# MCS® 51 MICROCONTROLLER FAMILY USER'S MANUAL

ORDER NO.: 272383-002
FEBRUARY 1994

Additional copies of this document or other Intel literature may be obtained from:

Intel Corporation
Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

# MCS® 51 MICROCONTROLLER FAMILY USER'S MANUAL

## CONTENTS

# MCS® 51 Family of Microcontrollers Architectural Overview

1

# MCS® 51 FAMILY OF MICROCONTROLLERS ARCHITECTURAL OVERVIEW

CONTENTS PAGE

## INTRODUCTION

The 8051 is the original member of the MCS®-51 family, and is the core for all MCS-51 devices. The features of the 8051 core are:

- 8-bit CPU optimized for control applications
- Extensive Boolean processing (single-bit logic) capabilities
- 64K Program Memory address space
- 64K Data Memory address space
- 4K bytes of on-chip Program Memory
- 128 bytes of on-chip Data RAM
- 32 bidirectional and individually addressable I/O lines
- Two 16-bit timer/counters
- Full duplex UART
- 6-source/5-vector interrupt structure with two priority levels
- On-chip clock oscillator

The basic architectural structure of this 8051 core is shown in Figure 1.

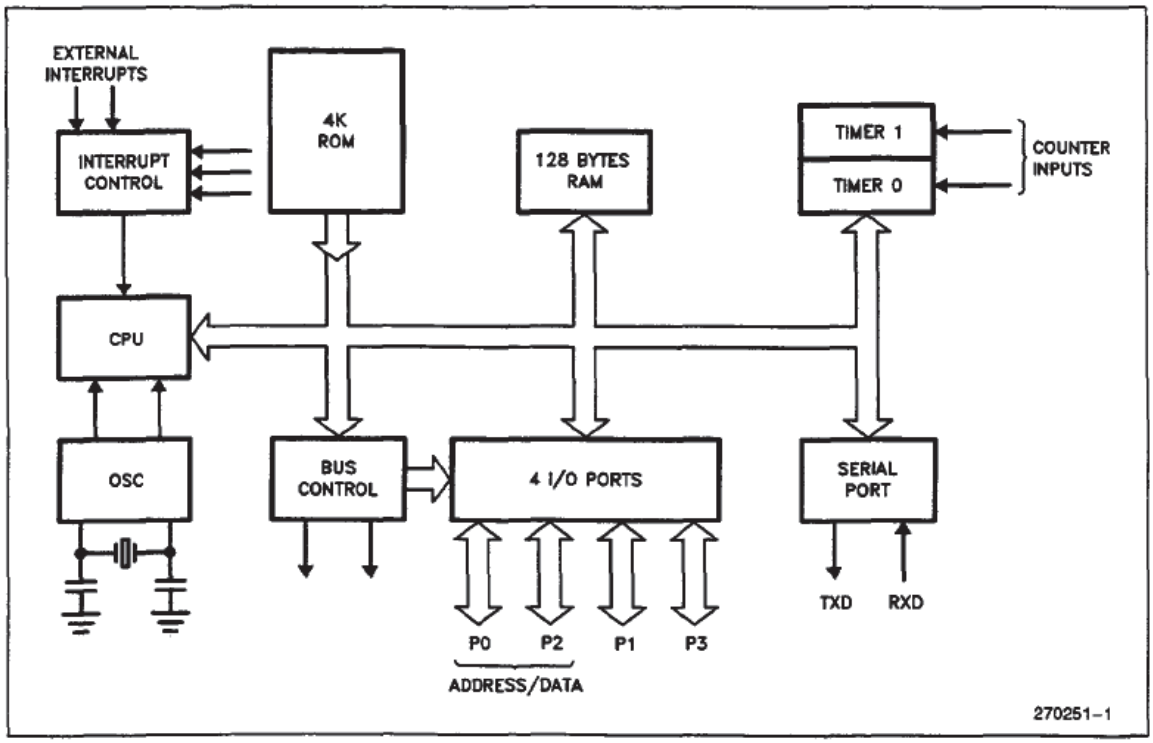



**Figure 1. Block Diagram of the 8051 Core**

**Table 1. The MCS® 51 Family of Microcontrollers**

| DEVICE | ROM/EPROM (bytes) | Register RAM (bytes) | Speed (MHz) | I/O Pins | Timer/ Counters | UART | Interrupt Sources | PCA Channels | A/D Channels | SEP | GSC | DMA Channels | Lock Bits | Power Down & Idle Modes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **8051 Product Line** | | | | | | | | | | | | | | |
| 8031AH | ROMLESS | 128 | 12 | 32 | 2 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | - | - |
| 8051AH | 4K ROM | 128 | 12 | 32 | 2 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | - |
| 8051AHP | 4K ROM | 128 | 12 | 32 | 2 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | P | - |
| 8751H | 4K EPROM | 128 | 12 | 32 | 2 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 1 | - |
| 8751BH | 4K EPROM | 128 | 12 | 32 | 2 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 2 | - |
| **8052 Product Line** | | | | | | | | | | | | | | |
| 8032AH | ROMLESS | 256 | 12 | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | - | - |
| 8052AH | 8K ROM | 256 | 12 | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | - |
| 8752BH | 8K EPROM | 256 | 12 | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 2 | - |
| **80C51 Product Line** | | | | | | | | | | | | | | |
| 80C31BH | ROMLESS | 128 | 12,16 | 32 | 2 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | - | Yes |
| 80C51BH | 4K ROM | 128 | 12,16 | 32 | 2 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | Yes |
| 80C51BHP | 4K ROM | 128 | 12,16 | 32 | 2 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | P | Yes |
| 87C51 | 4K EPROM | 128 | 12,16,20,24 | 32 | 2 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 3 | Yes |
| **8XC52/54/58 Product Line** | | | | | | | | | | | | | | |
| 80C32 | ROMLESS | 256 | 12,16,20,24 | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | - | Yes |
| 80C52 | 8K ROM | 256 | 12,16,20,24 | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 1* | Yes |
| 87C52 | 8K EPROM | 256 | 12,16,20,24 | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 3 | Yes |
| 80C54 | 16K ROM | 256 | 12,16,20,24 | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 1 | Yes |
| 87C54 | 16K EPROM | 256 | 12,16,20,24 | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 3 | Yes |
| 80C58 | 32K ROM | 256 | 12,16,20,24 | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 1 | Yes |
| 87C58 | 32K EPROM | 256 | 12,16,20,24 | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 3 | Yes |
| **8XL52/54/58 Product Line** | | | | | | | | | | | | | | |
| 80L52 | 8K ROM | 256 | 12,16,20* | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 1 | Yes |
| 87L52 | 8K OTP ROM | 256 | 12,16,20* | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 3 | Yes |
| 80L54 | 16K ROM | 256 | 12,16,20* | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 1 | Yes |
| 87L54 | 16K OTP ROM | 256 | 12,16,20* | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 3 | Yes |
| 80L58 | 32K ROM | 256 | 12,16,20* | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 1 | Yes |
| 87L58 | 32K OTP ROM | 256 | 12,16,20* | 32 | 3 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 3 | Yes |

**Table 1. The MCS® 51 Family of Microcontrollers**

| DEVICE | ROM/EPROM (bytes) | Register RAM (bytes) | Speed (MHz) | I/O Pins | Timer/ Counters | UART | Interrupt Sources | PCA Channels | A/D Channels | SEP | GSC | DMA Channels | Lock Bits | Power Down & Idle Modes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **8XC51FA/FB/FC Product Line** | | | | | | | | | | | | | | |
| 80C51FA | ROMLESS | 256 | 12,16 | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | - | Yes |
| 83C51FA | 8K ROM | 256 | 12,16 | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 0 | Yes |
| 87C51FA | 8K EPROM | 256 | 12,16,20,24 | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 3 | Yes |
| 83C51FB | 16K ROM | 256 | 12,16,20,24 | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 1 | Yes |
| 87C51FB | 16K EPROM | 256 | 12,16,20,24 | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 3 | Yes |
| 83C51FC | 32K ROM | 256 | 12,16,20,24 | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 1 | Yes |
| 87C51FC | 32K EPROM | 256 | 12,16,20,24 | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 3 | Yes |
| **8XL51FA/FB/FC Product Line** | | | | | | | | | | | | | | |
| 80L51FA | ROMLESS | 256 | 12,16,20* | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | - | Yes |
| 83L51FA | 8K ROM | 256 | 12,16,20* | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 1 | Yes |
| 87L51FA | 8K OTP ROM | 256 | 12,16,20* | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 3 | Yes |
| 83L51FB | 16K ROM | 256 | 12,16,20* | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 1 | Yes |
| 87L51FB | 16K OTP ROM | 256 | 12,16,20* | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 3 | Yes |
| 83L51FC | 32K ROM | 256 | 12,16,20* | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 1 | Yes |
| 87L51FC | 32K OTP ROM | 256 | 12,16,20* | 32 | 3 | 1 | 7 | 5 | 0 | 0 | 0 | 0 | 3 | Yes |
| **8XC51GX Product Line** | | | | | | | | | | | | | | |
| 80C51GB | ROMLESS | 256 | 12,16 | 48 | 3 | 1 | 15 | 10 | 8 | 1 | 0 | 0 | - | Yes |
| 83C51GB | 8K ROM | 256 | 12,16 | 48 | 3 | 1 | 15 | 10 | 8 | 1 | 0 | 0 | 1 | Yes |
| 87C51GB | 8K EPROM | 256 | 12,16 | 48 | 3 | 1 | 15 | 10 | 8 | 1 | 0 | 0 | 3 | Yes |
| **8XC152 Product Line*** | | | | | | | | | | | | | | |
| 80C152JA | ROMLESS | 256 | 16.5 | 40 | 2 | 1 | 11 | 0 | 0 | 1 | 1 | 2 | - | Yes |
| 80C152JB | ROMLESS | 256 | 16.5 | 56 | 2 | 1 | 11 | 0 | 0 | 1 | 1 | 2 | - | Yes |
| 83C152JA | 8K ROM | 256 | 16.5 | 40 | 2 | 1 | 11 | 0 | 0 | 1 | 1 | 2 | 0 | Yes |
| **8XC51SL Product Line*** | | | | | | | | | | | | | | |
| 80C51SL-BG | ROMLESS | 256 | 16 | 24 | 2 | 1 | 10 | 0 | 4 | 0 | 1 | 0 | - | Yes |
| 81C51SL-BG | 8K *ROM | 256 | 16 | 24 | 2 | 1 | 10 | 0 | 4 | 0 | 1 | 0 | 0 | Yes |
| 83C51SL-BG | 8K ROM | 256 | 16 | 24 | 2 | 1 | 10 | 0 | 4 | 0 | 1 | 0 | 0 | Yes |
| 80C51SLAH | ROMLESS | 256 | 16 | 24 | 2 | 1 | 10 | 0 | 4 | 0 | 1 | 0 | - | Yes |
| 81C51SLAH | 16K *ROM | 256 | 16 | 24 | 2 | 1 | 10 | 0 | 4 | 0 | 1 | 0 | 0 | Yes |
| 83C51SLAH | 16K ROM | 256 | 16 | 24 | 2 | 1 | 10 | 0 | 4 | 0 | 1 | 0 | 0 | Yes |
| 87C51SLAH | 16K EPROM | 256 | 16 | 24 | 2 | 1 | 10 | 0 | 4 | 0 | 1 | 0 | 0 | Yes |
| 80C51SLAL | ROMLESS | 256 | 16 | 24 | 2 | 1 | 10 | 0 | 4 | 0 | 1 | 0 | - | Yes |
| 81C51SLAL | 16K *ROM | 256 | 16 | 24 | 2 | 1 | 10 | 0 | 4 | 0 | 1 | 0 | 0 | Yes |
| 83C51SLAL | 16K ROM | 256 | 16 | 24 | 2 | 1 | 10 | 0 | 4 | 0 | 1 | 0 | 0 | Yes |
| 87C51SLAL | 16K EPROM | 256 | 16 | 24 | 2 | 1 | 10 | 0 | 4 | 0 | 1 | 0 | 0 | Yes |

ROM/OTP ROM/EPROM (bytes): *ROM = SystemSoft Standard BIOS
Speed (Mhz): 24I = 24 MHz Internal-only operation
20* = 20MHz Available for Commerical Temperature Range Only
Lock Bits: 1* = 1 Lock Bit for 20MHz & 24MHz parts, no Lock Bit for 12 & 16MHz parts
P = Program verification disabled, external memory access limited to 4K
8XC152 Product Line* = Communication Controller
8XC51SL Product Line* = Keyboard Controller

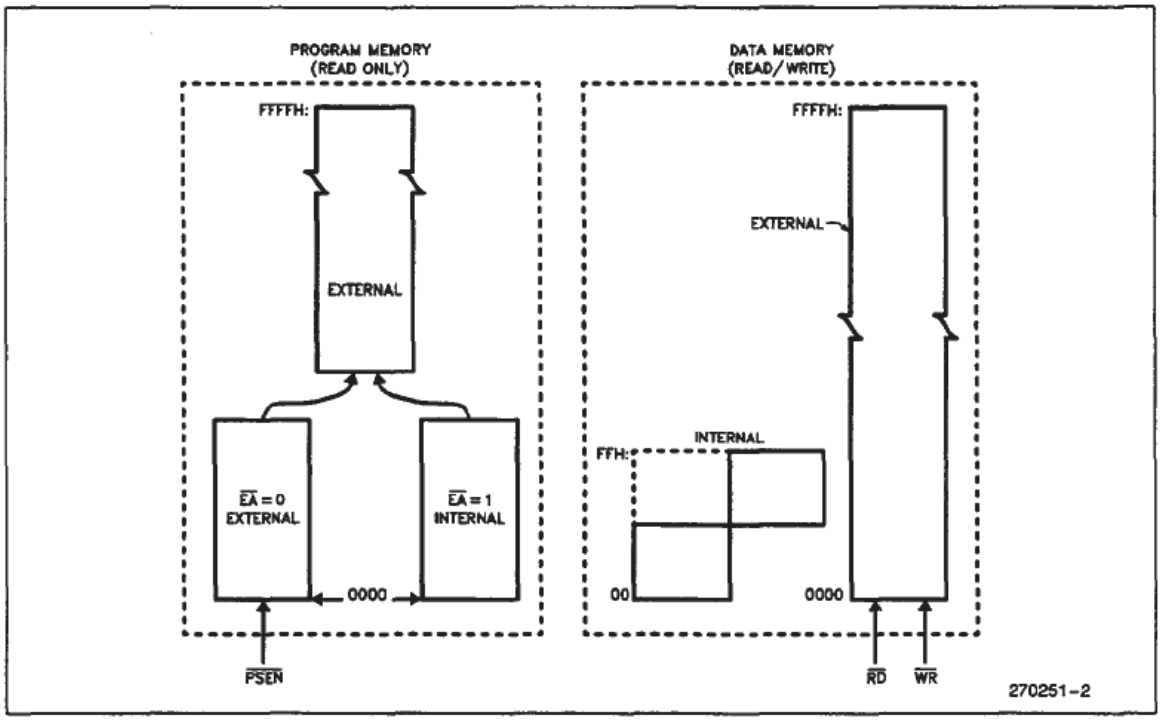



**Figure 2. MCS®-51 Memory Structure**

## CHMOS Devices

Functionally, the CHMOS devices (designated with "C" in the middle of the device name) are all fully compatible with the 8051, but being CMOS, draw less current than an HMOS counterpart. To further exploit the power savings available in CMOS circuitry, two reduced power modes are added:

- Software-invoked Idle Mode, during which the CPU is turned off while the RAM and other on-chip peripherals continue operating. In this mode, current draw is reduced to about 15% of the current drawn when the device is fully active.
- Software-invoked Power Down Mode, during which all on-chip activities are suspended. The on-chip RAM continues to hold its data. In this mode the device typically draws less than 10 μA.

