA+R received, ACK returned.
                        ! ACTION   : DATA will be transmitted, A bit received.
                        !              STA is set to restart MST mode after the bus is free again.
                        !– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –
                        .sect      stsb0
                        .base      0x1b0

01B0    8548DA                                mov   S1DAT,STD                   ! load DATA in S1DAT
01B3    75D8E5                                mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
01B6    01E8                                  ajmp  INITBASE2
```

```
                        !-------------------------------------------------------
                        ! STATE    : B8, DATA has been transmitted, ACK received.
                        ! ACTION   : DATA will be transmitted, ACK bit is received.
                        !-------------------------------------------------------
                        .sect      stsb8
                        .base      0x1b8
01B8    75D018                                mov   psw,#SELRB3
01BB    87DA                                  mov   S1DAT,@r1
01BD    01F8                                  ajmp  SCON

                        .sect      scn
                        .base      0xf8

00F8    75D8C5          SCON:                 mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                        ! clr  SI, set  AA
00FB    09                                    inc   r1
00FC    D0D0                                  pop   psw
00FE    32                                    reti
                        !-------------------------------------------------------
                        ! STATE    : C0, DATA has been transmitted, NOT ACK received.
                        ! ACTION   : Enter not addressed SLV mode.
                        !-------------------------------------------------------
                        .sect      stsc0
                        .base      0x1c0
01C0    75D8C5                                mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                        ! clr  SI, set  AA
01C3    D0D0                                  pop   psw
01C5    32                                    reti

                        !-------------------------------------------------------
                        ! STATE    : C8, Last DATA has been transmitted (AA=0), ACK received.
                        ! ACTION   : Enter not addressed SLV mode.
                        !-------------------------------------------------------
                        .sect      stsc8
                        .base      0x1c8
01C8    75D8C5                                mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                        ! clr  SI, set  AA
01CB    D0D0                                  pop   psw
01CD    32                                    reti


                        !*******************************************************************************************************
                        !*******************************************************************************************************
                        ! END OF SI01 INTERRUPT ROUTINE
                        !*******************************************************************************************************
                        !*******************************************************************************************************
```

**Reset Circuitry**

The reset circuitry for the 8XC552 is connected to the reset pin RST. A Schmitt trigger is used at the input for noise rejection (see Figure 25). The output of the Schmitt trigger is sampled by the reset circuitry every machine cycle.

A reset is accomplished by holding the RST pin HIGH for at least two machine cycles (24 oscillator periods) while the oscillator is running. The CPU responds by executing an internal reset. During reset, ALE and PSEN output a HIGH level. In order to perform a correct reset, this level must not be affected by external elements. The RST line can also be pulled HIGH internally by a pull-up transistor activated by the watchdog timer T3. The length of the output pulse from T3 is 3 machine cycles. A pulse of such short duration is necessary in order to recover from a processor or system fault as fast as possible.

Note that the short reset pulse from Timer T3 cannot discharge the power-on reset capacitor (see Figure 26). Consequently, when the watchdog timer is also used to set external devices, this capacitor arrangement should not be connected to the RST pin, and a different circuit should be used to perform the power-on reset operation. A timer T3 overflow, if enabled, will force a reset condition to the 8XC552 by an internal connection, whether the output RST is tied LOW or not.

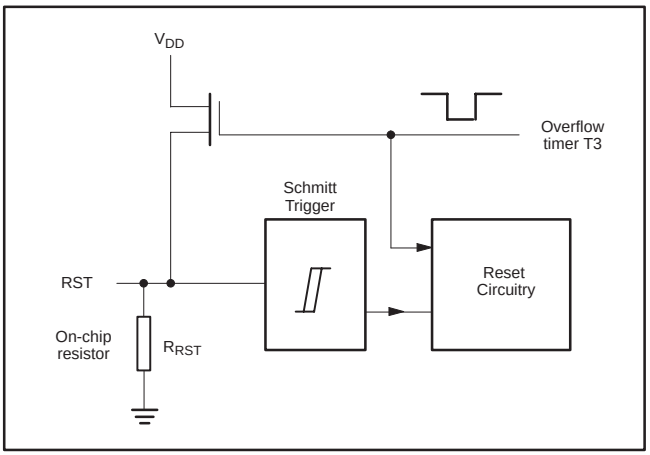



**Figure 25. On-Chip Reset Configuration**

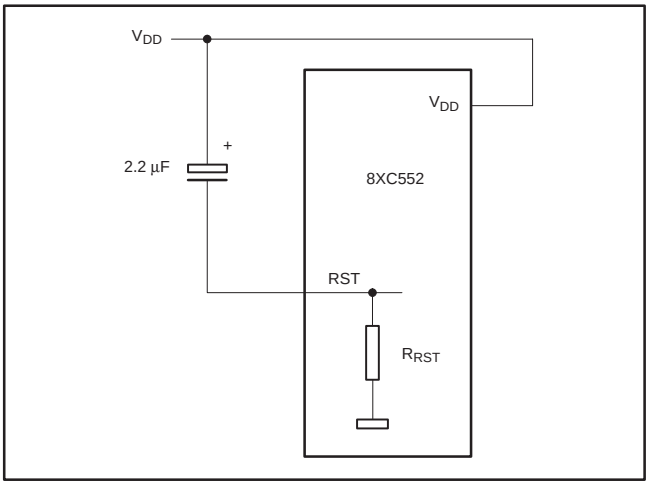



**Figure 26. Power-On Reset**

The internal reset is executed during the second cycle in which RST is HIGH and is repeated every cycle until RST goes low. It leaves the internal registers as follows:

| RESGISTER | CONTENT |
|---|---|
| ACC | 0000 0000 |
| ADCON | xx00 0000 |
| ADCH | xxxx xxxx |
| B | 0000 0000 |
| CML0-CML2 | 0000 0000 |
| CMH0-CMH2 | 0000 0000 |
| CTCON | 0000 0000 |
| CTL0-CTL3 | xxxx xxxx |
| CTH0-CTH3 | xxxx xxxx |
| DPL | 0000 0000 |
| DPH | 0000 0000 |
| IEN0 | 0000 0000 |
| IEN1 | 0000 0000 |
| IP0 | 0000 0000 |
| IP1 | 0000 0000 |
| PCH | 0000 0000 |
| PCL | 0000 0000 |
| PCON | 0xx0 0000 |
| PSW | 0000 0000 |
| PWM0 | 0000 0000 |
| PWM1 | 0000 0000 |
| PWMP | 0000 0000 |
| P0-P4 | 1111 1111 |
| PS | xxxx xxxx |
| RTE | 0000 0000 |
| S0BUF | xxxx xxxx |
| S0CON | 0000 0000 |
| S1ADR | 0000 0000 |
| S1CON | 0000 0000 |
| S1DAT | 0000 0000 |
| S1STA | 1111 1000 |
| SP | 0000 0111 |
| STE | 1100 0000 |
| TCON | 0000 0000 |
| TH0, TH1 | 0000 0000 |
| TMH2 | 0000 0000 |
| TL0, TL1 | 0000 0000 |
| TML2 | 0000 0000 |
| TMOD | 0000 0000 |
| TM2CON | 0000 0000 |
| TM2IR | 0000 0000 |
| T3 | 0000 0000 |

The internal RAM is not affected by reset. At power-on, the RAM content is indeterminate.

**Interrupts**

The 8XC552 has fifteen interrupt sources, each of which can be assigned one of two priority levels, as shown in Figure 27. The five interrupt sources common to the 80C51 are the external interrupts ($\overline{INT0}$ and $\overline{INT1}$), the timer 0 and timer 1 interrupts (IT0 and IT1), and the serial I/O interrupt (RI or TI). In the 8XC552, the standard serial interrupt is called SIO0. Since the subsystems which create these interrupts are identical on both parts, their functionality is likewise identical. The only differences are the locations of the enable and priority register configurations and the priority structure. This is detailed below along with the specifics of the interrupts unique to the 8XC552.

The eight Timer T2 interrupts are generated by flags CTI0-CT13, CMI0-CMI2, and by the logical OR of flags T2OV and T2BO. Flags CTI0 to CT13 are set by input signals CT0I to CT3i. Flags CMI0 to CMI2 are set when a match occurs between Timer T2 and the compare registers CM0, CM1, and CM2. When an 8-bit or 16-bit overflow occurs, flags T2BO and T2OV are set, respectively. These nine flags are not cleared by hardware and must be reset by software to avoid recurring interrupts.

The ADC interrupt is generated by the ADCI flag in the ADC control register (ADCON). This flag is set when an ADC conversion result is ready to be read. ADCI is not cleared by hardware and must be reset by software to avoid recurring interrupts.

The SIO1 ($I^2C$) interrupt is generated by the SI flag in the SIO1 control register (S1CON). This flag is set when S1STA is loaded with a valid status code.

The ADCI flag may be reset by software. It cannot be set by software. All other flags that generate interrupts may be set or cleared by software, and the effect is the same as setting or resetting the flags by hardware. Thus, interrupts may be generated by software and pending interrupts can be canceled by software.

**Interrupt Enable Registers:** Each interrupt source can be individually enabled or disabled by setting or clearing a bit in the interrupt enable special function registers IEN0 and IEN1. All interrupt sources can also be globally enabled or disabled by setting or clearing bit EA in IEN0. The interrupt enable registers are described in Figures 28 and 29.

**Interrupt Priority Structure:** Each interrupt source can be assigned one of two priority levels. Interrupt priority levels are defined by the interrupt priority special function registers IP0 and IP1. IP0 and IP1 are described in Figures 30 and 31.

Interrupt priority levels are as follows:
“0”—low priority
“1”—high priority

A low priority interrupt may be interrupted by a high priority interrupt. A high priority interrupt cannot be interrupted by any other interrupt source. If two requests of different priority occur simultaneously, the high priority level request is serviced. If requests of the same priority 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. This second priority structure is shown in Table 8.

The above Priority Within Level structure is only used when there are simultaneous requests of the same priority level.

**Interrupt Handling:** The interrupt sources are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. If one of the flags was in a set condition at S5P2 of the previous machine cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of higher or equal priority level is already in progress.
2. The current machine cycle is not the final cycle in the execution of the instruction in progress. (No interrupt request will be serviced until the instruction in progress is completed.)
3. The instruction in progress is RETI or any access to the interrupt priority or interrupt enable registers. (No interrupt will be serviced after RETI or after a read or write to IP0, IP1, IE0, or IE1 until at least one other instruction has been subsequently executed.)