Although the 80C51BH is functionally compatible with its HMOS counterpart, specific differences between the two types of devices must be considered in the design of an application circuit if one wishes to ensure complete interchangeability between the HMOS and CHMOS devices. These considerations are discussed in the Application Note AP-252, "Designing with the 80C51BH".

For more information on the individual devices and features listed in Table 1, refer to the Hardware Descriptions and Data Sheets of the specific device.

## MEMORY ORGANIZATION IN MCS®-51 DEVICES

### Logical Separation of Program and Data Memory

All MCS-51 devices have separate address spaces for Program and Data Memory, as shown in Figure 2. The logical separation of Program and Data Memory allows the Data Memory to be accessed by 8-bit addresses, which can be more quickly stored and manipulated by an 8-bit CPU. Nevertheless, 16-bit Data Memory addresses can also be generated through the DPTR register.

Program Memory can only be read, not written to. There can be up to 64K bytes of Program Memory. In the ROM and EPROM versions of these devices the lowest 4K, 8K or 16K bytes of Program Memory are provided on-chip. Refer to Table 1 for the amount of on-chip ROM (or EPROM) on each device. In the ROMless versions all Program Memory is external. The read strobe for external Program Memory is the signal $\overline{PSEN}$ (Program Store Enable).

Data Memory occupies a separate address space from Program Memory. Up to 64K bytes of external RAM can be addressed in the external Data Memory space. The CPU generates read and write signals, $\overline{RD}$ and $\overline{WR}$, as needed during external Data Memory accesses.

External Program Memory and external Data Memory may be combined if desired by applying the $\overline{RD}$ and $\overline{PSEN}$ signals to the inputs of an AND gate and using the output of the gate as the read strobe to the external Program/Data memory.

## Program Memory

Figure 3 shows a map of the lower part of the Program Memory. After reset, the CPU begins execution from location 0000H.

As shown in Figure 3, each interrupt is assigned a fixed location in Program Memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose Program Memory.

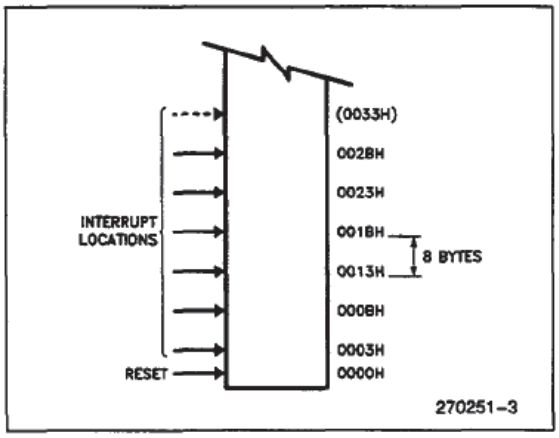



**Figure 3. MCS®-51 Program Memory**

The interrupt service locations are spaced at 8-byte intervals: 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1, 001BH for Timer 1, etc. If an interrupt service routine is short enough (as is often the case in control applications), it can reside entirely within that 8-byte interval. Longer service routines can use a jump instruction to skip over subsequent interrupt locations, if other interrupts are in use.

The lowest 4K (or 8K or 16K) bytes of Program Memory can be either in the on-chip ROM or in an external ROM. This selection is made by strapping the $\overline{EA}$ (External Access) pin to either $V_{CC}$ or $V_{SS}$.

In the 4K byte ROM devices, if the $\overline{EA}$ pin is strapped to $V_{CC}$, then program fetches to addresses 0000H through 0FFFH are directed to the internal ROM. Program fetches to addresses 1000H through FFFFH are directed to external ROM.

In the 8K byte ROM devices, $\overline{EA} = V_{CC}$ selects addresses 0000H through 1FFFH to be internal, and addresses 2000H through FFFFH to be external.

In the 16K byte ROM devices, $\overline{EA} = V_{CC}$ selects addresses 0000H through 3FFFH to be internal, and addresses 4000H through FFFFH to be external.

If the $\overline{EA}$ pin is strapped to $V_{SS}$, then all program fetches are directed to external ROM. The ROMless parts must have this pin externally strapped to $V_{SS}$ to enable them to execute properly.

The read strobe to external ROM, $\overline{PSEN}$, is used for all external program fetches. $\overline{PSEN}$ is not activated for internal program fetches.

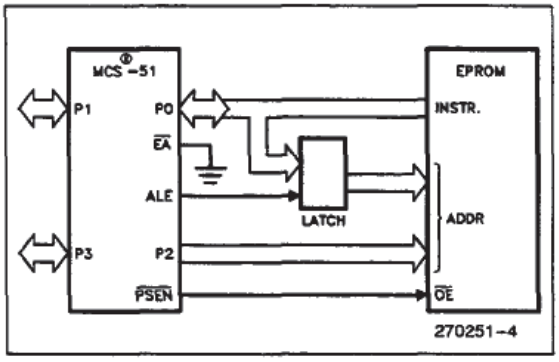



**Figure 4. Executing from External Program Memory**

The hardware configuration for external program execution is shown in Figure 4. Note that 16 I/O lines (Ports 0 and 2) are dedicated to bus functions during external Program Memory fetches. Port 0 (P0 in Figure 4) serves as a multiplexed address/data bus. It emits the low byte of the Program Counter (PCL) as an address, and then goes into a float state awaiting the arrival of the code byte from the Program Memory. During the time that the low byte of the Program Counter is valid on P0, the signal ALE (Address Latch Enable) clocks this byte into an address latch. Meanwhile, Port 2 (P2 in Figure 4) emits the high byte of the Program Counter (PCH). Then $\overline{PSEN}$ strobes the EPROM and the code byte is read into the microcontroller.

Program Memory addresses are always 16 bits wide, even though the actual amount of Program Memory used may be less than 64K bytes. External program execution sacrifices two of the 8-bit ports, P0 and P2, to the function of addressing the Program Memory.

## Data Memory

The right half of Figure 2 shows the internal and external Data Memory spaces available to the MCS-51 user.

Figure 5 shows a hardware configuration for accessing up to 2K bytes of external RAM. The CPU in this case is executing from internal ROM. Port 0 serves as a multiplexed address/data bus to the RAM, and 3 lines of Port 2 are being used to page the RAM. The CPU generates $\overline{RD}$ and $\overline{WR}$ signals as needed during external RAM accesses.

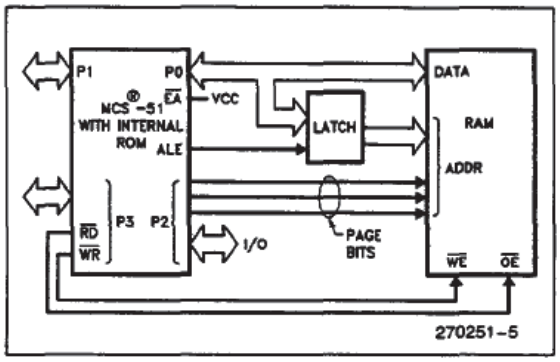



**Figure 5. Accessing External Data Memory. If the Program Memory is Internal, the Other Bits of P2 are Available as I/O.**

There can be up to 64K bytes of external Data Memory. External Data Memory addresses can be either 1 or 2 bytes wide. One-byte addresses are often used in conjunction with one or more other I/O lines to page the RAM, as shown in Figure 5. Two-byte addresses can also be used, in which case the high address byte is emitted at Port 2.

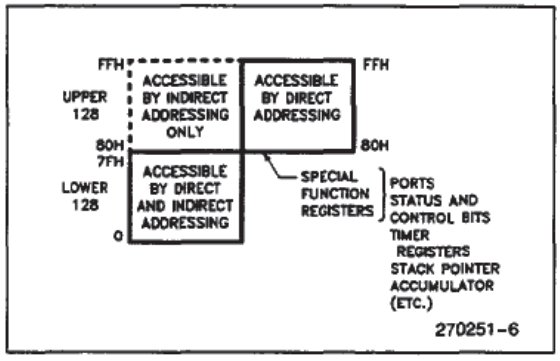



**Figure 6. Internal Data Memory**

Internal Data Memory is mapped in Figure 6. The memory space is shown divided into three blocks, which are generally referred to as the Lower 128, the Upper 128, and SFR space.

Internal Data Memory addresses are always one byte wide, which implies an address space of only 256 bytes. However, the addressing modes for internal RAM can in fact accommodate 384 bytes, using a simple trick. Direct addresses higher than 7FH access one memory space, and indirect addresses higher than 7FH access a different memory space. Thus Figure 6 shows the Upper 128 and SFR space occupying the same block of addresses, 80H through FFH, although they are physically separate entities.

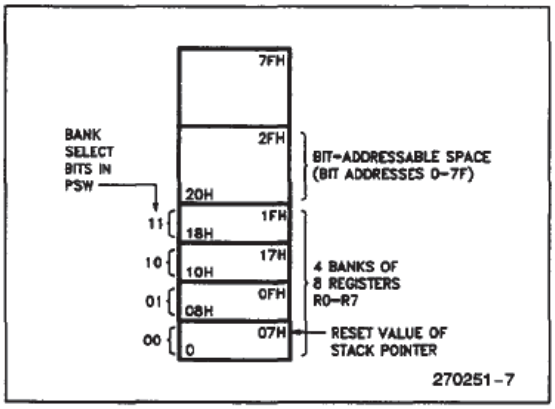



**Figure 7. The Lower 128 Bytes of Internal RAM**

The Lower 128 bytes of RAM are present in all MCS-51 devices as mapped in Figure 7. The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing.

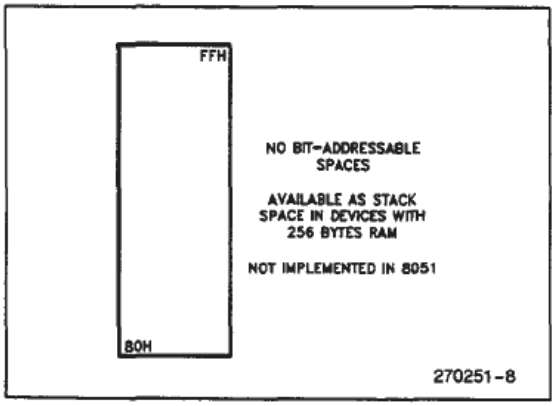



**Figure 8. The Upper 128 Bytes of Internal RAM**

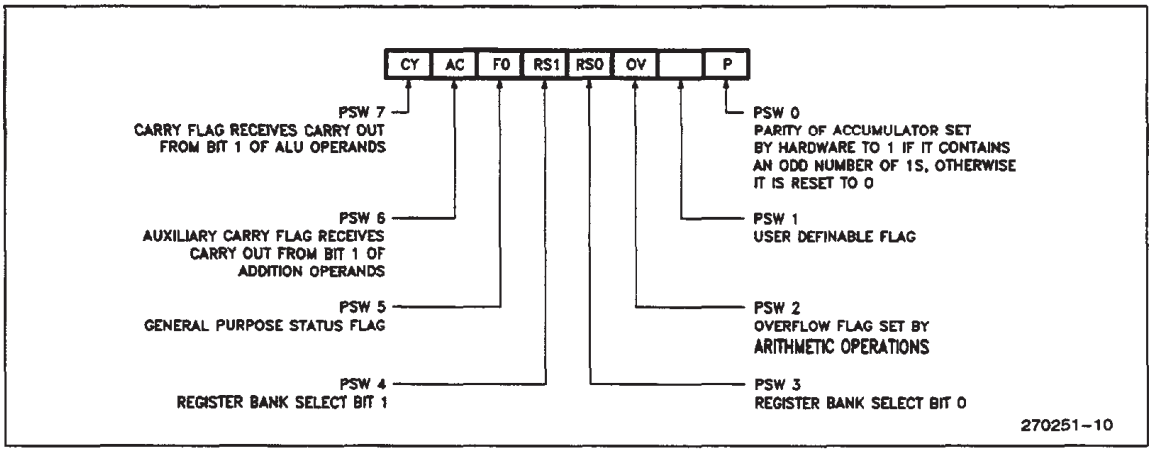



**Figure 10. PSW (Program Status Word) Register in MCS®-51 Devices**

The next 16 bytes above the register banks form a block of bit-addressable memory space. The MCS-51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00H through 7FH.

All of the bytes in the Lower 128 can be accessed by either direct or indirect addressing. The Upper 128 (Figure 8) can only be accessed by indirect addressing. The Upper 128 bytes of RAM are not implemented in the 8051, but are in the devices with 256 bytes of RAM. (See Table 1).

Figure 9 gives a brief look at the Special Function Register (SFR) space. SFRs include the Port latches, timers, peripheral controls, etc. These registers can only be accessed by direct addressing. In general, all MCS-51 microcontrollers have the same SFRs as the 8051, and at the same addresses in SFR space. However, enhancements to the 8051 have additional SFRs that are not present in the 8051, nor perhaps in other proliferations of the family.