The polling cycle is repeated with every machine cycle, and the values polled are the values present at S5P2 of the previous machine cycle. Note that if an interrupt flag is active but is not being responded to because of one of the above conditions, and if the flag is inactive when the blocking condition is removed, then the blocked interrupt will not be serviced. Thus, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate service routine. In some cases it also clears the flag which generated the interrupt, and in others it does not. It clears the Timer 0, Timer 1, and external interrupt flags. An external interrupt flag (IEO or IE1) is cleared only if it was transition-activated. All other interrupt flags are not cleared by hardware and must be cleared by the software. The LCALL pushes the contents of the program counter on to the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to as shown in Table 9.

Execution proceeds from the vector address until the RETI instruction is encountered. The RETI instruction clears the “priority level active” flip-flop that was set when this interrupt was acknowledged. It then pops the top two bytes from the stack and reloads the program counter. Execution of the interrupted program continues from where it was interrupted.

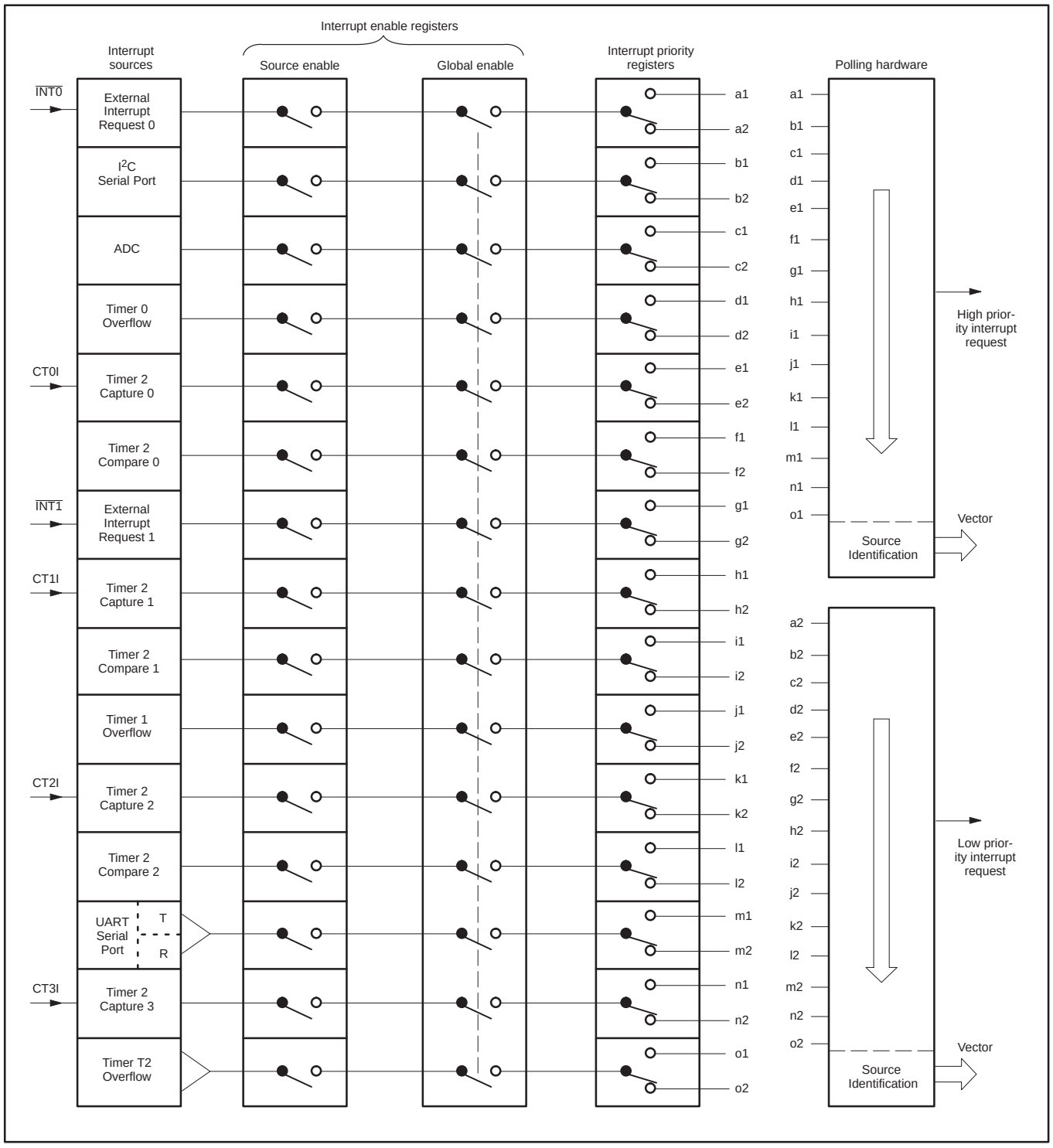



**Figure 27. The Interrupt System**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **IEN0 (A8H)** | EA | EAD | ES1 | ES0 | ET1 | EX1 | ET0 | EX0 |
| | (MSB) | | | | | | | (LSB) |

| BIT | SYMBOL | FUNCTION |
|---|---|---|
| IEN0.7 | EA | Global enable/disable control<br>0 = No interrupt is enabled<br>1 = Any individually enabled interrupt will be accepted |
| IEN0.6 | EAD | Eanble ADC interrupt |
| IEN0.5 | ES1 | Enable SIO1 ($I^2C$) interrupt |
| IEN0.4 | ES0 | Enable SIO0 (UART) interrupt |
| IEN0.3 | ET1 | Enable Timer 1 interrupt |
| IEN0.2 | EX1 | Enable External interrupt 1 |
| IEN0.1 | ET0 | Enable Timer 0 interrupt |
| IEN0.0 | EX0 | Enable External interrupt 0 |

SU00762

**Figure 28. Interrupt Enable Register (IEN0)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **IEN1 (E8H)** | ET2 | ECM2 | ECM1 | ECM0 | ECT3 | ECT2 | ECT1 | ECT0 |
| | (MSB) | | | | | | | (LSB) |

| BIT | SYMBOL | FUNCTION |
|---|---|---|
| IEN1.7 | ET2 | Enable Timer T2 overflow interrupt(s) |
| IEN1.6 | ECM2 | Enable T2 Comparator 2 interrupt |
| IEN1.5 | ECM1 | Enable T2 Comparator 1 interrupt |
| IEN1.4 | ECM0 | Enable T2 Comparator 0 interrupt |
| IEN1.3 | ECT3 | Enable T2 Capture register 3 interrupt |
| IEN1.2 | ECT2 | Enable T2 Capture register 2 interrupt |
| IEN1.1 | ECT1 | Enable T2 Capture register 1 interrupt |
| IEN1.0 | ECT0 | Enable T2 Capture register 0 interrupt |

SU00755

In all cases, if the enable bit is 0, then the interrupt is disabled, and if the enable bit is 1, then the interrupt is enabled.

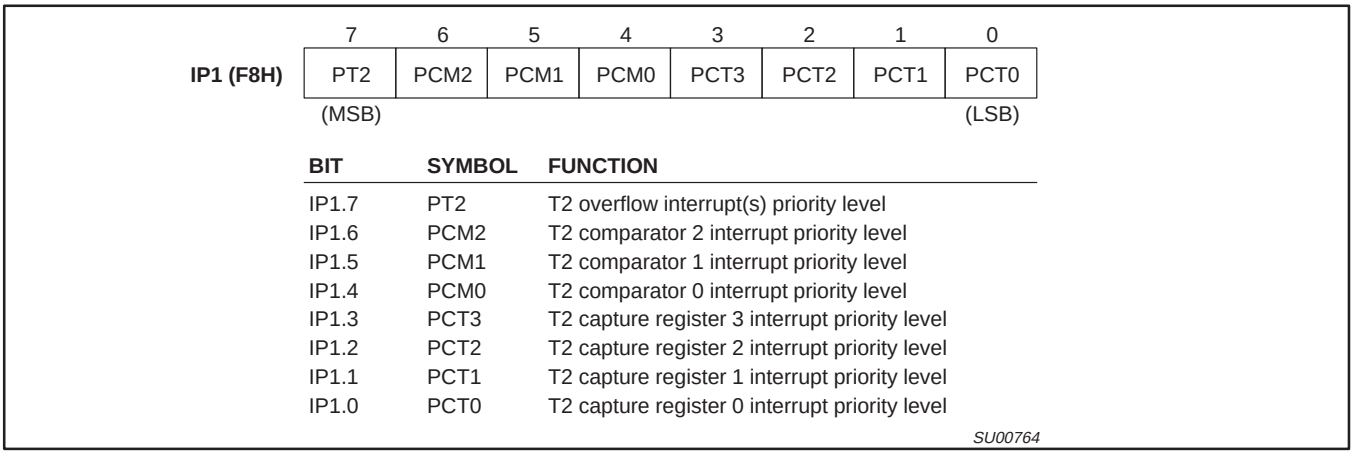
**Figure 29. Interrupt Enable Register (IEN1)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **IP0 (B8H)** | – | PAD | PS1 | PS0 | PT1 | PX1 | PT0 | PX0 |
| | (MSB) | | | | | | | (LSB) |

| BIT | SYMBOL | FUNCTION |
|---|---|---|
| IP0.7 | – | Unused |
| IP0.6 | PAD | ADC interrupt priority level |
| IP0.5 | PS1 | SIO1 ($I^2C$) interrupt priority level |
| IP0.4 | PS0 | SIO0 (UART) interrupt priority level |
| IP0.3 | PT1 | Timer 1 interrupt priority level |
| IP0.2 | PX1 | External interrupt 1 priority level |
| IP0.1 | PT0 | Timer 0 interrupt priority level |
| IP0.0 | PX0 | External interrupt 0 priority level |

SU00763

**Figure 30. Interrupt Priority Register (IP0)**




| BIT | SYMBOL | FUNCTION |
|---|---|---|
| IP1.7 | PT2 | T2 overflow interrupt(s) priority level |
| IP1.6 | PCM2 | T2 comparator 2 interrupt priority level |
| IP1.5 | PCM1 | T2 comparator 1 interrupt priority level |
| IP1.4 | PCM0 | T2 comparator 0 interrupt priority level |
| IP1.3 | PCT3 | T2 capture register 3 interrupt priority level |
| IP1.2 | PCT2 | T2 capture register 2 interrupt priority level |
| IP1.1 | PCT1 | T2 capture register 1 interrupt priority level |
| IP1.0 | PCT0 | T2 capture register 0 interrupt priority level |

SU00764

**Figure 31. Interrupt Priority Register (IP1)**

**Table 8. Interrupt Priority Structure**

| SOURCE | NAME | PRIORITY WITHIN LEVEL |
|---|---|---|
| | | (highest) |
| External interrupt 0 | X0 | ↑ |
| SIO1 ($I^2C$) | S1 | |
| ADC completion | ADC | |
| Timer 0 overflow | T0 | |
| T2 capture 0 | CT0 | |
| T2 compare 0 | CM0 | |
| External interrupt 1 | X1 | |
| T2 capture 1 | CT1 | |
| T2 compare 1 | CM1 | |
| Timer 1 overflow | T1 | |
| T2 capture 2 | CT2 | |
| T2 compare 2 | CM2 | |
| SIO0 (UART) | S0 | |
| T2 capture 3 | CT3 | |
| Timer T2 overflow | T2 | ↓ |
| | | (lowest) |

**Table 9. Interrupt Vector Addresses**

| SOURCE | NAME | VECTOR ADDRESS |
|---|---|---|
| External interrupt 0 | X0 | 0003H |
| Timer 0 overflow | T0 | 000BH |
| External interrupt 1 | X1 | 0013H |
| Timer 1 overflow | T1 | 001BH |
| SIO0 (UART) | S0 | 0023H |
| SIO1 ($I^2C$) | S1 | 002BH |
| T2 capture 0 | CT0 | 0033H |
| T2 capture 1 | CT1 | 003BH |
| T2 capture 2 | CT2 | 0043H |
| T2 capture 3 | CT3 | 004BH |
| ADC completion | ADC | 0053H |
| T2 compare 0 | CM0 | 005BH |
| T2 compare 1 | CM1 | 0063H |
| T2 compare 2 | CM2 | 006BH |
| T2 overflow | T2 | 0073H |

**I/O Port Structure**
The 8XC552 has six 8-bit ports. Each port consists of a latch (special function registers P0 to P5), an input buffer, and an output driver (port 0 to 4 only). Ports 0-3 are the same as in the 80C51, with the exception of the additional functions of port 1. The parallel I/O function of port 4 is equal to that of ports 1, 2, and 3. Port 5 may be used as an input port only.

Figure 32 shows the bit latch and I/O buffer functional diagrams of the unique 8XC552 ports. A bit latch corresponds to one bit in a port's SFR and is represented as a D type flip-flop. A "write to latch" signal from the CPU latches a bit from the internal bus and a "read latch" signal from the CPU places the Q output of the flip-flop on the internal bus. A "read pin" signal from the CPU places the actual port pin level on the internal bus. Some instructions that read a port read the actual port pin levels, and other instructions read the latch (SFR) contents.

**Port 1 Operation**
Port 1 operates the same as it does in the 8051 with the exception of port lines P1.6 and P1.7, which may be selected as the SCL and SDA lines of serial port SIO1 ($I^2C$). Because the $I^2C$ bus may be active while the device is disconnected from $V_{DD}$, these pins are provided with open drain drivers. Therefore pins P1.6 and P1.7 do not have internal pull-ups.

**Port 5 Operation**
Port 5 may be used to input up to 8 analog signals to the ADC. Unused ADC inputs may be used to input digital inputs. These inputs have an inherent hysteresis to prevent the input logic from drawing excessive current from the power lines when driven by analog signals. Channel to channel crosstalk (Ct) should be taken into consideration when both analog and digital signals are simultaneously input to Port 5 (see, D.C. characteristics in data sheet).

Port 5 is not bidirectional and may not be configured as an output port. All six ports are multifunctional, and their alternate functions are listed in Table 10. A more detailed description of these features can be found in the relevant parts of this section.

**Pulse Width Modulated Outputs**
The 8XC552 contains two pulse width modulated output channels (see Figure 33). These channels generate pulses of programmable length and interval. The repetition frequency is defined by an 8-bit prescaler PWMP, which supplies the clock for the counter. The prescaler and counter are common to both PWM channels. The 8-bit counter counts modulo 255, i.e., from 0 to 254 inclusive. The value of the 8-bit counter is compared to the contents of two registers: PWM0 and PWM1. Provided the contents of either of these registers is greater than the counter value, the corresponding $\overline{PWM0}$ or $\overline{PWM1}$ output is set LOW. If the contents of these registers are equal to, or less than the counter value, the output will be HIGH. The pulse-width-ratio is therefore defined by the contents of the registers PWM0 and PWM1. The pulse-width-ratio is in the range of 0 to 1 and may be programmed in increments of 1/255.

Buffered PWM outputs may be used to drive DC motors. The rotation speed of the motor would be proportional to the contents of PWMn. The PWM outputs may also be configured as a dual DAC. In this application, the PWM outputs must be integrated using conventional operational amplifier circuitry. If the resulting output voltages have to be accurate, external buffers with their own analog supply should be used to buffer the PWM outputs before they are integrated. The repetition frequency $f_{PWM}$, at the PWMn outputs is give by:

$$f_{PWM} = \frac{f_{OSC}}{2 \times (1 + PWMP) \times 255}$$

This gives a repetition frequency range of 123Hz to 31.4kHz ($f_{OSC}$ = 16MHz). At fosc = 24MHz, the frequency range is 184Hz to 47.1Hz. By loading the PWM registers with either 00H or FFH, the PWM channels will output a constant HIGH or LOW level, respectively. Since the 8-bit counter counts modulo 255, it can never actually reach the value of the PWM registers when they are loaded with FFH.

When a compare register (PWM0 or PWM1) is loaded with a new value, the associated output is updated immediately. It does not have to wait until the end of the current counter period. Both $\overline{PWMn}$ output pins are driven by push-pull drivers. These pins are not used for any other purpose.

Prescaler frequency control register PWMP

| PWMP (FEH) | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | MSB | | | | | | | LSB |

PWMP.0-7 Prescaler division factor = PWMP + 1.

Reading PWMP gives the current reload value. The actual count of the prescaler cannot be read.

| PWM0 (FCH)<br>PWM1 (FDH) | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | MSB | | | | | | | LSB |

$$\text{PWM0/1.0-7\} Low/high ratio of } \overline{PWMn} = \frac{(PWMn)}{255 - (PWMn)}$$

**Analog-to-Digital Converter**
The analog input circuitry consists of an 8-input analog multiplexer and a 10-bit, straight binary, successive approximation ADC. The analog reference voltage and analog power supplies are connected via separate input pins. The conversion takes 50 machine cycles, i.e., 37.5µs at an oscillator frequency of 16MHz, 25µs at an oscillator frequency of 24MHz. Input voltage swing is from 0V to +5V. Because the internal DAC employs a ratiometric potentiometer, there are no discontinuities in the converter characteristic. Figure 34 shows a functional diagram of the analog input circuitry.