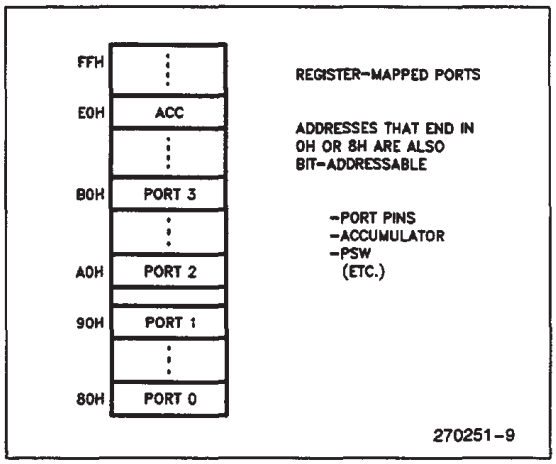



**Figure 9. SFR Space**

Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 000B. The bit addresses in this area are 80H through FFH.

## THE MCS®-51 INSTRUCTION SET

All members of the MCS-51 family execute the same instruction set. The MCS-51 instruction set is optimized for 8-bit control applications. It provides a variety of fast addressing modes for accessing the internal RAM to facilitate byte operations on small data structures. The instruction set provides extensive support for one-bit variables as a separate data type, allowing direct bit manipulation in control and logic systems that require Boolean processing.

An overview of the MCS-51 instruction set is presented below, with a brief description of how certain instructions might be used. References to "the assembler" in this discussion are to Intel's MCS-51 Macro Assembler, ASM51. More detailed information on the instruction set can be found in the MCS-51 Macro Assembler User's Guide (Order No. 9800937 for ISIS Systems, Order No. 122752 for DOS Systems).

### Program Status Word

The Program Status Word (PSW) contains several status bits that reflect the current state of the CPU. The PSW, shown in Figure 10, resides in SFR space. It contains the Carry bit, the Auxiliary Carry (for BCD operations), the two register bank select bits, the Overflow flag, a Parity bit, and two user-definable status flags.

The Carry bit, other than serving the functions of a Carry bit in arithmetic operations, also serves as the "Accumulator" for a number of Boolean operations.

The bits RS0 and RS1 are used to select one of the four register banks shown in Figure 7. A number of instructions refer to these RAM locations as R0 through R7. The selection of which of the four banks is being referred to is made on the basis of the bits RS0 and RS1 at execution time.

The Parity bit reflects the number of 1s in the Accumulator: P = 1 if the Accumulator contains an odd number of 1s, and P = 0 if the Accumulator contains an even number of 1s. Thus the number of 1s in the Accumulator plus P is always even.

Two bits in the PSW are uncommitted and may be used as general purpose status flags.

## Addressing Modes

The addressing modes in the MCS-51 instruction set are as follows:

### DIRECT ADDRESSING

In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal Data RAM and SFRs can be directly addressed.

### INDIRECT ADDRESSING

In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed.

The address register for 8-bit addresses can be R0 or R1 of the selected register bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit "data pointer" register, DPTR.

### REGISTER INSTRUCTIONS

The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed. One of four banks is selected at execution time by the two bank select bits in the PSW.

### REGISTER-SPECIFIC INSTRUCTIONS

Some instructions are specific to a certain register. For example, some instructions always operate on the Accumulator, or Data Pointer, etc., so no address byte is needed to point to it. The opcode itself does that. Instructions that refer to the Accumlator as A assemble as accumulator-specific opcodes.

### IMMEDIATE CONSTANTS

The value of a constant can follow the opcode in Program Memory. For example,

```
MOV A, #100
```

loads the Accumulator with the decimal number 100. The same number could be specified in hex digits as 64H.

### INDEXED ADDRESSING

Only Program Memory can be accessed with indexed addressing, and it can only be read. This addressing mode is intended for reading look-up tables in Program Memory. A 16-bit base register (either DPTR or the Program Counter) points to the base of the table, and the Accumulator is set up with the table entry number. The address of the table entry in Program Memory is formed by adding the Accumulator data to the base pointer.

Another type of indexed addressing is used in the "case jump" instruction. In this case the destination address of a jump instruction is computed as the sum of the base pointer and the Accumulator data.

## Arithmetic Instructions

The menu of arithmetic instructions is listed in Table 2. The table indicates the addressing modes that can be used with each instruction to access the <byte> operand. For example, the ADD A, <byte> instruction can be written as:

```
ADD    A,7FH     (direct addressing)
ADD    A,@R0     (indirect addressing)
ADD    A,R7      (register addressing)
ADD    A,#127    (immediate constant)
```

The execution times listed in Table 2 assume a 12 MHz clock frequency. All of the arithmetic instructions execute in 1 μs except the INC DPTR instruction, which takes 2 μs, and the Multiply and Divide instructions, which take 4 μs.

Note that any byte in the internal Data Memory space can be incremented or decremented without going through the Accumulator.

One of the INC instructions operates on the 16-bit Data Pointer. The Data Pointer is used to generate 16-bit addresses for external memory, so being able to increment it in one 16-bit operation is a useful feature.

The MUL AB instruction multiplies the Accumulator by the data in the B register and puts the 16-bit product into the concatenated B and Accumulator registers.

**Table 2. A List of the MCS®-51 Arithmetic Instructions**

| Mnemonic | Operation | Addressing Modes | | | | Execution Time (μs) |
|---|---|---|---|---|---|---|
| | | Dir | Ind | Reg | Imm | |
| ADD A,<byte> | A = A + <byte> | X | X | X | X | 1 |
| ADDC A,<byte> | A = A + <byte> + C | X | X | X | X | 1 |
| SUBB A,<byte> | A = A − <byte> − C | X | X | X | X | 1 |
| INC A | A = A + 1 | Accumulator only | | | | 1 |
| INC <byte> | <byte> = <byte> + 1 | X | X | X | | 1 |
| INC DPTR | DPTR = DPTR + 1 | Data Pointer only | | | | 2 |
| DEC A | A = A − 1 | Accumulator only | | | | 1 |
| DEC <byte> | <byte> = <byte> − 1 | X | X | X | | 1 |
| MUL AB | B:A = B x A | ACC and B only | | | | 4 |
| DIV AB | A = Int [A/B]<br>B = Mod [A/B] | ACC and B only | | | | 4 |
| DA A | Decimal Adjust | Accumulator only | | | | 1 |

The DIV AB instruction divides the Accumulator by the data in the B register and leaves the 8-bit quotient in the Accumulator, and the 8-bit remainder in the B register.

Oddly enough, DIV AB finds less use in arithmetic "divide" routines than in radix conversions and programmable shift operations. An example of the use of DIV AB in a radix conversion will be given later. In shift operations, dividing a number by $2^n$ shifts its n bits to the right. Using DIV AB to perform the division completes the shift in 4 μs and leaves the B register holding the bits that were shifted out.

The DA A instruction is for BCD arithmetic operations. In BCD arithmetic, ADD and ADDC instructions should always be followed by a DA A operation, to ensure that the result is also in BCD. Note that DA A will not convert a binary number to BCD. The DA A operation produces a meaningful result only as the second step in the addition of two BCD bytes.

**Table 3. A List of the MCS®-51 Logical Instructions**

| Mnemonic | Operation | Addressing Modes | | | | Execution Time (μs) |
|---|---|---|---|---|---|---|
| | | Dir | Ind | Reg | Imm | |
| ANL A,<byte> | A = A .AND. <byte> | X | X | X | X | 1 |
| ANL <byte>,A | <byte> = <byte> .AND. A | X | | | | 1 |
| ANL <byte>,#data | <byte> = <byte> .AND. #data | X | | | | 2 |
| ORL A,<byte> | A = A .OR. <byte> | X | X | X | X | 1 |
| ORL <byte>,A | <byte> = <byte> .OR. A | X | | | | 1 |
| ORL <byte>,#data | <byte> = <byte> .OR. #data | X | | | | 2 |
| XRL A,<byte> | A = A .XOR. <byte> | X | X | X | X | 1 |
| XRL <byte>,A | <byte> = <byte> .XOR. A | X | | | | 1 |
| XRL <byte>,#data | <byte> = <byte> .XOR. #data | X | | | | 2 |
| CRL A | A = 00H | Accumulator only | | | | 1 |
| CPL A | A = .NOT. A | Accumulator only | | | | 1 |
| RL A | Rotate ACC Left 1 bit | Accumulator only | | | | 1 |
| RLC A | Rotate Left through Carry | Accumulator only | | | | 1 |
| RR A | Rotate ACC Right 1 bit | Accumulator only | | | | 1 |
| RRC A | Rotate Right through Carry | Accumulator only | | | | 1 |
| SWAP A | Swap Nibbles in A | Accumulator only | | | | 1 |

## Logical Instructions

Table 3 shows the list of MCS-51 logical instructions. The instructions that perform Boolean operations (AND, OR, Exclusive OR, NOT) on bytes perform the operation on a bit-by-bit basis. That is, if the Accumulator contains 00110101B and <byte> contains 01010011B, then

```
ANL     A,<byte>
```

will leave the Accumulator holding 00010001B.

The addressing modes that can be used to access the <byte> operand are listed in Table 3. Thus, the ANL A,<byte> instruction may take any of the forms

```
ANL     A,7FH       (direct addressing)
ANL     A,@R1       (indirect addressing)
ANL     A,R6        (register addressing)
ANL     A,#53H      (immediate constant)
```

All of the logical instructions that are Accumulator-specific execute in 1μs (using a 12 MHz clock). The others take 2 μs.

Note that Boolean operations can be performed on any byte in the lower 128 internal Data Memory space or the SFR space using direct addressing, without having to use the Accumulator. The XRL <byte>, #data instruction, for example, offers a quick and easy way to invert port bits, as in

```
XRL     P1,#0FFH
```

If the operation is in response to an interrupt, not using the Accumulator saves the time and effort to stack it in the service routine.

The Rotate instructions (RL A, RLC A, etc.) shift the Accumulator 1 bit to the left or right. For a left rotation, the MSB rolls into the LSB position. For a right rotation, the LSB rolls into the MSB position.

The SWAP A instruction interchanges the high and low nibbles within the Accumulator. This is a useful operation in BCD manipulations. For example, if the Accumulator contains a binary number which is known to be less than 100, it can be quickly converted to BCD by the following code:

```
MOV     B,#10
DIV     AB
SWAP    A
ADD     A,B
```

Dividing the number by 10 leaves the tens digit in the low nibble of the Accumulator, and the ones digit in the B register. The SWAP and ADD instructions move the tens digit to the high nibble of the Accumulator, and the ones digit to the low nibble.

## Data Transfers

### INTERNAL RAM

Table 4 shows the menu of instructions that are available for moving data around within the internal memory spaces, and the addressing modes that can be used with each one. With a 12 MHz clock, all of these instructions execute in either 1 or 2 μs.

The MOV <dest>, <src> instruction allows data to be transferred between any two internal RAM or SFR locations without going through the Accumulator. Remember the Upper 128 byes of data RAM can be accessed only by indirect addressing, and SFR space only by direct addressing.

Note that in all MCS-51 devices, the stack resides in on-chip RAM, and grows upwards. The PUSH instruction first increments the Stack Pointer (SP), then copies the byte into the stack. PUSH and POP use only direct addressing to identify the byte being saved or restored,

**Table 4. A List of the MCS®-51 Data Transfer Instructions that Access Internal Data Memory Space**

| Mnemonic | Operation | Addressing Modes | | | | Execution Time (μs) |
|---|---|---|---|---|---|---|
| | | Dir | Ind | Reg | Imm | |
| MOV A,<src> | A = <src> | X | X | X | X | 1 |
| MOV <dest>,A | <dest> = A | X | X | X | | 1 |
| MOV <dest>, <src> | <dest> = <src> | X | X | X | X | 2 |
| MOV DPTR,#data16 | DPTR = 16-bit immediate constant. | | | | X | 2 |
| PUSH <src> | INC SP : MOV "@SP",<src> | X | | | | 2 |
| POP <dest> | MOV <dest>, "@SP" : DEC SP | X | | | | 2 |
| XCH A,<byte> | ACC and <byte> exchange data | X | X | X | | 1 |
| XCHD A,@Ri | ACC and @Ri exchange low nibbles | | X | | | 1 |

but the stack itself is accessed by indirect addressing using the SP register. This means the stack can go into the Upper 128, if they are implemented, but not into SFR space.

In devices that do not implement the Upper 128, if the SP points to the Upper 128, PUSHed bytes are lost, and POPped bytes are indeterminate.

The Data Transfer instructions include a 16-bit MOV that can be used to initialize the Data Pointer (DPTR) for look-up tables in Program Memory, or for 16-bit external Data Memory accesses.

The XCH A, <byte> instruction causes the Accumulator and addressed byte to exchange data. The XCHD A,@Ri instruction is similar, but only the low nibbles are involved in the exchange.

To see how XCH and XCHD can be used to facilitate data manipulations, consider first the problem of shifting an 8-digit BCD number two digits to the right. Figure 11 shows how this can be done using direct MOVs, and for comparison how it can be done using XCH instructions. To aid in understanding how the code works, the contents of the registers that are holding the BCD number and the content of the Accumulator are shown alongside each instruction to indicate their status after the instruction has been executed.

| | | 2A | 2B | 2C | 2D | 2E | ACC |
|---|---|---|---|---|---|---|---|
| MOV | A,2EH | 00 | 12 | 34 | 56 | 78 | 78 |
| MOV | 2EH,2DH | 00 | 12 | 34 | 56 | 56 | 78 |
| MOV | 2DH,2CH | 00 | 12 | 34 | 34 | 56 | 78 |
| MOV | 2CH,2BH | 00 | 12 | 12 | 34 | 56 | 78 |
| MOV | 2BH,#0 | 00 | 00 | 12 | 34 | 56 | 78 |

(a) Using direct MOVs: 14 bytes, 9 μs

| | | 2A | 2B | 2C | 2D | 2E | ACC |
|---|---|---|---|---|---|---|---|
| CLR | A | 00 | 12 | 34 | 56 | 78 | 00 |
| XCH | A,2BH | 00 | 00 | 34 | 56 | 78 | 12 |
| XCH | A,2CH | 00 | 00 | 12 | 56 | 78 | 34 |
| XCH | A,2DH | 00 | 00 | 12 | 34 | 78 | 56 |
| XCH | A,2EH | 00 | 00 | 12 | 34 | 56 | 78 |

(b) Using XCHs: 9 bytes, 5 μs

**Figure 11. Shifting a BCD Number Two Digits to the Right**

After the routine has been executed, the Accumulator contains the two digits that were shifted out on the right. Doing the routine with direct MOVs uses 14 code bytes and 9 μs of execution time (assuming a 12 MHz clock). The same operation with XCHs uses less code and executes almost twice as fast.

To right-shift by an odd number of digits, a one-digit shift must be executed. Figure 12 shows a sample of code that will right-shift a BCD number one digit, using the XCHD instruction. Again, the contents of the registers holding the number and of the Accumulator are shown alongside each instruction.

| | | | 2A | 2B | 2C | 2D | 2E | ACC |
|---|---|---|---|---|---|---|---|---|
| | MOV | R1,#2EH | 00 | 12 | 34 | 56 | 78 | XX |
| | MOV | R0,#2DH | 00 | 12 | 34 | 56 | 78 | XX |
| | loop for R1 = 2EH: | | | | | | | |
| LOOP: | MOV | A,@R1 | 00 | 12 | 34 | 56 | 78 | 78 |
| | XCHD | A,@R0 | 00 | 12 | 34 | 58 | 78 | 76 |
| | SWAP | A | 00 | 12 | 34 | 58 | 78 | 67 |
| | MOV | @R1,A | 00 | 12 | 34 | 58 | 67 | 67 |
| | DEC | R1 | 00 | 12 | 34 | 58 | 67 | 67 |
| | DEC | R0 | 00 | 12 | 34 | 58 | 67 | 67 |
| | CJNE | R1,#2AH,LOOP | | | | | | |
| | loop for R1 = 2DH: | | 00 | 12 | 38 | 45 | 67 | 45 |
| | loop for R1 = 2CH: | | 00 | 18 | 23 | 45 | 67 | 23 |
| | loop for R1 = 2BH: | | 08 | 01 | 23 | 45 | 67 | 01 |
| | CLR | A | 08 | 01 | 23 | 45 | 67 | 00 |
| | XCH | A,2AH | 00 | 01 | 23 | 45 | 67 | 08 |

**Figure 12. Shifting a BCD Number One Digit to the Right**

First, pointers R1 and R0 are set up to point to the two bytes containing the last four BCD digits. Then a loop is executed which leaves the last byte, location 2EH, holding the last two digits of the shifted number. The pointers are decremented, and the loop is repeated for location 2DH. The CJNE instruction (Compare and Jump if Not Equal) is a loop control that will be described later.

The loop is executed from LOOP to CJNE for R1 = 2EH, 2DH, 2CH and 2BH. At that point the digit that was originally shifted out on the right has propagated to location 2AH. Since that location should be left with 0s, the lost digit is moved to the Accumulator.

### EXTERNAL RAM

Table 5 shows a list of the Data Transfer instructions that access external Data Memory. Only indirect addressing can be used. The choice is whether to use a one-byte address, @Ri, where Ri can be either R0 or R1 of the selected register bank, or a two-byte address, @DPTR. The disadvantage to using 16-bit addresses if only a few K bytes of external RAM are involved is that 16-bit addresses use all 8 bits of Port 2 as address bus. On the other hand, 8-bit addresses allow one to address a few K bytes of RAM, as shown in Figure 5, without having to sacrifice all of Port 2.

All of these instructions execute in 2 μs, with a 12 MHz clock.

**Table 5. A List of the MCS®-51 Data Transfer Instructions that Access External Data Memory Space**

| Address Width | Mnemonic | Operation | Execution Time (μs) |
|---|---|---|---|
| 8 bits | MOVX A,@Ri | Read external RAM @Ri | 2 |
| 8 bits | MOVX @Ri,A | Write external RAM @Ri | 2 |
| 16 bits | MOVX A,@DPTR | Read external RAM @DPTR | 2 |
| 16 bits | MOVX @DPTR,A | Write external RAM @DPTR | 2 |

Note that in all external Data RAM accesses, the Accumulator is always either the destination or source of the data.

The read and write strobes to external RAM are activated only during the execution of a MOVX instruction. Normally these signals are inactive, and in fact if they're not going to be used at all, their pins are available as extra I/O lines. More about that later.

### LOOKUP TABLES

Table 6 shows the two instructions that are available for reading lookup tables in Program Memory. Since these instructions access only Program Memory, the lookup tables can only be read, not updated. The mnemonic is MOVC for "move constant".

If the table access is to external Program Memory, then the read strobe is $\overline{\text{PSEN}}$.

**Table 6. The MCS®-51 Lookup Table Read Instructions**

| Mnemonic | Operation | Execution Time (μs) |
|---|---|---|
| MOVC A,@A+DPTR | Read Pgm Memory at (A+DPTR) | 2 |
| MOVC A,@A+PC | Read Pgm Memory at (A+PC) | 2 |

The first MOVC instruction in Table 6 can accommodate a table of up to 256 entries, numbered 0 through 255. The number of the desired entry is loaded into the Accumulator, and the Data Pointer is set up to point to beginning of the table. Then

```
MOVC    A,@A+DPTR
```

copies the desired table entry into the Accumulator.

The other MOVC instruction works the same way, except the Program Counter (PC) is used as the table base, and the table is accessed through a subroutine. First the number of the desired entry is loaded into the Accumulator, and the subroutine is called:

```
        MOV     A,ENTRY_NUMBER
        CALL    TABLE
```

The subroutine "TABLE" would look like this:

```
TABLE:  MOVC    A,@A+PC
        RET
```

The table itself immediately follows the RET (return) instruction in Program Memory. This type of table can have up to 255 entries, numbered 1 through 255. Number 0 can not be used, because at the time the MOVC instruction is executed, the PC contains the address of the RET instruction. An entry numbered 0 would be the RET opcode itself.

## Boolean Instructions

MCS-51 devices contain a complete Boolean (single-bit) processor. The internal RAM contains 128 addressable bits, and the SFR space can support up to 128 other addressable bits. All of the port lines are bit-addressable, and each one can be treated as a separate single-bit port. The instructions that access these bits are not just conditional branches, but a complete menu of move, set, clear, complement, OR, and AND instructions. These kinds of bit operations are not easily obtained in other architectures with any amount of byte-oriented software.

**Table 7. A List of the MCS®-51 Boolean Instructions**

| Mnemonic | | Operation | Execution Time (μs) |
|---|---|---|---|
| ANL | C,bit | C = C .AND. bit | 2 |
| ANL | C,/bit | C = C .AND. .NOT. bit | 2 |
| ORL | C,bit | C = C .OR. bit | 2 |
| ORL | C,/bit | C = C .OR. .NOT. bit | 2 |
| MOV | C,bit | C = bit | 1 |
| MOV | bit,C | bit = C | 2 |
| CLR | C | C = 0 | 1 |
| CLR | bit | bit = 0 | 1 |
| SETB | C | C = 1 | 1 |
| SETB | bit | bit = 1 | 1 |
| CPL | C | C = .NOT. C | 1 |
| CPL | bit | bit = .NOT. bit | 1 |
| JC | rel | Jump if C = 1 | 2 |
| JNC | rel | Jump if C = 0 | 2 |
| JB | bit,rel | Jump if bit = 1 | 2 |
| JNB | bit,rel | Jump if bit = 0 | 2 |
| JBC | bit,rel | Jump if bit = 1; CLR bit | 2 |

The instruction set for the Boolean processor is shown in Table 7. All bit accesses are by direct addressing. Bit addresses 00H through 7FH are in the Lower 128, and bit addresses 80H through FFH are in SFR space.

Note how easily an internal flag can be moved to a port pin:

```
MOV     C,FLAG
MOV     P1.0,C
```

In this example, FLAG is the name of any addressable bit in the Lower 128 or SFR space. An I/O line (the LSB of Port 1, in this case) is set or cleared depending on whether the flag bit is 1 or 0.

The Carry bit in the PSW is used as the single-bit Accumulator of the Boolean processor. Bit instructions that refer to the Carry bit as C assemble as Carry-specific instructions (CLR C, etc). The Carry bit also has a direct address, since it resides in the PSW register, which is bit-addressable.

Note that the Boolean instruction set includes ANL and ORL operations, but not the XRL (Exclusive OR) operation. An XRL operation is simple to implement in software. Suppose, for example, it is required to form the Exclusive OR of two bits:

```
C = bit1 .XRL. bit2
```

The software to do that could be as follows:

```
        MOV     C,bit1
        JNB     bit2,OVER
        CPL     C
OVER:   (continue)
```

First, bit1 is moved to the Carry. If bit2 = 0, then C now contains the correct result. That is, bit1 .XRL. bit2 = bit1 if bit2 = 0. On the other hand, if bit2 = 1 C now contains the complement of the correct result. It need only be inverted (CPL C) to complete the operation.

This code uses the JNB instruction, one of a series of bit-test instructions which execute a jump if the addressed bit is set (JC, JB, JBC) or if the addressed bit is not set (JNC, JNB). In the above case, bit2 is being tested, and if bit2 = 0 the CPL C instruction is jumped over.

JBC executes the jump if the addressed bit is set, and also clears the bit. Thus a flag can be tested and cleared in one operation.

All the PSW bits are directly addressable, so the Parity bit, or the general purpose flags, for example, are also available to the bit-test instructions.

### RELATIVE OFFSET

The destination address for these jumps is specified to the assembler by a label or by an actual address in Program Memory. However, the destination address assembles to a **relative** offset byte. This is a signed (two's complement) offset byte which is added to the PC in two's complement arithmetic if the jump is executed.

The range of the jump is therefore −128 to +127 Program Memory bytes relative to the first byte following the instruction.

## Jump Instructions

Table 8 shows the list of unconditional jumps.

**Table 8. Unconditional Jumps in MCS®-51 Devices**

| Mnemonic | Operation | Execution Time (μs) |
|---|---|---|
| JMP addr | Jump to addr | 2 |
| JMP @A+DPTR | Jump to A+DPTR | 2 |
| CALL addr | Call subroutine at addr | 2 |
| RET | Return from subroutine | 2 |
| RETI | Return from interrupt | 2 |
| NOP | No operation | 1 |

The Table lists a single "JMP addr" instruction, but in fact there are three—SJMP, LJMP and AJMP—which differ in the format of the destination address. JMP is a generic mnemonic which can be used if the programmer does not care which way the jump is encoded.

The SJMP instruction encodes the destination address as a relative offset, as described above. The instruction is 2 bytes long, consisting of the opcode and the relative offset byte. The jump distance is limited to a range of −128 to +127 bytes relative to the instruction following the SJMP.

The LJMP instruction encodes the destination address as a 16-bit constant. The instruction is 3 bytes long, consisting of the opcode and two address bytes. The destination address can be anywhere in the 64K Program Memory space.

The AJMP instruction encodes the destination address as an 11-bit constant. The instruction is 2 bytes long, consisting of the opcode, which itself contains 3 of the 11 address bits, followed by another byte containing the low 8 bits of the destination address. When the instruction is executed, these 11 bits are simply substituted for the low 11 bits in the PC. The high 5 bits stay the same. Hence the destination has to be within the same 2K block as the instruction following the AJMP.

In all cases the programmer specifies the destination address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the destination address into the correct format for the given instruction. If the format required by the instruction will not support the distance to the specified destination address, a "Destination out of range" message is written into the List file.

The JMP @A+DPTR instruction supports case jumps. The destination address is computed at execution time as the sum of the 16-bit DPTR register and the Accumulator. Typically, DPTR is set up with the address of a jump table, and the Accumulator is given an index to the table. In a 5-way branch, for example, an integer 0 through 4 is loaded into the Accumulator. The code to be executed might be as follows:

```
MOV     DPTR,#JUMP_TABLE
MOV     A,INDEX_NUMBER
RL      A
JMP     @A+DPTR
```

The RL A instruction converts the index number (0 through 4) to an even number on the range 0 through 8, because each entry in the jump table is 2 bytes long:

```
JUMP_TABLE:
        AJMP    CASE_0
        AJMP    CASE_1
        AJMP    CASE_2
        AJMP    CASE_3
        AJMP    CASE_4
```

Table 8 shows a single "CALL addr" instruction, but there are two of them—LCALL and ACALL—which differ in the format in which the subroutine address is given to the CPU. CALL is a generic mnemonic which can be used if the programmer does not care which way the address is encoded.

The LCALL instruction uses the 16-bit address format, and the subroutine can be anywhere in the 64K Program Memory space. The ACALL instruction uses the 11-bit format, and the subroutine must be in the same 2K block as the instruction following the ACALL.

In any case the programmer specifies the subroutine address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the address into the correct format for the given instructions.

Subroutines should end with a RET instruction, which returns execution to the instruction following the CALL.

RETI is used to return from an interrupt service routine. The only difference between RET and RETI is that RETI tells the interrupt control system that the interrupt in progress is done. If there is no interrupt in progress at the time RETI is executed, then the RETI is functionally identical to RET.

Table 9 shows the list of conditional jumps available to the MCS-51 user. All of these jumps specify the destination address by the relative offset method, and so are limited to a jump distance of −128 to +127 bytes from the instruction following the conditional jump instruction. Important to note, however, the user specifies to the assembler the actual destination address the same way as the other jumps: as a label or a 16-bit constant.

**Table 9. Conditional Jumps in MCS®-51 Devices**

| Mnemonic | Operation | Addressing Modes | | | | Execution Time (μs) |
|---|---|---|---|---|---|---|
| | | Dir | Ind | Reg | Imm | |
| JZ rel | Jump if A = 0 | Accumulator only | | | | 2 |
| JNZ rel | Jump if A ≠ 0 | Accumulator only | | | | 2 |
| DJNZ <byte>,rel | Decrement and jump if not zero | X | | X | | 2 |
| CJNE A,<byte>,rel | Jump if A ≠ <byte> | X | | | X | 2 |
| CJNE <byte>,#data,rel | Jump if <byte> ≠ #data | | X | X | | 2 |

There is no Zero bit in the PSW. The JZ and JNZ instructions test the Accumulator data for that condition.

The DJNZ instruction (Decrement and Jump if Not Zero) is for loop control. To execute a loop N times, load a counter byte with N and terminate the loop with a DJNZ to the beginning of the loop, as shown below for N = 10:

```
        MOV     COUNTER,#10
LOOP:   (begin loop)
          .
          .
          .
        (end loop)
        DJNZ    COUNTER,LOOP
        (continue)
```

The CJNE instruction (Compare and Jump if Not Equal) can also be used for loop control as in Figure 12. Two bytes are specified in the operand field of the instruction. The jump is executed only if the two bytes are not equal. In the example of Figure 12, the two bytes were the data in R1 and the constant 2AH. The initial data in R1 was 2EH. Every time the loop was executed, R1 was decremented, and the looping was to continue until the R1 data reached 2AH.

Another application of this instruction is in "greater than, less than" comparisons. The two bytes in the operand field are taken as unsigned integers. If the first is less than the second, then the Carry bit is set (1). If the first is greater than or equal to the second, then the Carry bit is cleared.