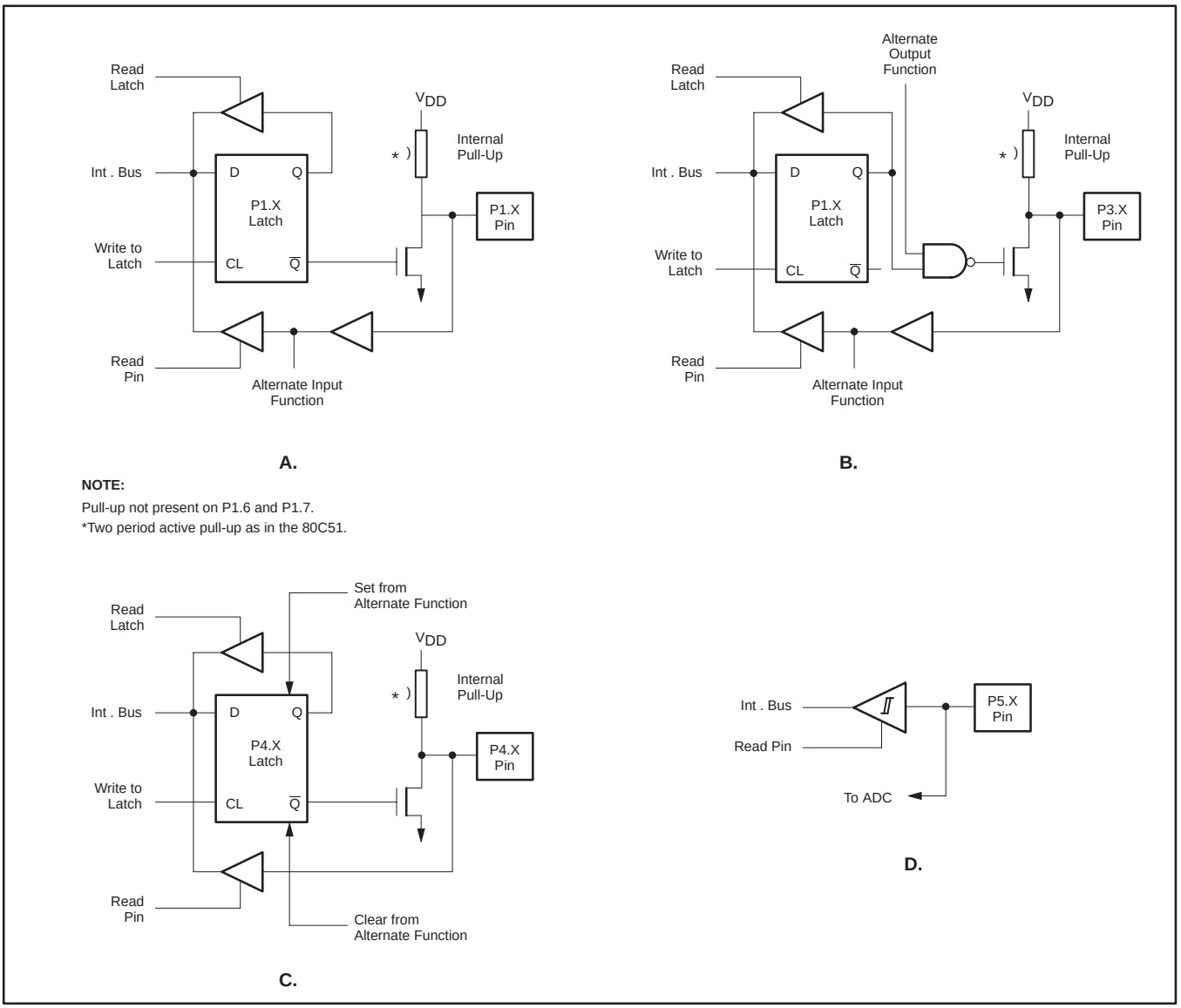



**Figure 32. Port Bit Latches and I/O Buffers**

**Table 10. Input/Output Ports**

| PORT PIN | ALTERNATE FUNCTION | |
|---|---|---|
| P0.0 | AD0 | Multiplexed lower order address/data bus used during external memory accesses |
| P0.1 | AD1 | |
| P0.2 | AD2 | |
| P0.3 | AD3 | |
| P0.4 | AD4 | |
| P0.5 | AD5 | |
| P0.6 | AD6 | |
| P0.7 | AD7 | |
| P1.0 | CT0I | Capture timiner input signals for timer T2 |
| P1.1 | CT1I | |
| P1.2 | CT2I | |
| P1.3 | CT3I | |
| P1.4 | T2 | T2 event input |
| P1.5 | RT2 | T2 timer reset signal. Rising edge triggered |
| P1.6 | SCL | Serial port clock line $I^2C$ bus |
| P1.7 | SDA | Serial port data line $I^2C$ bus |
| P2.0 | A8 | High order address byte used during external memory accesses |
| P2.1 | A9 | |
| P2.2 | A10 | |
| P2.3 | A11 | |
| P2.4 | A12 | |
| P2.5 | A13 | |
| P2.6 | A14 | |
| P2.7 | A15 | |
| P3.0 | RxD | Serial input port (UART) |
| P3.1 | TxD | Serial output port (UART) |
| P3.2 | $\overline{INT0}$ | External interrupt 0 |
| P3.3 | $\overline{INT1}$ | External interrupt 1 |
| P3.4 | T0 | Timer 0 external input |
| P3.5 | T1 | Timer 1 external input |
| P3.6 | $\overline{WR}$ | External data memory write strobe |
| P3.7 | $\overline{RD}$ | External data memory read strobe |
| P4.0 | CMSR0 | Timer T2: compare and set/reset outputs on a match with timer T2 |
| P4.1 | CMSR1 | |
| P4.2 | CMSR2 | |
| P4.3 | CMSR3 | |
| P4.4 | CMSR4 | |
| P4.5 | CMSR5 | Timer T2: compare and toggle outputs on a match with timer T2 |
| P4.6 | CMT0 | |
| P4.7 | CMT1 | |
| P5.0 | ADC0 | Eight analogue ADC inputs |
| P5.1 | ADC1 | |
| P5.2 | ADC2 | |
| P5.3 | ADC3 | |
| P5.4 | ADC4 | |
| P5.5 | ADC5 | |
| P5.6 | ADC6 | |
| P5.7 | ADC7 | |

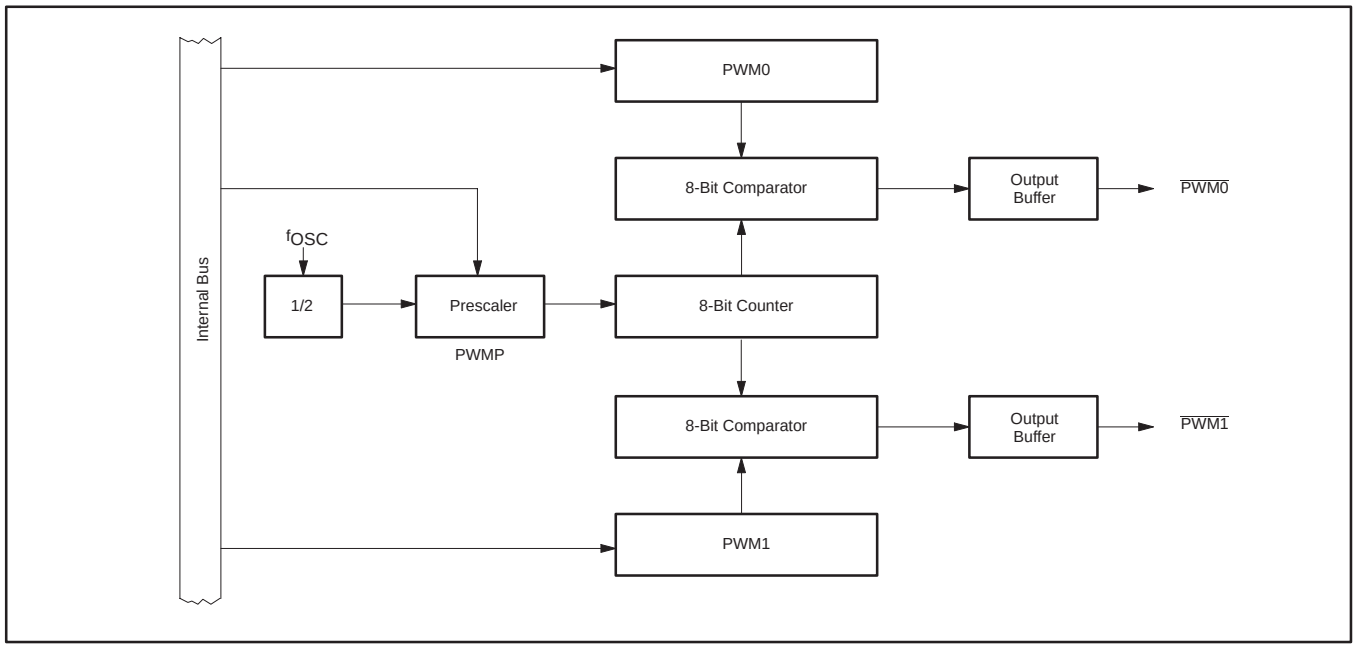



**Figure 33. Functional Diagram of Pulse Width Modulated Outputs**

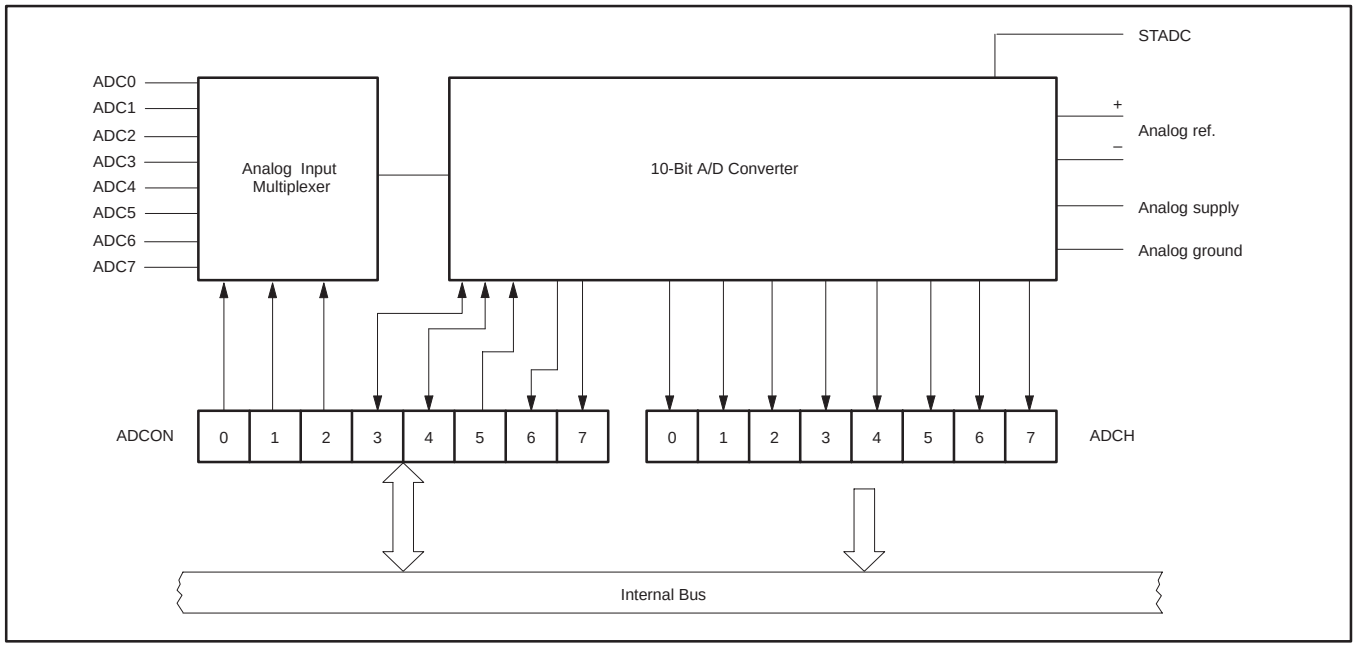



**Figure 34. Functional Diagram of Analog Input Circuitry**

**Analog-to-Digital Conversion:** Figure 35 shows the elements of a successive approximation (SA) ADC. The ADC contains a DAC which converts the contents of a successive approximation register to a voltage (VDAC) which is compared to the analog input voltage (Vin). The output of the comparator is fed to the successive approximation control logic which controls the successive approximation register. A conversion is initiated by setting ADCS in the ADCON register. ADCS can be set by software only or by either hardware or software.

The software only start mode is selected when control bit ADCON.5 (ADEX) = 0. A conversion is then started by setting control bit ADCON.3 (ADCS). The hardware or software start mode is selected when ADCON.5 = 1, and a conversion may be started by setting ADCON.3 as above or by applying a rising edge to external pin STADC. When a conversion is started by applying a rising edge, a low level must be applied to STADC for at least one machine cycle followed by a high level for at least one machine cycle.

The low-to-high transition of STADC is recognized at the end of a machine cycle, and the conversion commences at the beginning of the next cycle. When a conversion is initiated by software, the conversion starts at the beginning of the machine cycle which follows the instruction that sets ADCS. ADCS is actually implemented with two flip-flops: a command flip-flop which is affected by set operations, and a status flag which is accessed during read operations.

The next two machine cycles are used to initiate the converter. At the end of the first cycle, the ADCS status flag is set and a value of “1” will be returned if the ADCS flag is read while the conversion is in progress. Sampling of the analog input commences at the end of the second cycle.

During the next eight machine cycles, the voltage at the previously selected pin of port 5 is sampled, and this input voltage should be stable in order to obtain a useful sample. In any event, the input voltage slew rate must be less than 10V/ms in order to prevent an undefined result.

The successive approximation control logic first sets the most significant bit and clears all other bits in the successive approximation register (10 0000 0000B). The output of the DAC (50% full scale) is compared to the input voltage Vin. If the input voltage is greater than VDAC, then the bit remains set; otherwise it is cleared.

The successive approximation control logic now sets the next most significant bit (11 0000 0000B or 01 0000 0000B, depending on the previous result), and VDAC is compared to Vin again. If the input voltage is greater than VDAC, then the bit being tested remains set; otherwise the bit being tested is cleared. This process is repeated until all ten bits have been tested, at which stage the result of the conversion is held in the successive approximation register. Figure 36 shows a conversion flow chart. The bit pointer identifies the bit under test. The conversion takes four machine cycles per bit.