## CPU TIMING

All MCS-51 microcontrollers have an on-chip oscillator which can be used if desired as the clock source for the CPU. To use the on-chip oscillator, connect a crystal or ceramic resonator between the XTAL1 and XTAL2 pins of the microcontroller, and capacitors to ground as shown in Figure 13.

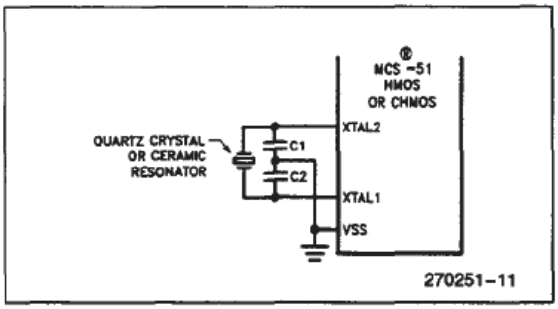



**Figure 13. Using the On-Chip Oscillator**

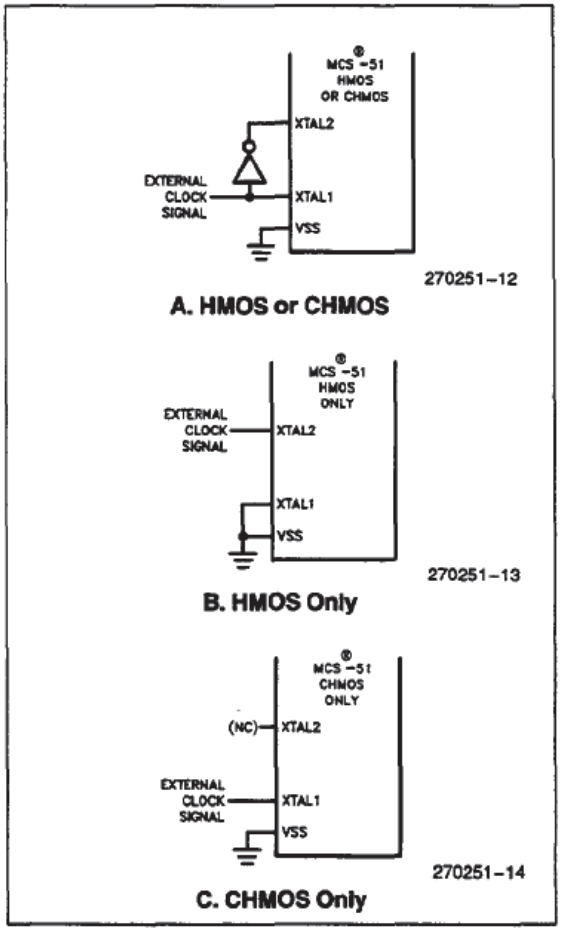



**Figure 14. Using an External Clock**

Examples of how to drive the clock with an external oscillator are shown in Figure 14. Note that in the HMOS devices (8051, etc.) the signal at the XTAL2 pin actually drives the internal clock generator. In the CHMOS devices (80C51BH, etc.) the signal at the XTAL1 pin drives the internal clock generator. If only one pin is going to be driven with the external oscillator signal, make sure it is the right pin.

The internal clock generator defines the sequence of states that make up the MCS-51 machine cycle.

## Machine Cycles

A machine cycle consists of a sequence of 6 states, numbered S1 through S6. Each state time lasts for two oscillator periods. Thus a machine cycle takes 12 oscillator periods or 1 μs if the oscillator frequency is 12 MHz.

Each state is divided into a Phase 1 half and a Phase 2 half. Figure 15 shows the fetch/execute sequences in

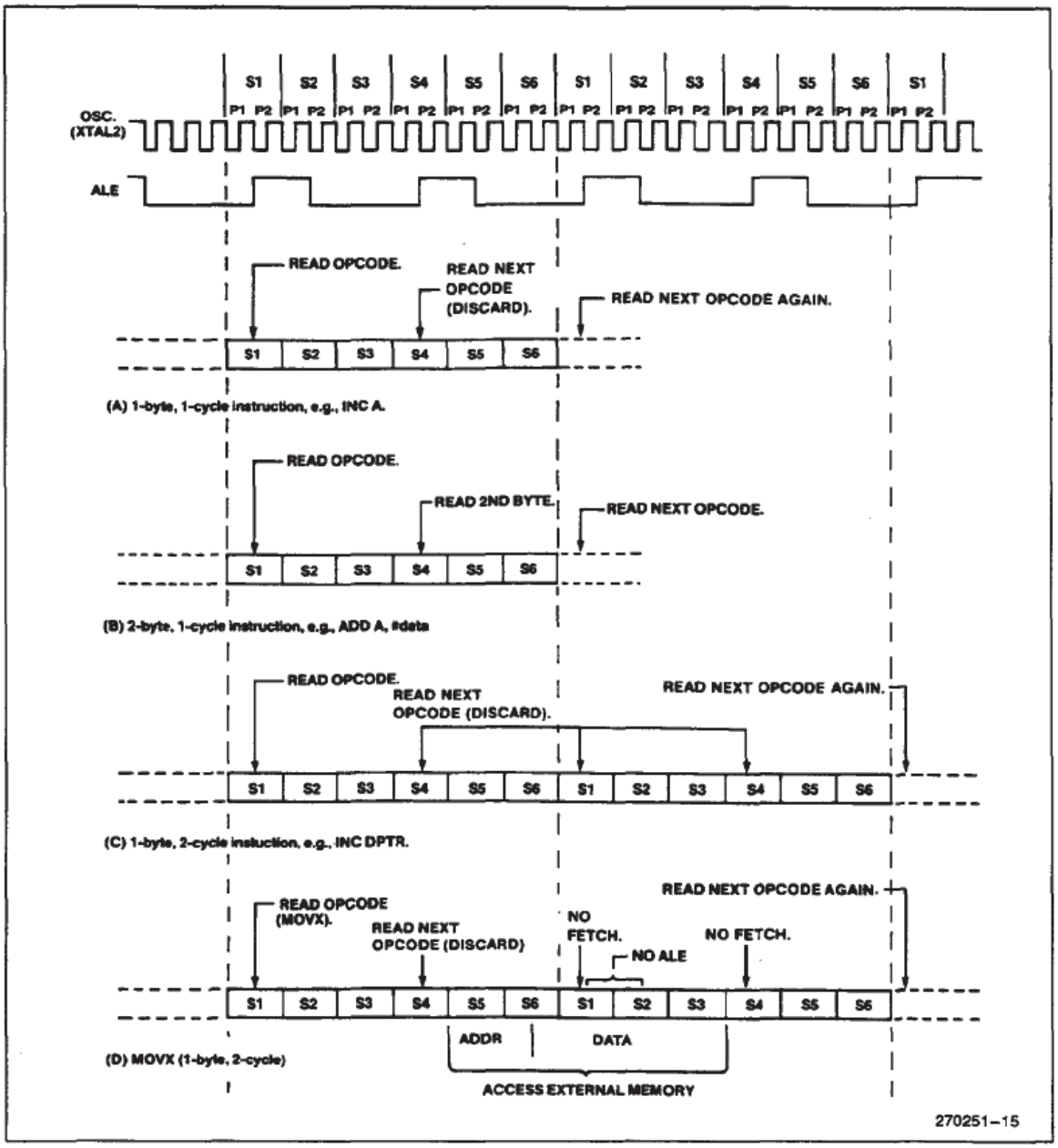



**Figure 15. State Sequences in MCS®-51 Devices**

states and phases for various kinds of instructions. Normally two program fetches are generated during each machine cycle, even if the instruction being executed doesn't require it. If the instruction being executed doesn't need more code bytes, the CPU simply ignores the extra fetch, and the Program Counter is not incremented.

Execution of a one-cycle instruction (Figure 15A and B) begins during State 1 of the machine cycle, when the opcode is latched into the Instruction Register. A second fetch occurs during S4 of the same machine cycle. Execution is complete at the end of State 6 of this machine cycle.

The MOVX instructions take two machine cycles to execute. No program fetch is generated during the second cycle of a MOVX instruction. This is the only time program fetches are skipped. The fetch/execute sequence for MOVX instructions is shown in Figure 15(D).

The fetch/execute sequences are the same whether the Program Memory is internal or external to the chip. Execution times do not depend on whether the Program Memory is internal or external.

Figure 16 shows the signals and timing involved in program fetches when the Program Memory is external. If Program Memory is external, then the Program Memory read strobe $\overline{PSEN}$ is normally activated twice per machine cycle, as shown in Figure 16(A).

If an access to external Data Memory occurs, as shown in Figure 16(B), two $\overline{PSEN}$s are skipped, because the address and data bus are being used for the Data Memory access.

Note that a Data Memory bus cycle takes twice as much time as a Program Memory bus cycle. Figure 16 shows the relative timing of the addresses being emitted at Ports 0 and 2, and of ALE and $\overline{PSEN}$. ALE is used to latch the low address byte from P0 into the address latch.

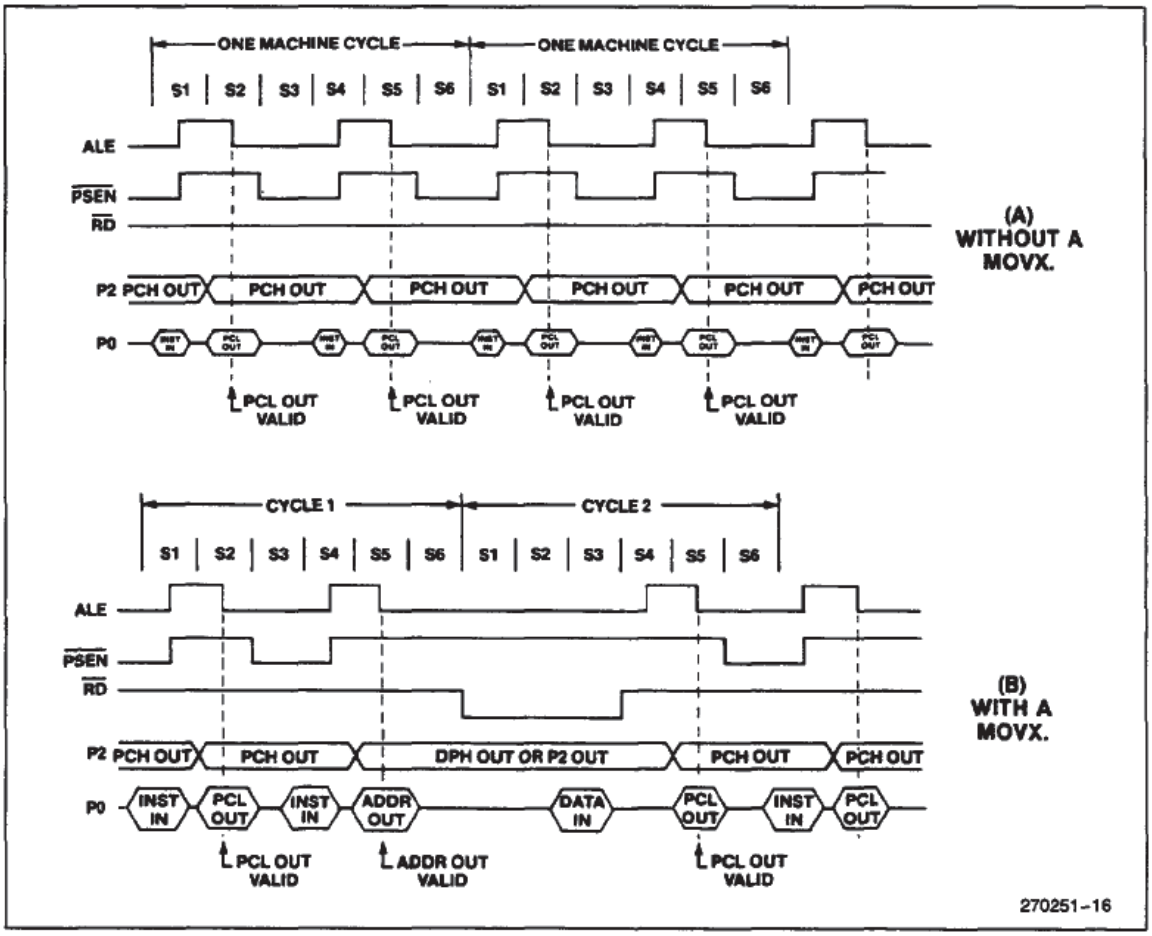



**Figure 16. Bus Cycles in MCS®-51 Devices Executing from External Program Memory**

When the CPU is executing from internal Program Memory, $\overline{PSEN}$ is not activated, and program addresses are not emitted. However, ALE continues to be activated twice per machine cycle and so is available as a clock output signal. Note, however, that one ALE is skipped during the execution of the MOVX instruction.

## Interrupt Structure

The 8051 core provides 5 interrupt sources: 2 external interrupts, 2 timer interrupts, and the serial port interrupt. What follows is an overview of the interrupt structure for the 8051. Other MCS-51 devices have additional interrupt sources and vectors as shown in Table 1. Refer to the appropriate chapters on other devices for further information on their interrupts.

### INTERRUPT ENABLES

Each of the interrupt sources can be individually enabled or disabled by setting or clearing a bit in the SFR named IE (Interrupt Enable). This register also contains a global disable bit, which can be cleared to disable all interrupts at once. Figure 17 shows the IE register for the 8051.

| (MSB) | | | | | | | (LSB) |
|---|---|---|---|---|---|---|---|
| EA | — | — | ES | ET1 | EX1 | ET0 | EX0 |

Enable bit = 1 enables the interrupt.
Enable bit = 0 disables it.

| Symbol | Position | Function |
|---|---|---|
| EA | IE.7 | disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit. |
| — | IE.6 | reserved* |
| — | IE.5 | reserved* |
| ES | IE.4 | Serial Port Interrupt enable bit. |
| ET1 | IE.3 | Timer 1 Overflow Interrupt enable bit. |
| EX1 | IE.2 | External Interrupt 1 enable bit. |
| ET0 | IE.1 | Timer 0 Overflow Interrupt enable bit. |
| EX0 | IE.0 | External Interrupt 0 enable bit. |

*These reserved bits are used in other MCS-51 devices.

**Figure 17. IE (Interrupt Enable) Register in the 8051**

### INTERRUPT PRIORITIES

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in the SFR named IP (Interrupt Priority). Figure 18 shows the IP register in the 8051.

A low-priority interrrupt can be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

If two interrupt requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence.

Figure 19 shows, for the 8051, how the IE and IP registers and the polling sequence work to determine which if any interrupt will be serviced.

| (MSB) | | | | | | | (LSB) |
|---|---|---|---|---|---|---|---|
| — | — | — | PS | PT1 | PX1 | PT0 | PX0 |

Priority bit = 1 assigns high priority.
Priority bit = 0 assigns low priority.

| Symbol | Position | Function |
|---|---|---|
| — | IP.7 | reserved* |
| — | IP.6 | reserved* |
| — | IP.5 | reserved* |
| PS | IP.4 | Serial Port interrupt priority bit. |
| PT1 | IP.3 | Timer 1 interrupt priority bit. |
| PX1 | IP.2 | External Interrupt 1 priority bit. |
| PT0 | IP.1 | Timer 0 interrupt priority bit. |
| PX0 | IP.0 | External Interrupt 0 priority bit. |

*These reserved bits are used in other MCS-51 devices.

**Figure 18. IP (Interrupt Priority) Register in the 8051**

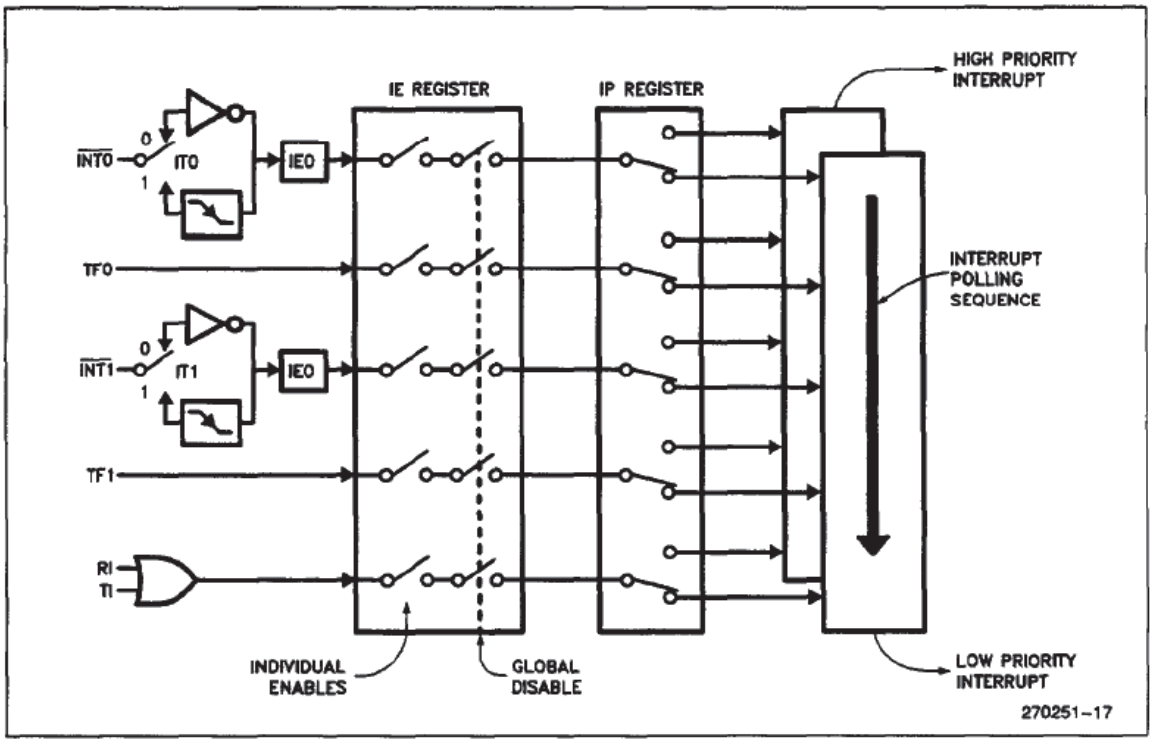



**Figure 19. 8051 Interrupt Control System**

In operation, all the interrupt flags are latched into the interrupt control system during State 5 of every machine cycle. The samples are polled during the following machine cycle. If the flag for an enabled interrupt is found to be set (1), the interrupt system generates an LCALL to the appropriate location in Program Memory, unless some other condition blocks the interrupt. Several conditions can block an interrupt, among them that an interrupt of equal or higher priority level is already in progress.

The hardware-generated LCALL causes the contents of the Program Counter to be pushed onto the stack, and reloads the PC with the beginning address of the service routine. As previously noted (Figure 3), the service routine for each interrupt begins at a fixed location.

Only the Program Counter is automatically pushed onto the stack, not the PSW or any other register. Having only the PC be automatically saved allows the programmer to decide how much time to spend saving which other registers. This enhances the interrupt response time, albeit at the expense of increasing the programmer's burden of responsibility. As a result, many interrupt functions that are typical in control applications—toggling a port pin, for example, or reloading a timer, or unloading a serial buffer—can often be completed in less time than it takes other architectures to commence them.

## SIMULATING A THIRD PRIORITY LEVEL IN SOFTWARE

Some applications require more than the two priority levels that are provided by on-chip hardware in MCS-51 devices. In these cases, relatively simple software can be written to produce the same effect as a third priority level.

First, interrupts that are to have higher priority than 1 are assigned to priority 1 in the IP (Interrupt Priority) register. The service routines for priority 1 interrupts that are supposed to be interruptible by "priority 2" interrupts are written to include the following code:

```
        PUSH    IE
        MOV     IE,#MASK
        CALL    LABEL
          *******
(execute service routine)
          *******
        POP     IE
        RET
LABEL:  RETI
```

As soon as any priority 1 interrupt is acknowledged, the IE (Interrupt Enable) register is re-defined so as to disable all but "priority 2" interrupts. Then, a CALL to LABEL executes the RETI instruction, which clears the priority 1 interrupt-in-progress flip-flop. At this point any priority 1 interrupt that is enabled can be serviced, but only "priority 2" interrupts are enabled.

POPping IE restores the original enable byte. Then a normal RET (rather than another RETI) is used to terminate the service routine. The additional software adds 10 μs (at 12 MHz) to priority 1 interrupts.

## ADDITIONAL REFERENCES

The following application notes are found in the *Embedded Control Applications* handbook. (Order Number: 270648)

1. AP-69 "An Introduction to the Intel MCS®-51 Single-Chip Microcomputer Family"
2. AP-70 "Using the Intel MCS®-51 Boolean Processing Capabilities"

# MCS® 51 Programmer's Guide and Instruction Set

2

# MCS® 51 PROGRAMMER'S GUIDE AND INSTRUCTION SET

CONTENTS PAGE

The information presented in this chapter is collected from the MCS®-51 Architectural Overview and the Hardware Description of the 8051, 8052 and 80C51 chapters of this book. The material has been selected and rearranged to form a quick and convenient reference for the programmers of the MCS-51. This guide pertains specifically to the 8051, 8052 and 80C51.

## MEMORY ORGANIZATION

## PROGRAM MEMORY

The 8051 has separate address spaces for Program Memory and Data Memory. The Program Memory can be up to 64K bytes long. The lower 4K (8K for the 8052) may reside on-chip.

Figure 1 shows a map of the 8051 program memory, and Figure 2 shows a map of the 8052 program memory.

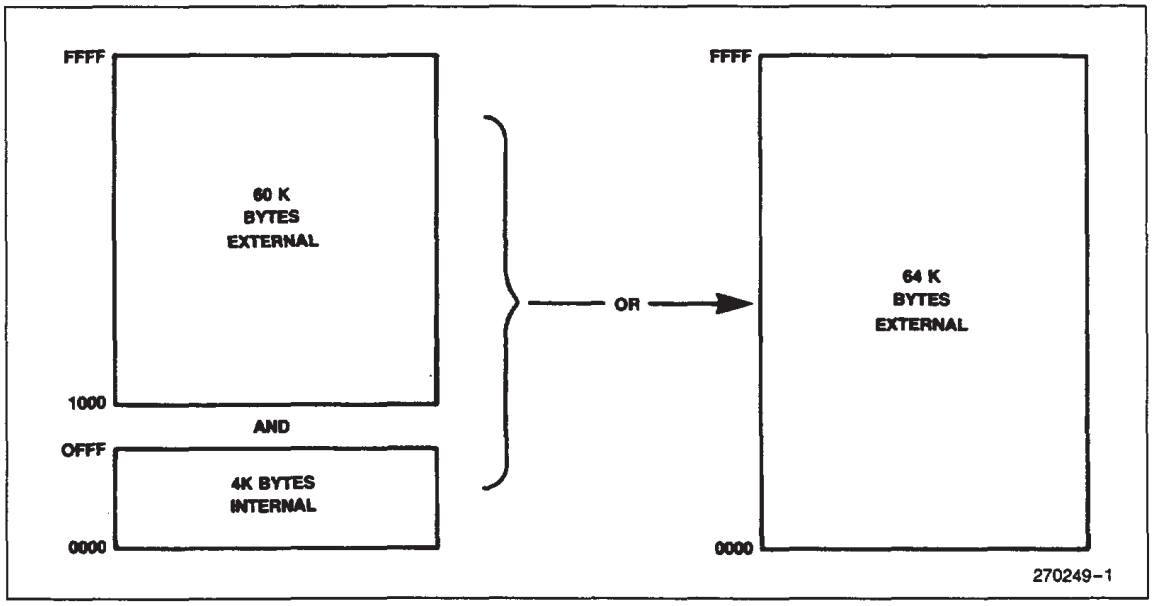



**Figure 1. The 8051 Program Memory**

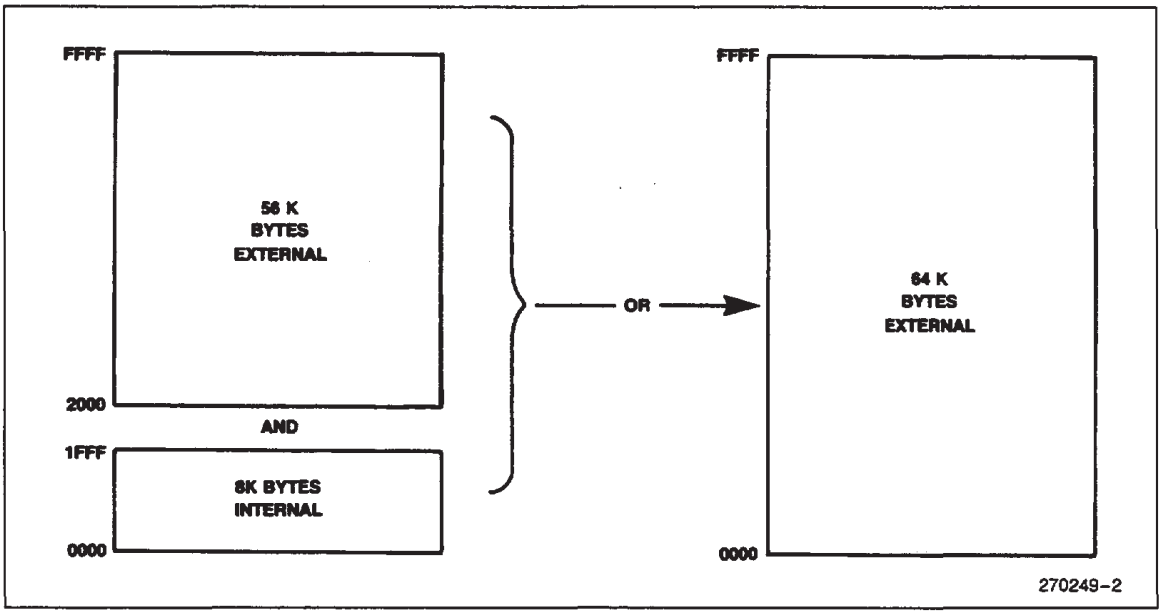



**Figure 2. The 8052 Program Memory**

## Data Memory:

The 8051 can address up to 64K bytes of Data Memory external to the chip. The "MOVX" instruction is used to access the external data memory. (Refer to the MCS-51 Instruction Set, in this chapter, for detailed description of instructions).

The 8051 has 128 bytes of on-chip RAM (256 bytes in the 8052) plus a number of Special Function Registers (SFRs). The lower 128 bytes of RAM can be accessed either by direct addressing (MOV data addr) or by indirect addressing (MOV @Ri). Figure 3 shows the 8051 and the 8052 Data Memory organization.