The end of the 10-bit conversion is flagged by control bit ADCON.4 (ADCI). The upper 8 bits of the result are held in special function register ADCH, and the two remaining bits are held in ADCON.7 (ADC.1) and ADCON.6 (ADC.0). The user may ignore the two least significant bits in ADCON and use the ADC as an 8-bit converter (8 upper bits in ADCH). In any event, the total actual conversion time is 50 machine cycles for the 8XC552 or 24 machine cycles for the 8XC562. ADCI will be set and the ADCS status flag will be reset 50 (or 24) cycles after the command flip-flop (ADCS) is set.

Control bits ADCON.0, ADCON.1, and ADCON.2 are used to control an analog multiplexer which selects one of eight analog channels (see Figure 37). An ADC conversion in progress is unaffected by an external or software ADC start. The result of a completed conversion remains unaffected provided ADCI = logic 1; a new ADC conversion already in progress is aborted when the idle or power-down mode is entered. The result of a completed conversion (ADCI = logic 1) remains unaffected when entering the idle mode.

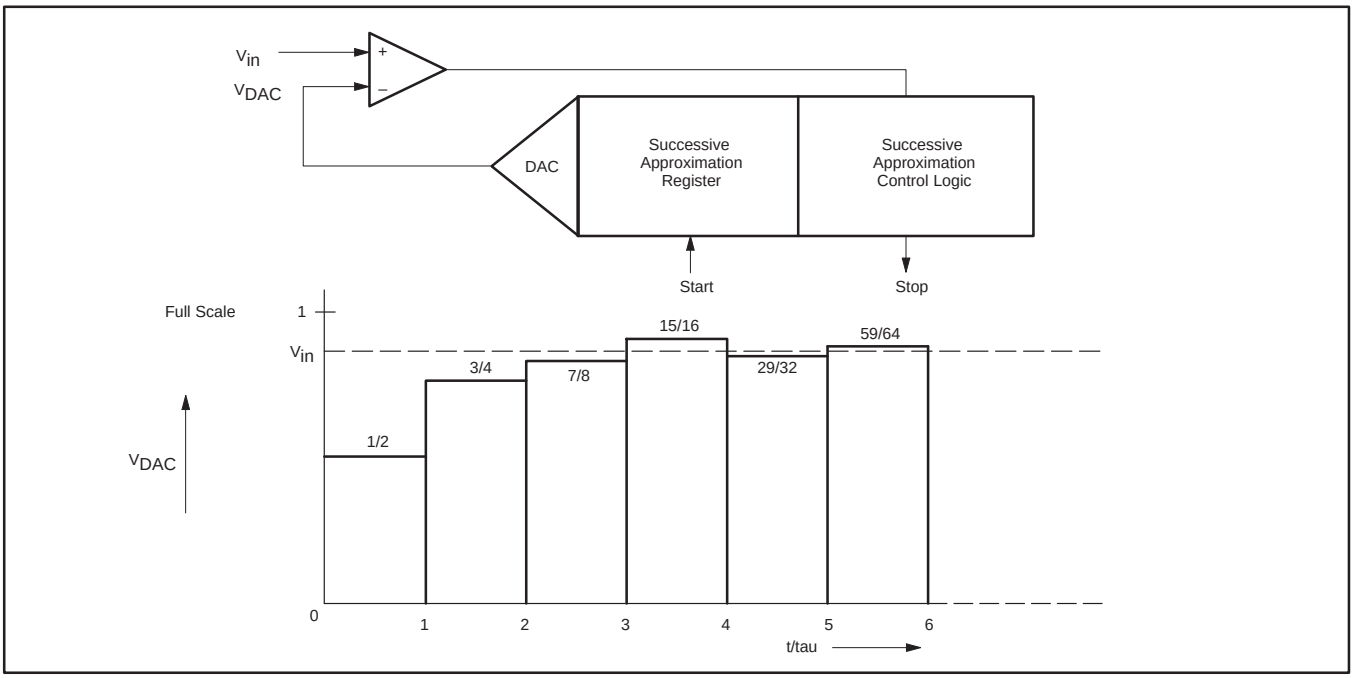



**Figure 35. Successive Approximation ADC**

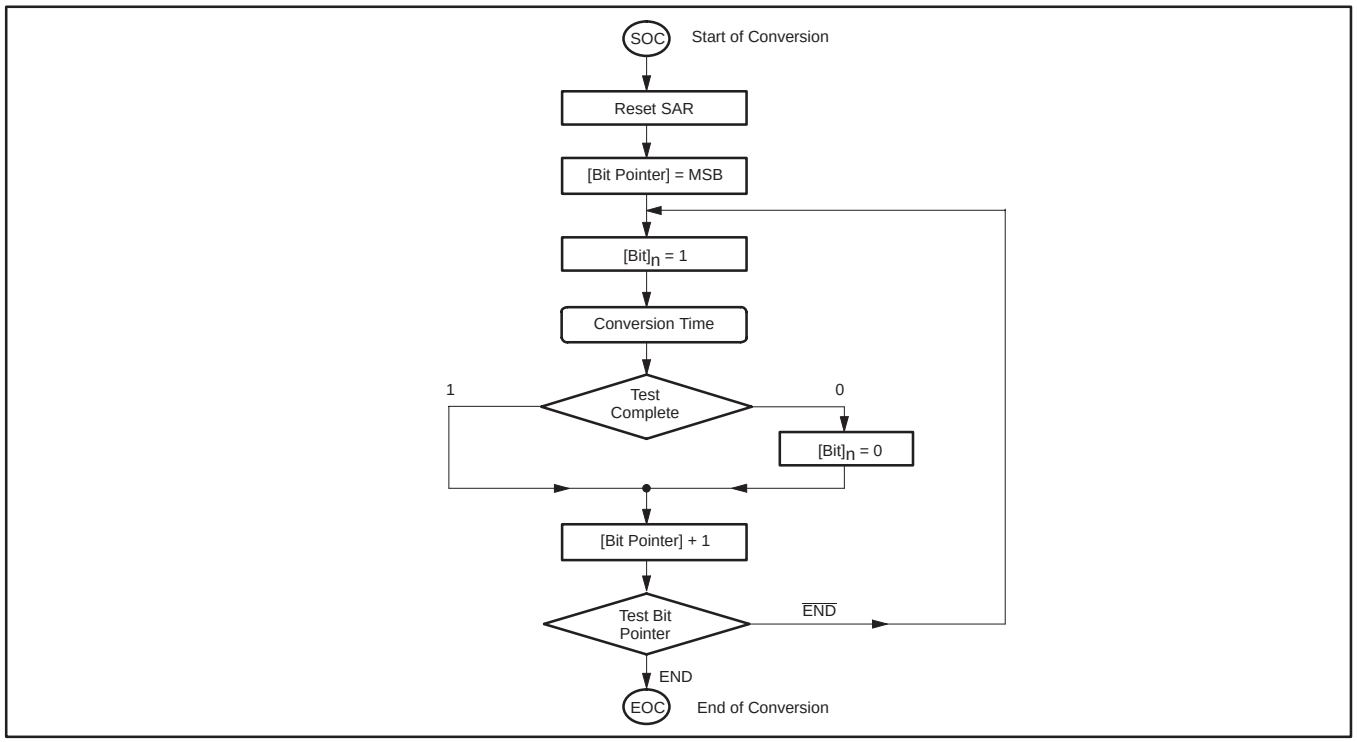



**Figure 36. A/D Conversion Flowchart**

ADCON (C5H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ADC.1 | ADC.0 | ADEX | ADCI | ADCS | AADR2 | AADR1 | AADR0 |
| (MSB) | | | | | | | (LSB) |

| Bit | Symbol | Function |
|---|---|---|
| ADCON.7 | ADC.1 | Bit 1 of ADC result |
| ADCON.6 | ADC.0 | Bit 0 of ADC result |
| ADCON.5 | ADEX | Enable external start of conversion by STADC<br>0 = Conversion can be started by software only (by setting ADCS)<br>1 = Conversion can be started by software or externally (by a rising edge on STADC) |
| ADCON.4 | ADCI | ADC interrupt flag: this flag is set when an A/D conversion result is ready to be read. An interrupt is invoked if it is enabled. The flag may be cleared by the interrupt service routine. While this flag is set, the ADC cannot start a new conversion. ADCI cannot be set by software. |
| ADCON.3 | ADCS | ADC start and status: setting this bit starts an A/D conversion. It may be set by software or by the external signal STADC. The ADC logic ensures that this signal is HIGH while the ADC is busy. On completion of the conversion, ADCS is reset immediately after the interrupt flag has been set. ADCS cannot be reset by software. A new conversion may not be started while either ADCS or ADCI is high. |

| ADCI | ADCS | ADC Status |
|---|---|---|
| 0 | 0 | ADC not busy; a conversion can be started |
| 0 | 1 | ADC busy; start of a new conversion is blocked |
| 1 | 0 | Conversion completed; start of a new conversion requires ADCI=0 |
| 1 | 1 | Conversion completed; start of a new conversion requires ADCI=0 |

If ADCI is cleared by software while ADCS is set at the same time, a new A/D conversion with the same channel number may be started. But it is recommended to reset ADCI ***before*** ADCS is set.

| | | |
|---|---|---|
| ADCON.2<br>ADCON.1<br>ADCON.0 | AADR2<br>AADR1<br>AADR0 | Analogue input select: this binary coded address selects one of the eight analogue port bits of P5 to be input to the converter. It can only be changed when ADCI and ADCS are both LOW. |

| AADR2 | AADR1 | AADR0 | Selected Analog Channel |
|---|---|---|---|
| 0 | 0 | 0 | ADC0 (P5.0) |
| 0 | 0 | 1 | ADC1 (P5.1) |
| 0 | 1 | 0 | ADC2 (P5.2) |
| 0 | 1 | 1 | ADC3 (P5.3) |
| 1 | 0 | 0 | ADC4 (P5.4) |
| 1 | 0 | 1 | ADC5 (P5.5) |
| 1 | 1 | 0 | ADC6 (P5.6) |
| 1 | 1 | 1 | ADC7 (P5.7) |

**Figure 37. ADC Control Register (ADCON)**

**ADC Resolution and Analog Supply:** Figure 38 shows how the ADC is realized. The ADC has its own supply pins ($AV_{DD}$ and $AV_{SS}$) and two pins (Vref+ and Vref–) connected to each end of the DAC's resistance-ladder. The ladder has 1023 equally spaced taps, separated by a resistance of "R". The first tap is located 0.5 x R above Vref–, and the last tap is located 1.5 x R below Vref+. This gives a total ladder resistance of 1024 x R. This structure ensures that the DAC is monotonic and results in a symmetrical quantization error as shown in Figure 40.