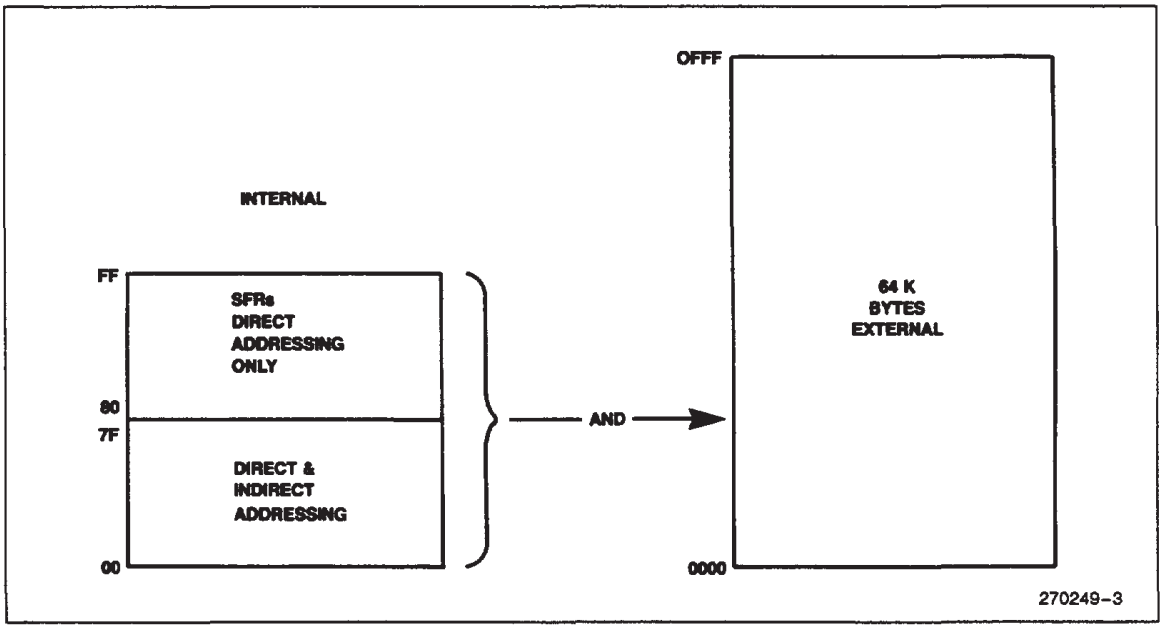



Figure 3a. The 8051 Data Memory

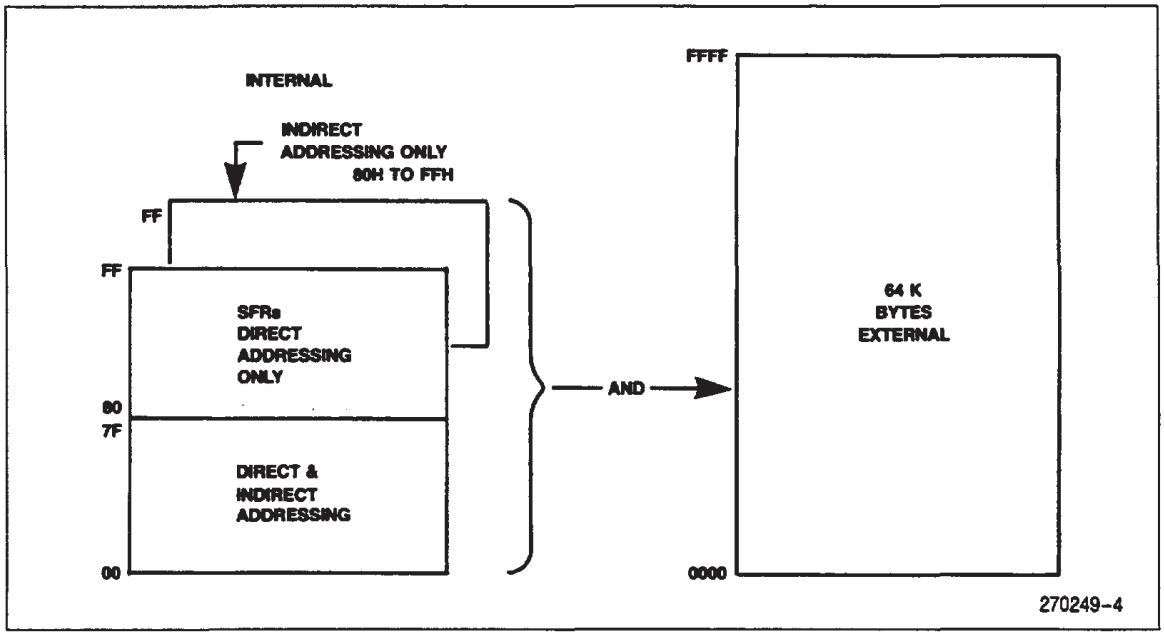



Figure 3b. The 8052 Data Memory

## INDIRECT ADDRESS AREA:

Note that in Figure 3b the SFRs and the indirect address RAM have the same addresses (80H–0FFH). Nevertheless, they are two separate areas and are accessed in two different ways.

For example the instruction

```
MOV    80H,#0AAH
```

writes 0AAH to Port 0 which is one of the SFRs and the instruction

```
MOV    R0,#80H

MOV    @R0,#0BBH
```

writes 0BBH in location 80H of the data RAM. Thus, after execution of both of the above instructions Port 0 will contain 0AAH and location 80 of the RAM will contain 0BBH.

Note that the stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space in those devices which implement 256 bytes of internal RAM.

## DIRECT AND INDIRECT ADDRESS AREA:

The 128 bytes of RAM which can be accessed by both direct and indirect addressing can be divided into 3 segments as listed below and shown in Figure 4.

**1. Register Banks 0-3:** Locations 0 through 1FH (32 bytes). ASM-51 and the device after reset default to register bank 0. To use the other register banks the user must select them in the software (refer to the MCS-51 Micro Assembler User's Guide). Each register bank contains 8 one-byte registers, 0 through 7.

Reset initializes the Stack Pointer to location 07H and it is incremented once to start from location 08H which is the first register (RO) of the second register bank. Thus, in order to use more than one register bank, the SP should be intialized to a different location of the RAM where it is not used for data storage (ie, higher part of the RAM).

**2. Bit Addressable Area:** 16 bytes have been assigned for this segment, 20H-2FH. Each one of the 128 bits of this segment can be directly addressed (0-7FH).

The bits can be referred to in two ways both of which are acceptable by the ASM-51. One way is to refer to their addresses, ie. 0 to 7FH. The other way is with reference to bytes 20H to 2FH. Thus, bits 0–7 can also be referred to as bits 20.0–20.7, and bits 8-FH are the same as 21.0–21.7 and so on.

Each of the 16 bytes in this segment can also be addressed as a byte.

**3. Scratch Pad Area:** Bytes 30H through 7FH are available to the user as data RAM. However, if the stack pointer has been initialized to this area, enough number of bytes should be left aside to prevent SP data destruction.

Figure 4 shows the different segments of the on-chip RAM.

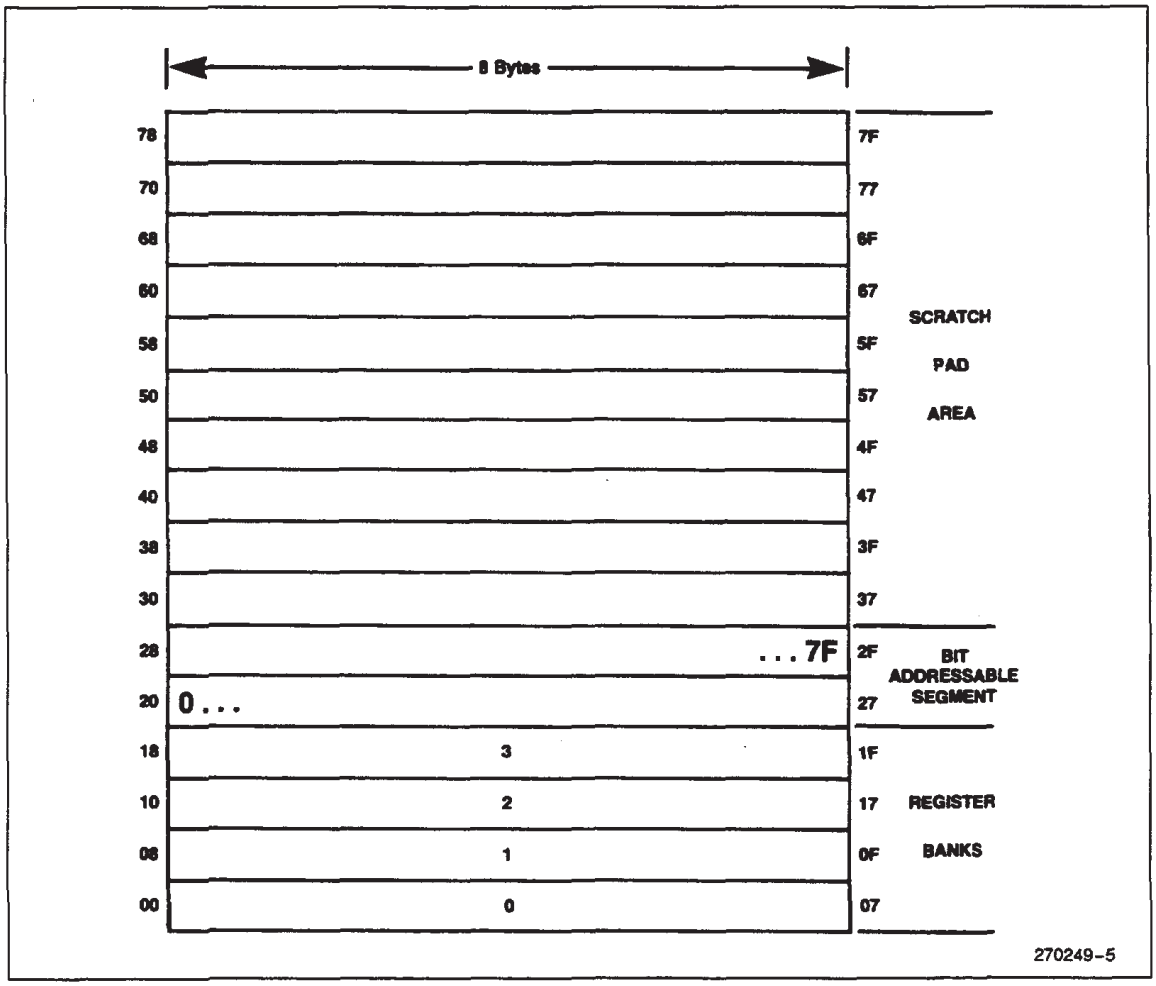



**Figure 4. 128 Bytes of RAM Direct and Indirect Addressable**

## SPECIAL FUNCTION REGISTERS:

Table 1 contains a list of all the SFRs and their addresses.

Comparing Table 1 and Figure 5 shows that all of the SFRs that are byte and bit addressable are located on the first column of the diagram in Figure 5.

**Table 1**

| Symbol | Name | Address |
|---|---|---|
| *ACC | Accumulator | 0E0H |
| *B | B Register | 0F0H |
| *PSW | Program Status Word | 0D0H |
| SP | Stack Pointer | 81H |
| DPTR | Data Pointer 2 Bytes | |
| DPL | Low Byte | 82H |
| DPH | High Byte | 83H |
| *P0 | Port 0 | 80H |
| *P1 | Port 1 | 90H |
| *P2 | Port 2 | 0A0H |
| *P3 | Port 3 | 0B0H |
| *IP | Interrupt Priority Control | 0B8H |
| *IE | Interrupt Enable Control | 0A8H |
| TMOD | Timer/Counter Mode Control | 89H |
| *TCON | Timer/Counter Control | 88H |
| *+T2CON | Timer/Counter 2 Control | 0C8H |
| TH0 | Timer/Counter 0 High Byte | 8CH |
| TL0 | Timer/Counter 0 Low Byte | 8AH |
| TH1 | Timer/Counter 1 High Byte | 8DH |
| TL1 | Timer/Counter 1 Low Byte | 8BH |
| +TH2 | Timer/Counter 2 High Byte | 0CDH |
| +TL2 | Timer/Counter 2 Low Byte | 0CCH |
| +RCAP2H | T/C 2 Capture Reg. High Byte | 0CBH |
| +RCAP2L | T/C 2 Capture Reg. Low Byte | 0CAH |
| *SCON | Serial Control | 98H |
| SBUF | Serial Data Buffer | 99H |
| PCON | Power Control | 87H |

* = Bit addressable
+ = 8052 only

## WHAT DO THE SFRs CONTAIN JUST AFTER POWER-ON OR A RESET?

Table 2 lists the contents of each SFR after power-on or a hardware reset.

**Table 2. Contents of the SFRs after reset**

| Register | Value in Binary |
|---|---|
| *ACC | 00000000 |
| *B | 00000000 |
| *PSW | 00000000 |
| SP | 00000111 |
| DPTR | |
| DPH | 00000000 |
| DPL | 00000000 |
| *P0 | 11111111 |
| *P1 | 11111111 |
| *P2 | 11111111 |
| *P3 | 11111111 |
| *IP | 8051 XXX00000,<br>8052 XX000000 |
| *IE | 8051 0XX00000,<br>8052 0X000000 |
| TMOD | 00000000 |
| *TCON | 00000000 |
| *+T2CON | 00000000 |
| TH0 | 00000000 |
| TL0 | 00000000 |
| TH1 | 00000000 |
| TL1 | 00000000 |
| +TH2 | 00000000 |
| +TL2 | 00000000 |
| +RCAP2H | 00000000 |
| +RCAP2L | 00000000 |
| *SCON | 00000000 |
| SBUF | Indeterminate |
| PCON | HMOS 0XXXXXXX<br>CHMOS 0XXX0000 |

X = Undefined
* = Bit Addressable
+ = 8052 only

## SFR MEMORY MAP

8 Bytes

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| F8 | | | | | | | | | FF |
| F0 | B | | | | | | | | F7 |
| E8 | | | | | | | | | EF |
| E0 | ACC | | | | | | | | E7 |
| D8 | | | | | | | | | DF |
| D0 | PSW | | | | | | | | D7 |
| C8 | T2CON | | RCAP2L | RCAP2H | TL2 | TH2 | | | CF |
| C0 | | | | | | | | | C7 |
| B8 | IP | | | | | | | | BF |
| B0 | P3 | | | | | | | | B7 |
| A8 | IE | | | | | | | | AF |
| A0 | P2 | | | | | | | | A7 |
| 98 | SCON | SBUF | | | | | | | 9F |
| 90 | P1 | | | | | | | | 97 |
| 88 | TCON | TMOD | TL0 | TL1 | TH0 | TH1 | | | 8F |
| 80 | P0 | SP | DPL | DPH | | | | PCON | 87 |

↑
Bit
Addressable

Figure 5

Those SFRs that have their bits assigned for various functions are listed in this section. A brief description of each bit is provided for quick reference. For more detailed information refer to the Architecture Chapter of this book.

## PSW: PROGRAM STATUS WORD. BIT ADDRESSABLE.

| CY | AC | F0 | RS1 | RS0 | OV | — | P |
|---|---|---|---|---|---|---|---|

| | | |
|---|---|---|
| CY | PSW.7 | Carry Flag. |
| AC | PSW.6 | Auxiliary Carry Flag. |
| F0 | PSW.5 | Flag 0 available to the user for general purpose. |
| RS1 | PSW.4 | Register Bank selector bit 1 (SEE NOTE 1). |
| RS0 | PSW.3 | Register Bank selector bit 0 (SEE NOTE 1). |
| OV | PSW.2 | Overflow Flag. |
| — | PSW.1 | User definable flag. |
| P | PSW.0 | Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of '1' bits in the accumulator. |

**NOTE:**
1. The value presented by RS0 and RS1 selects the corresponding register bank.

| RS1 | RS0 | Register Bank | Address |
|---|---|---|---|
| 0 | 0 | 0 | 00H-07H |
| 0 | 1 | 1 | 08H-0FH |
| 1 | 0 | 2 | 10H-17H |
| 1 | 1 | 3 | 18H-1FH |

## PCON: POWER CONTROL REGISTER. NOT BIT ADDRESSABLE.

| SMOD | — | — | — | GF1 | GF0 | PD | IDL |
|---|---|---|---|---|---|---|---|

| | |
|---|---|
| SMOD | Double baud rate bit. If Timer 1 is used to generate baud rate and SMOD = 1, the baud rate is doubled when the Serial Port is used in modes 1, 2, or 3. |
| — | Not implemented, reserved for future use.* |
| — | Not implemented, reserved for future use.* |
| — | Not implemented, reserved for future use.* |
| GF1 | General purpose flag bit. |
| GF0 | General purpose flag bit. |
| PD | Power Down bit. Setting this bit activates Power Down operation in the 80C51BH. (Available only in CHMOS). |
| IDL | Idle Mode bit. Setting this bit activates Idle Mode operation in the 80C51BH. (Available only in CHMOS). |

If 1s are written to PD and IDL at the same time, PD takes precedence.