For input voltages between Vref– and (Vref–) + 1/2 LSB, the 10-bit result of an A/D conversion will be 00 0000 0000B = 000H. For input voltages between (Vref+) – 3/2 LSB and Vref+, the result of a conversion will be 11 1111 1111B = 3FFH. AVref+ and AVref– may be between $AV_{DD}$ + 0.2V and $AV_{SS}$ – 0.2V. AVref+ should be positive with respect to AVref–, and the input voltage (Vin) should be between AVref+ and AVref–. If the analog input voltage range is from 2V to 4V, then 10-bit resolution can be obtained over this range if AVref+ = 4V and AVref– = 2V.

The result can always be calculated from the following formula:

$$\text{Result} = 1024 \times \frac{V_{IN} - AV_{ref-}}{AV_{ref+} - AV_{ref-}}$$

**Power Reduction Modes**

The 8XC552 has two reduced power modes of operation: the idle mode and the power-down mode. These modes are entered by setting bits in the PCON special function register. When the 8XC552 enters the idle mode, the following functions are disabled:

| | |
|---|---|
| CPU | (halted) |
| Timer T2 | (halted and reset) |
| PWM0, PWM1 | (reset; outputs are high) |
| ADC | (conversion aborted if in progress). |

In idle mode, the following functions remain active:

Timer 0
Timer 1
Timer T3
SIO0 SIO1
External interrupts

When the 8XC552 enters the power-down mode, the oscillator is stopped. The power-down mode is entered by setting the PD bit in the PCON register. The PD bit can only be set if the $\overline{EW}$ input is tied HIGH.

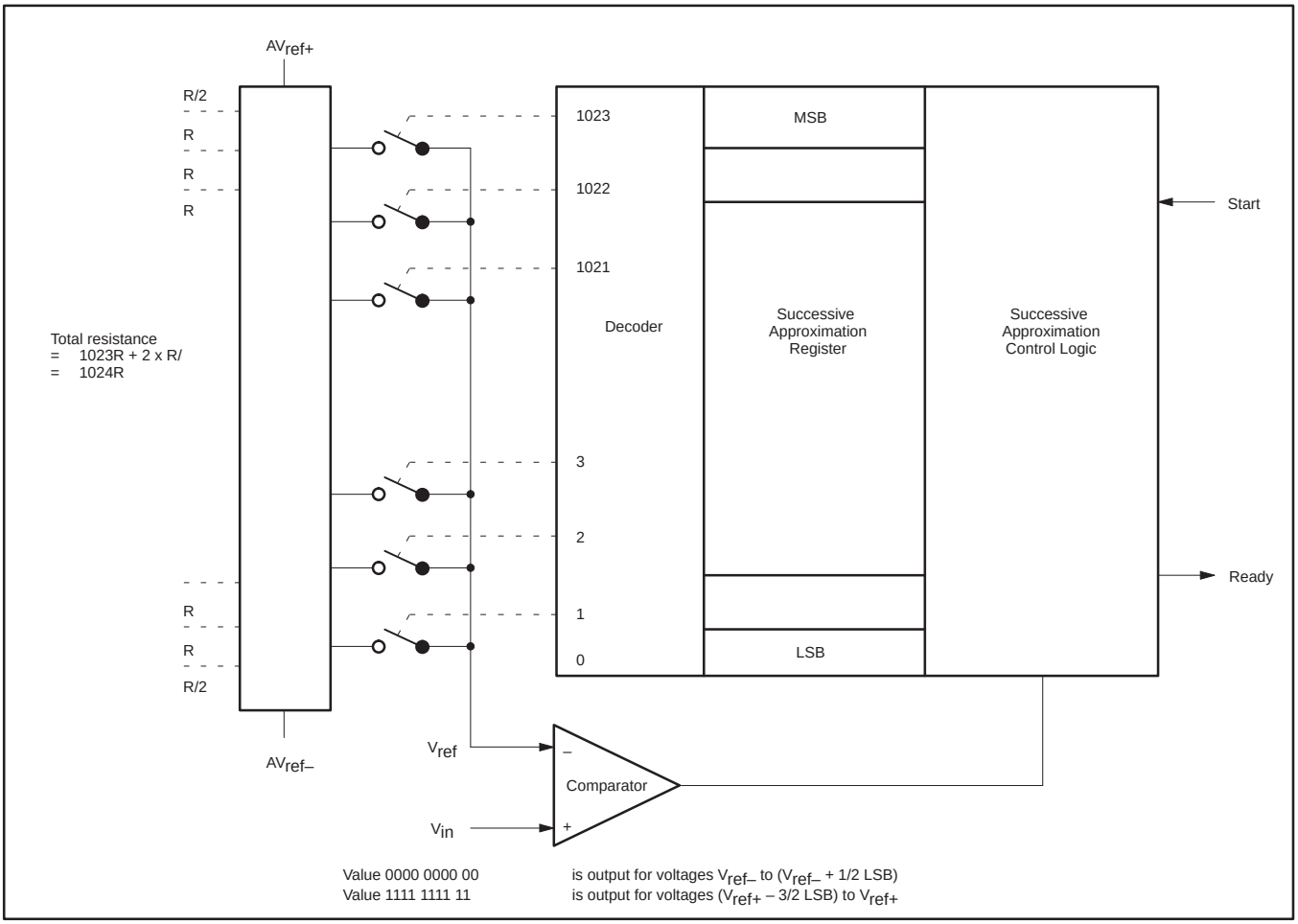
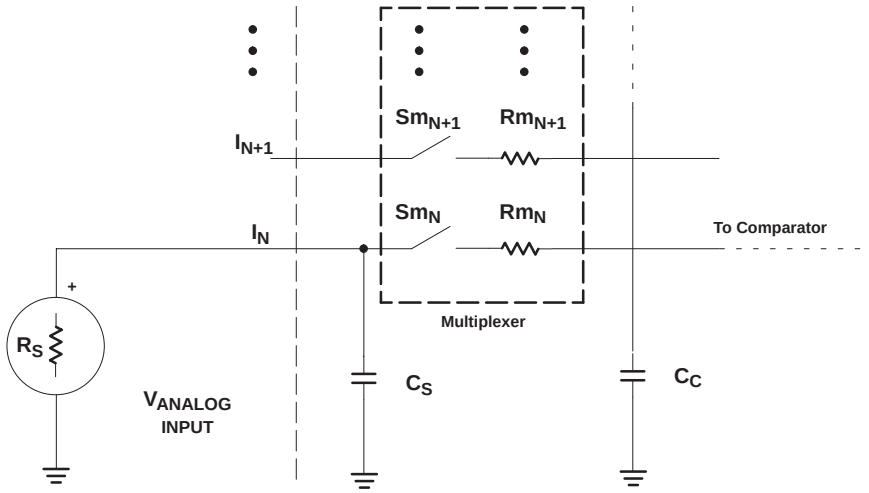



**Figure 38. ADC Realization**




Rm = 0.5 - 3 kohms
CS + CC = 15pF maximum
RS = Recommended < 9.6 kohms for 1 LSB @ 12MHz

**NOTE:**
Because the analog to digital converter has a sampled-data comparator, the input looks capacitive to a source. When a conversion is initiated, switch Sm closes for 8tcy (8µs @ 12MHz crystal frequency) during which time capacitance Cs + Cc is charged. It should be noted that the sampling causes the analog input to present a varying load to an analog source.

**Figure 39. A/D Input: Equivalent Circuit**

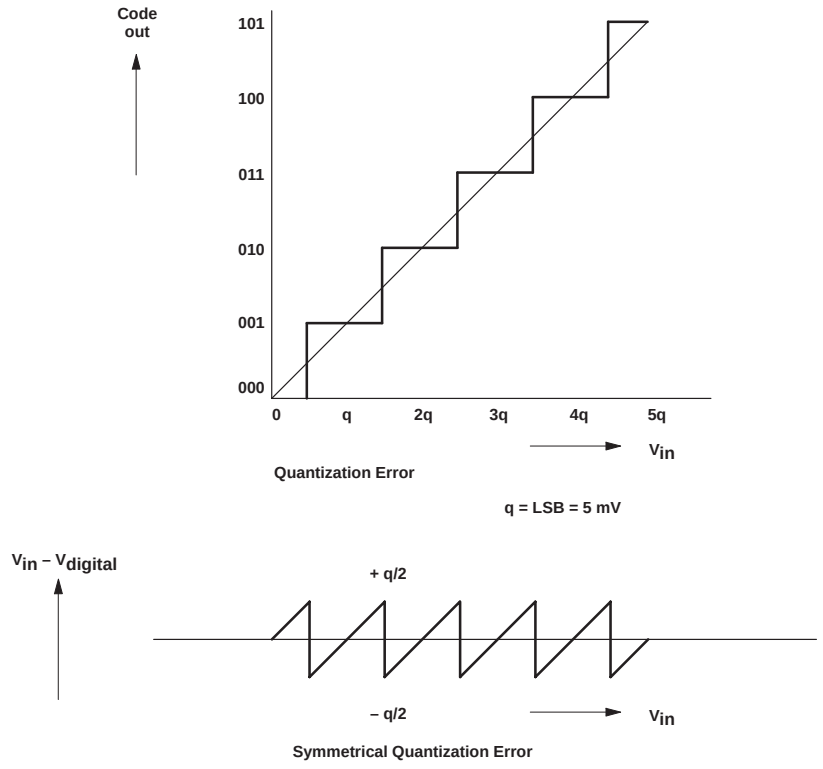



**Figure 40. Effective Conversion Characteristic**

**Power-Down Mode:** The instruction that sets PCON.1 will be the last instruction executed in the normal operating mode before the power-down mode is entered. In the power-down mode, the on-chip oscillator is stopped. This freezes all functions; only the on-chip RAM and special function registers are held. The port pins output the contents of their respective special function registers. A hardware reset is the only way to terminate the power-down mode. Reset re-defines all the special function registers, but does not change the on-chip RAM.

In the power-down mode, $V_{DD}$ and $AV_{DD}$ can be reduced to minimize power consumption. $V_{DD}$ and $AV_{DD}$ must not be reduced before the power-down mode is entered and must be restored to the normal operating voltage before the power-down mode is terminated. The reset that terminates the power-down mode also freezes the oscillator. The reset should not be activated before $V_{DD}$ and $AV_{DD}$ are restored to their normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize (normally less than 10ms).

The status of the external pins during power-down is shown in Table 11. If the power-down mode is entered while the 8XC552 is executing out of external program memory, the port data that is held in the P2 special function register is restored to port 2. If a port latch contains a “1”, the port pin is held HIGH during the power-down mode by the strong pull-up transistor.

**Power Control Register PCON:** The idle and power-down modes are entered by writing to bits in PCON. PCON is not bit addressable. See Figure 41.

**Memory Organization**

The memory organization of the 8XC552 is the same as in the 80C51, with the exception that the 8XC552 has 8k ROM, 256 bytes RAM, and additional SFRs. Addressing modes are the same in the 8XC552 and the 80C51. Details of the differences are given in the following paragraphs.

In the 8XC552, the lower 8k of the 64k program memory address space is filled by internal ROM. By tying the $\overline{EA}$ pin high, the processor fetches instructions from internal program ROM. Bus expansion for accessing program memory from 8k upwards is automatic since external instruction fetches occur automatically when the program counter exceeds 8191. If the $\overline{EA}$ pin is tied low, all program memory fetches are from external memory. The execution speed of the 8XC552 is the same regardless of whether fetches are from external or internal program memory. If all storage is on-chip, then byte location 8191 should be left vacant to prevent an undesired pre-fetch from external program memory address 8192.

Certain locations in program memory are reserved for specific programs. Locations 0000H to 0002H are reserved for the initialization program. Following reset, the CPU always begins execution at locations 0000H. Locations 0003H to 0075H are reserved for the fifteen interrupt request service routines.

Functionally, the internal data memory is the most flexible of the address spaces. The internal data memory space is subdivided into a 256-byte internal data RAM address space and a 128-byte special function register (SFR) address space, as shown in Figure 42.

The internal data RAM address space is 0 to 255. Four 8-bit register banks occupy locations 0 to 31. 128 bit locations of the internal data RAM are accessible through direct addressing. These bits reside in 16 bytes of internal data RAM at locations 20H to 2FH. The stack can be located anywhere in the internal data RAM address space by loading the 8-bit stack pointer. The stack depth may be 256 bytes maximum.

The SFR address space is 128 to 255. All registers except the program counter and the four 8-bit register banks reside in this address space. Memory mapping the SFRs allows them to be accessed as easily as internal RAM, and as such, they can be operated on by most instructions. The 56 SFRs are listed in Figure 43, and their mapping in the SFR address space is shown in Figures 44 and 45. RAM bit addresses are the same as in the 80C51 and are summarized in Figure 46. The special function bit addresses are summarized in Figure 47.

**Table 11. External Pin Status During Idle and Power-Down Modes**

| MODE | MEMORY | ALE | $\overline{PSEN}$ | PORT 0 | PORT 1 | PORT 2 | PORT 3 | PORT 4 | $\overline{PWM0}$/$\overline{PWM1}$ |
|---|---|---|---|---|---|---|---|---|---|
| Idle (1) | Internal | 1 | 1 | Port data | Port data | Port data | Port data | Port data | HIGH |
| Idle (1) | External | 1 | 1 | Floating | Port data | Address | Port data | Port data | HIGH |
| Power-down | Internal | 0 | 0 | Port data | Port data | Port data | Port data | Port data | HIGH |
| Power-down | External | 0 | 0 | Floating | Port data | Port data | Port data | Port data | HIGH |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PCON (87H) | SMOD | – | – | WLE | GF1 | GF0 | PD | IDL |

| BIT | SYMBOL | FUNCTION |
|---|---|---|
| PCON.7 | SMOD | Double Baud rate bit. When set to logic 1 the baud rate is doubled when the serial port SIO0 is being used in modes 1, 2, or 3. |
| PCON.6 | – | (Reserved) |
| PCON.5 | – | (Reserved) |
| PCON.4 | WLE | Watchdog load enable. This flag must be set by software prior to loading timer T3 (watch-dog timer). It is cleared when timer T3 is loaded. |
| PCON.3 | GF1 | General-purpose flag bit |
| PCON.2 | GF0 | General-purpose flag bit |
| PCON.1 | PD | Power-down bit. Setting this bit activates the power-down mode. It can only be set if input $\overline{EW}$ is high. |
| PCON.0 | IDL | Idle mode bit. Setting this bit activates the idle mode. |

If logic 1s are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XX00000).

**Figure 41. Power Control Register (PCON)**

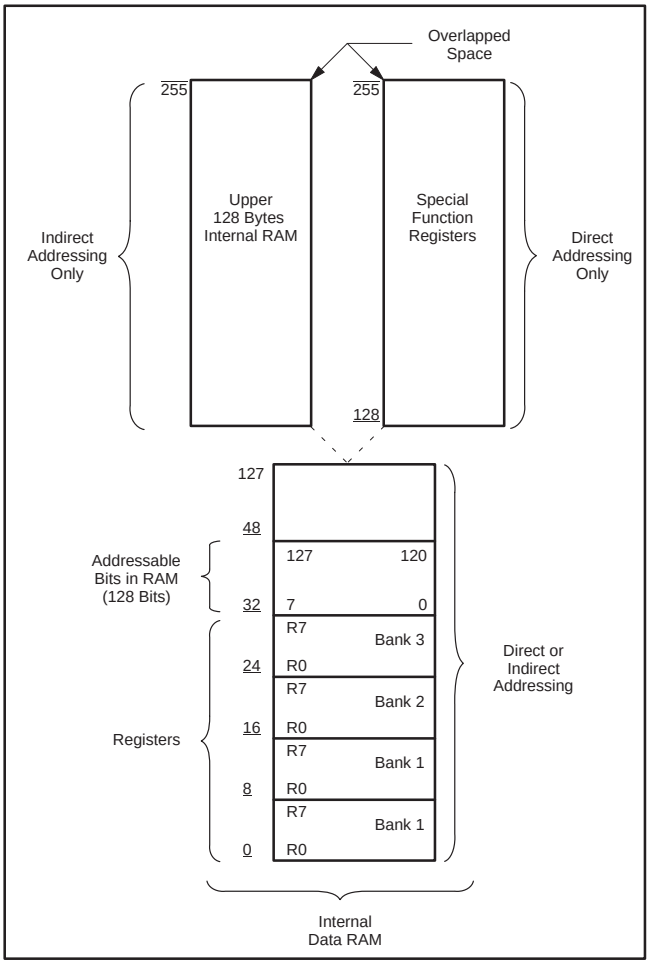



**Figure 42. Internal Data Memory Address Space**

ARITHMETIC REGISTERS:
ACCumulator,* B register,*
Program Status Word*

POINTERS:
Stack Pointer,
Data Pointer (High and Low)

PARALLEL I/O PORTS:
Port 5,* Port 4,*Port 3,*
Port 2,* Port 1,* Port 0*

INTERRUPT SYSTEM:
Interrupt Priority 0,*
Interrupt Priority 1,*
Interrupt Enable 0,*
Interrupt Enable 1*

PULSE WIDTH MODULATED O/Ps:
Pulse Width Modulation Prescaler
Pulse Width Modulation Register 0,
Pulse Width Modulation Register 1

SERIAL I/O PORTS:
Serial 0 CONtrol,* Serial 0 data BUFfer,
Serial 1 CONtrol,* Serial 1 DATa,
Serial 1 STAtus, Serial 1 ADDress, PCON

TIMERS:
Timer MODe, Timer CONtrol,*
Timer Low 0, Timer High 0,
Timer Low 1, Timer High 1,
TiMer T2 CONtrol, TiMer Low 2,
Timer High 2, Timer T3

CAPTURE AND COMPARE LOGIC:
CapTure CONtrol,
TiMer T2 Interrupt flag Register,
CapTure Low 0, CapTure High 0,
CapTure Low 1, CapTure High 1,
CapTure Low 2, CapTure High 2,
CapTure Low 3, CapTure High 3,
CoMpare Low 0, CoMpare High 0,
CoMpare Low 1, CoMpare High 1,
CoMpare Low 2, CoMpare High 2
SeT Enable, ReseT Enable

ADC
ADC cONtrol, ADC High byte

*NOTE: Bit and byte addressable

**Figure 43. Special Function Registers**

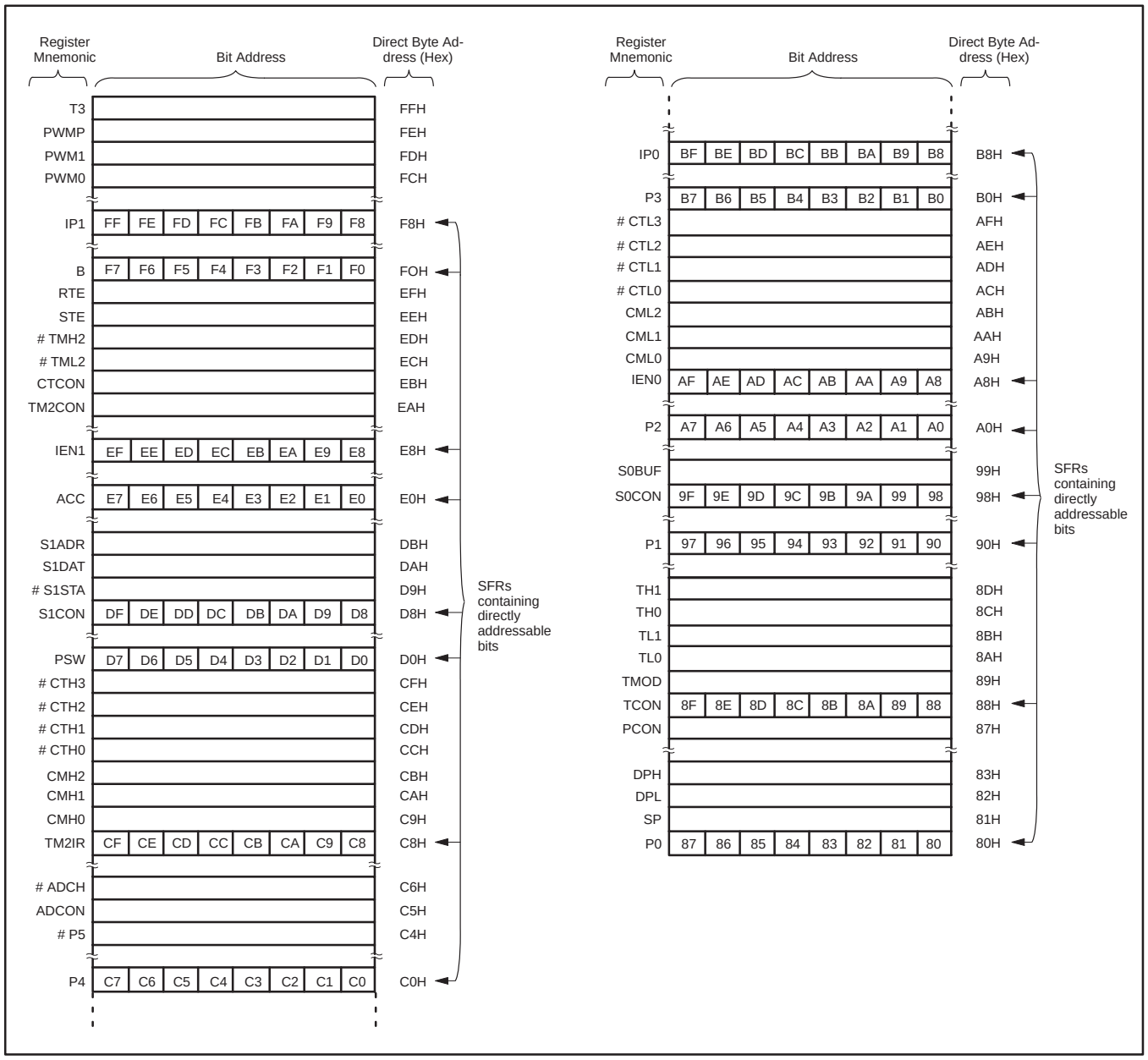



**Figure 44. Mapping of Special Function Registers**

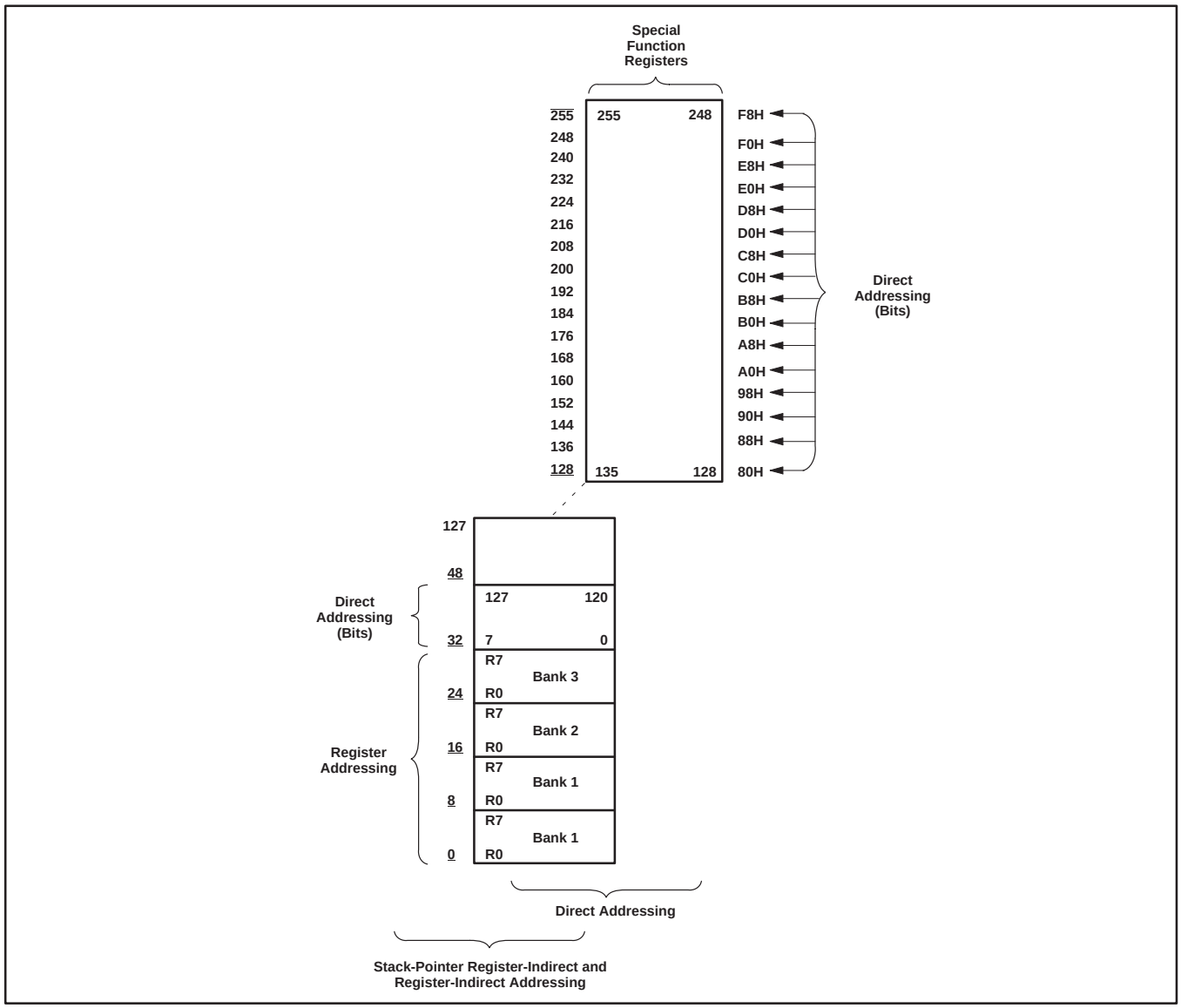



**Figure 45. Bit and Byte Addressing Overview of Internal Data Memory**

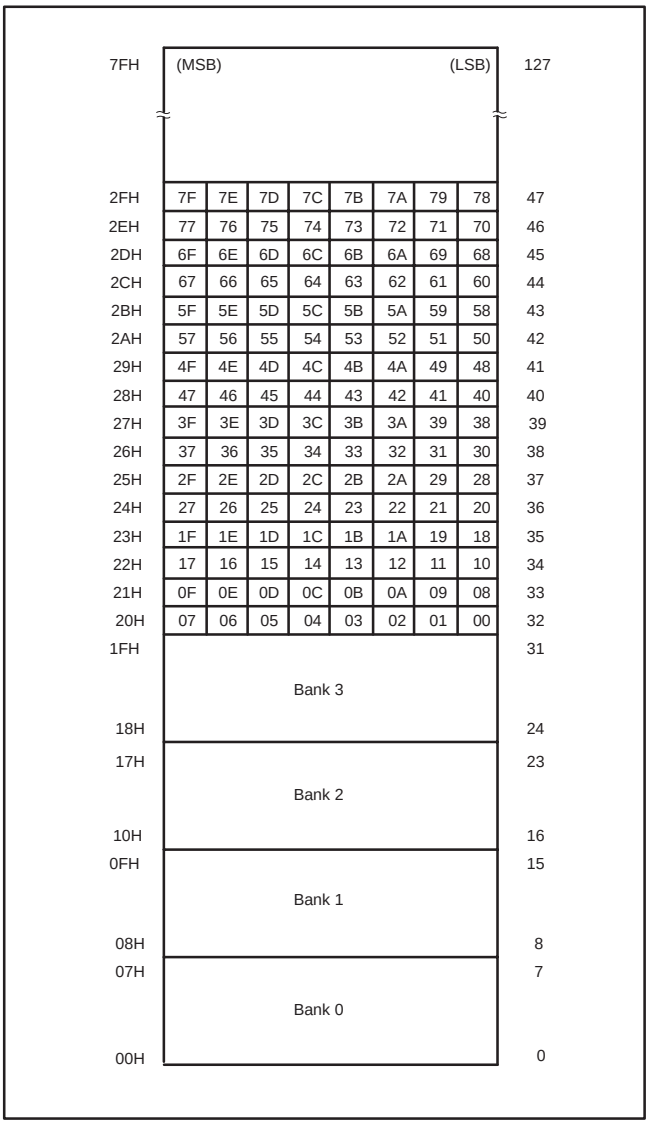



**Figure 46. RAM Bit Addresses**

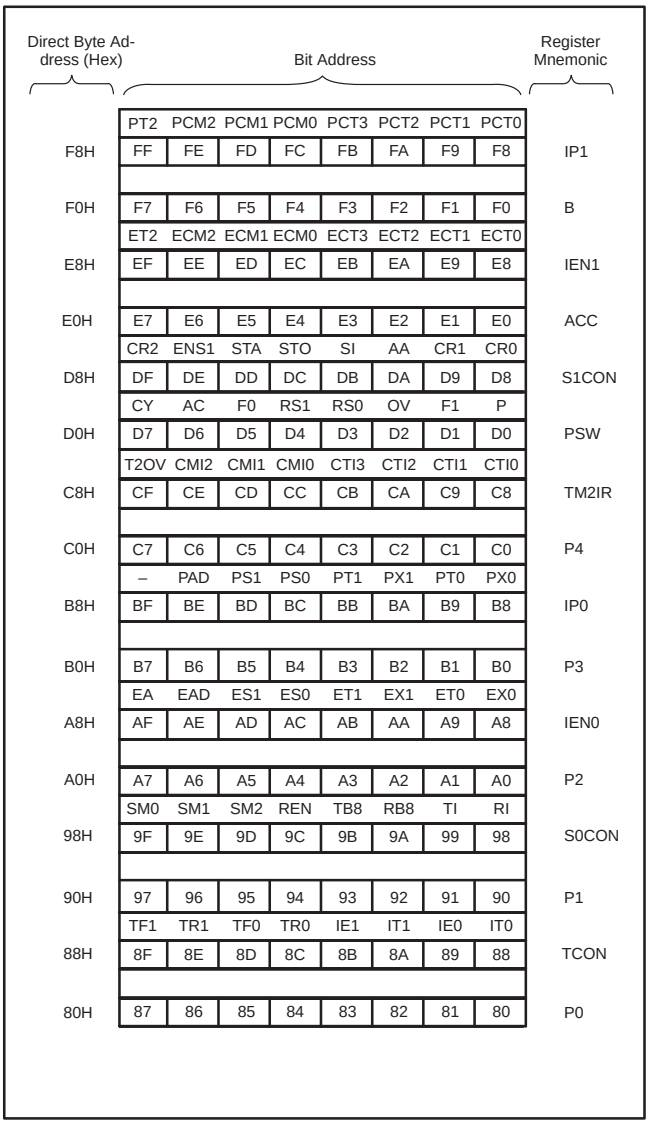



**Figure 47. Special Function Register Bit Address**

**NOTES**

## Definitions

**Short-form specification —** The data in a short-form specification is extracted from a full data sheet with the same type number and title. For detailed information see the relevant data sheet or data handbook.

**Limiting values definition —** Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.

**Application information —** Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

## Disclaimers

**Life support —** These products are not designed for use in life support appliances, devices or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

**Right to make changes —** Philips Semiconductors reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

**Philips Semiconductors**
**811 East Arques Avenue**
**P.O. Box 3409**
**Sunnyvale, California 94088–3409**
**Telephone 800-234-7381**

© Copyright Philips Electronics North America Corporation 1998
All rights reserved. Printed in U.S.A.

Date of release: 08-98

Document order number: 9397 750 04292

Let's make things better.

**Philips Semiconductors**

PHILIPS