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

## INTERRUPTS:

In order to use any of the interrupts in the MCS-51, the following three steps must be taken.

1. Set the EA (enable all) bit in the IE register to 1.
2. Set the corresponding individual interrupt enable bit in the IE register to 1.
3. Begin the interrupt service routine at the corresponding Vector Address of that interrupt. See Table below.

| Interrupt Source | Vector Address |
|---|---|
| IE0 | 0003H |
| TF0 | 000BH |
| IE1 | 0013H |
| TF1 | 001BH |
| RI & TI | 0023H |
| TF2 & EXF2 | 002BH |

In addition, for external interrupts, pins $\overline{INT0}$ and $\overline{INT1}$ (P3.2 and P3.3) must be set to 1, and depending on whether the interrupt is to be level or transition activated, bits IT0 or IT1 in the TCON register may need to be set to 1.

ITx = 0 level activated

ITx = 1 transition activated

## IE: INTERRUPT ENABLE REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

| EA | — | ET2 | ES | ET1 | EX1 | ET0 | EX0 |
|---|---|---|---|---|---|---|---|

| | | |
|---|---|---|
| EA | IE.7 | Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit. |
| — | IE.6 | Not implemented, reserved for future use.* |
| ET2 | IE.5 | Enable or disable the Timer 2 overflow or capture interrupt (8052 only). |
| ES | IE.4 | Enable or disable the serial port interrupt. |
| ET1 | IE.3 | Enable or disable the Timer 1 overflow interrupt. |
| EX1 | IE.2 | Enable or disable External Interrupt 1. |
| ET0 | IE.1 | Enable or disable the Timer 0 overflow interrupt. |
| EX0 | IE.0 | Enable or disable External Interrupt 0. |

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

## ASSIGNING HIGHER PRIORITY TO ONE OR MORE INTERRUPTS:

In order to assign higher priority to an interrupt the corresponding bit in the IP register must be set to 1.

Remember that while an interrupt service is in progress, it cannot be interrupted by a lower or same level interrupt.

## PRIORITY WITHIN LEVEL:

Priority within level is only to resolve simultaneous requests of the same priority level.

From high to low, interrupt sources are listed below:

IE0
TF0
IE1
TF1
RI or TI
TF2 or EXF2

## IP: INTERRUPT PRIORITY REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt has a lower priority and if the bit is 1 the corresponding interrupt has a higher priority.

| — | — | PT2 | PS | PT1 | PX1 | PT0 | PX0 |
|---|---|---|---|---|---|---|---|

— IP. 7 Not implemented, reserved for future use.*
— IP. 6 Not implemented, reserved for future use.*
PT2 IP. 5 Defines the Timer 2 interrupt priority level (8052 only).
PS IP. 4 Defines the Serial Port interrupt priority level.
PT1 IP. 3 Defines the Timer 1 interrupt priority level.
PX1 IP. 2 Defines External Interrupt 1 priority level.
PT0 IP. 1 Defines the Timer 0 interrupt priority level.
PX0 IP. 0 Defines the External Interrupt 0 priority level.

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

## TCON: TIMER/COUNTER CONTROL REGISTER. BIT ADDRESSABLE.

| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
|---|---|---|---|---|---|---|---|

TF1 TCON. 7 Timer 1 overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.

TR1 TCON. 6 Timer 1 run control bit. Set/cleared by software to turn Timer/Counter 1 ON/OFF.

TF0 TCON. 5 Timer 0 overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the service routine.

TR0 TCON. 4 Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.

IE1 TCON. 3 External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.

IT1 TCON. 2 Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

IE0 TCON. 1 External Interrupt 0 edge flag. Set by hardware when External Interrupt edge detected. Cleared by hardware when interrupt is processed.

IT0 TCON. 0 Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

## TMOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE.

| GATE | C/$\overline{T}$ | M1 | M0 | GATE | C/$\overline{T}$ | M1 | M0 |
|---|---|---|---|---|---|---|---|
| TIMER 1 | | | | TIMER 0 | | | |

GATE When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).

C/$\overline{T}$ Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).

M1 Mode selector bit. (NOTE 1)

M0 Mode selector bit. (NOTE 1)

**NOTE 1:**

| M1 | M0 | | Operating Mode |
|---|---|---|---|
| 0 | 0 | 0 | 13-bit Timer (MCS-48 compatible) |
| 0 | 1 | 1 | 16-bit Timer/Counter |
| 1 | 0 | 2 | 8-bit Auto-Reload Timer/Counter |
| 1 | 1 | 3 | (Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits, TH0 is an 8-bit Timer and is controlled by Timer 1 control bits. |
| 1 | 1 | 3 | (Timer 1) Timer/Counter 1 stopped. |

## TIMER SET-UP

Tables 3 through 6 give some values for TMOD which can be used to set up Timer 0 in different modes.

It is assumed that only one timer is being used at a time. If it is desired to run Timers 0 and 1 simultaneously, in any mode, the value in TMOD for Timer 0 must be ORed with the value shown for Timer 1 (Tables 5 and 6).

For example, if it is desired to run Timer 0 in mode 1 GATE (external control), and Timer 1 in mode 2 COUNTER, then the value that must be loaded into TMOD is 69H (09H from Table 3 ORed with 60H from Table 6).

Moreover, it is assumed that the user, at this point, is not ready to turn the timers on and will do that at a different point in the program by setting bit TRx (in TCON) to 1.

## TIMER/COUNTER 0

### As a Timer:

Table 3

| MODE | TIMER 0 FUNCTION | TMOD | |
|---|---|---|---|
| | | INTERNAL CONTROL (NOTE 1) | EXTERNAL CONTROL (NOTE 2) |
| 0 | 13-bit Timer | 00H | 08H |
| 1 | 16-bit Timer | 01H | 09H |
| 2 | 8-bit Auto-Reload | 02H | 0AH |
| 3 | two 8-bit Timers | 03H | 0BH |

### As a Counter:

Table 4

| MODE | COUNTER 0 FUNCTION | TMOD | |
|---|---|---|---|
| | | INTERNAL CONTROL (NOTE 1) | EXTERNAL CONTROL (NOTE 2) |
| 0 | 13-bit Timer | 04H | 0CH |
| 1 | 16-bit Timer | 05H | 0DH |
| 2 | 8-bit Auto-Reload | 06H | 0EH |
| 3 | one 8-bit Counter | 07H | 0FH |

**NOTES:**
1. The Timer is turned ON/OFF by setting/clearing bit TR0 in the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on $\overline{INT0}$ (P3.2) when TR0 = 1 (hardware control).

## TIMER/COUNTER 1

### As a Timer:

Table 5

| MODE | TIMER 1 FUNCTION | TMOD | |
|---|---|---|---|
| | | INTERNAL CONTROL (NOTE 1) | EXTERNAL CONTROL (NOTE 2) |
| 0 | 13-bit Timer | 00H | 80H |
| 1 | 16-bit Timer | 10H | 90H |
| 2 | 8-bit Auto-Reload | 20H | A0H |
| 3 | does not run | 30H | B0H |

### As a Counter:

Table 6

| MODE | COUNTER 1 FUNCTION | TMOD | |
|---|---|---|---|
| | | INTERNAL CONTROL (NOTE 1) | EXTERNAL CONTROL (NOTE 2) |
| 0 | 13-bit Timer | 40H | C0H |
| 1 | 16-bit Timer | 50H | D0H |
| 2 | 8-bit Auto-Reload | 60H | E0H |
| 3 | not available | — | — |

**NOTES:**
1. The Timer is turned ON/OFF by setting/clearing bit TR1 in the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on $\overline{INT1}$ (P3.3) when TR1 = 1 (hardware control).

## T2CON: TIMER/COUNTER 2 CONTROL REGISTER. BIT ADDRESSABLE

### 8052 Only

| TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/$\overline{T2}$ | CP/$\overline{RL2}$ |
|---|---|---|---|---|---|---|---|

TF2 T2CON. 7 Timer 2 overflow flag set by hardware and cleared by software. TF2 cannot be set when either RCLK = 1 or CLK = 1

EXF2 T2CON. 6 Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX, and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.

RCLK T2CON. 5 Receive clock flag. When set, causes the Serial Port to use Timer 2 overflow pulses for its receive clock in modes 1 & 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.

TLCK T2CON. 4 Transmit clock flag. When set, causes the Serial Port to use Timer 2 overflow pulses for its transmit clock in modes 1 & 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.

EXEN2 T2CON. 3 Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of negative transition on T2EX if Timer 2 is not being used to clock the Serial Port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.

TR2 T2CON. 2 Software START/STOP control for Timer 2. A logic 1 starts the Timer.

C/$\overline{T2}$ T2CON. 1 Timer or Counter select.

0 = Internal Timer. 1 = External Event Counter (falling edge triggered).

CP/$\overline{RL2}$ T2CON. 0 Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, Auto-Reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the Timer is forced to Auto-Reload on Timer 2 overflow.

## TIMER/COUNTER 2 SET-UP

Except for the baud rate generator mode, the values given for T2CON do not include the setting of the TR2 bit. Therefore, bit TR2 must be set, separately, to turn the Timer on.

### As a Timer:

Table 7

| MODE | T2CON | |
|---|---|---|
| | INTERNAL CONTROL (NOTE 1) | EXTERNAL CONTROL (NOTE 2) |
| 16-bit Auto-Reload | 00H | 08H |
| 16-bit Capture | 01H | 09H |
| BAUD rate generator receive & transmit same baud rate | 34H | 36H |
| receive only | 24H | 26H |
| transmit only | 14H | 16H |

### As a Counter:

Table 8

| MODE | TMOD | |
|---|---|---|
| | INTERNAL CONTROL (NOTE 1) | EXTERNAL CONTROL (NOTE 2) |
| 16-bit Auto-Reload | 02H | 0AH |
| 16-bit Capture | 03H | 0BH |

**NOTES:**
1. Capture/Reload occurs only on Timer/Counter overflow.
2. Capture/Reload occurs on Timer/Counter overflow and a 1 to 0 transition on T2EX (P1.1) pin except when Timer 2 is used in the baud rate generating mode.

## SCON: SERIAL PORT CONTROL REGISTER. BIT ADDRESSABLE.

| SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |
|---|---|---|---|---|---|---|---|

SM0 SCON. 7 Serial Port mode specifier. (NOTE 1).

SM1 SCON. 6 Serial Port mode specifier. (NOTE 1).

SM2 SCON. 5 Enables the multiprocessor communication feature in modes 2 & 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0. (See Table 9).

REN SCON. 4 Set/Cleared by software to Enable/Disable reception.

TB8 SCON. 3 The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.

RB8 SCON. 2 In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.

TI SCON. 1 Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.

RI SCON. 0 Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes (except see SM2). Must be cleared by software.

**NOTE 1:**

| SM0 | SM1 | Mode | Description | Baud Rate |
|---|---|---|---|---|
| 0 | 0 | 0 | SHIFT REGISTER | Fosc./12 |
| 0 | 1 | 1 | 8-Bit UART | Variable |
| 1 | 0 | 2 | 9-Bit UART | Fosc./64 OR Fosc./32 |
| 1 | 1 | 3 | 9-Bit UART | Variable |

## SERIAL PORT SET-UP:

Table 9

| MODE | SCON | SM2 VARIATION |
|---|---|---|
| 0 | 10H | Single Processor Environment (SM2 = 0) |
| 1 | 50H | |
| 2 | 90H | |
| 3 | D0H | |
| 0 | NA | Multiprocessor Environment (SM2 = 1) |
| 1 | 70H | |
| 2 | B0H | |
| 3 | F0H | |

## GENERATING BAUD RATES

### Serial Port in Mode 0:

Mode 0 has a fixed baud rate which is 1/12 of the oscillator frequency. To run the serial port in this mode none of the Timer/Counters need to be set up. Only the SCON register needs to be defined.

$$\text{Baud Rate} = \frac{\text{Osc Freq}}{12}$$

### Serial Port in Mode 1:

Mode 1 has a variable baud rate. The baud rate can be generated by either Timer 1 or Timer 2 (8052 only).

## USING TIMER/COUNTER 1 TO GENERATE BAUD RATES:

For this purpose, Timer 1 is used in mode 2 (Auto-Reload). Refer to Timer Setup section of this chapter.

$$\text{Baud Rate} = \frac{\text{K x Oscillator Freq.}}{32 \times 12 \times [256 - (\text{TH1})]}$$

If SMOD = 0, then K = 1.
If SMOD = 1, then K = 2. (SMOD is the PCON register).

Most of the time the user knows the baud rate and needs to know the reload value for TH1.
Therefore, the equation to calculate TH1 can be written as:

$$\text{TH1} = 256 - \frac{\text{K x Osc Freq.}}{384 \times \text{baud rate}}$$

TH1 must be an integer value. Rounding off TH1 to the nearest integer may not produce the desired baud rate. In this case, the user may have to choose another crystal frequency.

Since the PCON register is not bit addressable, one way to set the bit is logical ORing the PCON register. (ie, ORL PCON,#80H). The address of PCON is 87H.

## USING TIMER/COUNTER 2 TO GENERATE BAUD RATES:

For this purpose, Timer 2 must be used in the baud rate generating mode. Refer to Timer 2 Setup Table in this chapter. If Timer 2 is being clocked through pin T2 (P1.0) the baud rate is:

$$\text{Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

And if it is being clocked internally the baud rate is:

$$\text{Baud Rate} = \frac{\text{Osc Freq}}{32 \times [65536 - (\text{RCAP2H, RCAP2L})]}$$

To obtain the reload value for RCAP2H and RCAP2L the above equation can be rewritten as:

$$\text{RCAP2H, RCAP2L} = 65536 - \frac{\text{Osc Freq}}{32 \times \text{Baud Rate}}$$

## SERIAL PORT IN MODE 2:

The baud rate is fixed in this mode and is $1/32$ or $1/64$ of the oscillator frequency depending on the value of the SMOD bit in the PCON register.

In this mode none of the Timers are used and the clock comes from the internal phase 2 clock.

SMOD = 1, Baud Rate = $1/32$ Osc Freq.

SMOD = 0, Baud Rate = $1/64$ Osc Freq.

To set the SMOD bit: ORL PCON,#80H. The address of PCON is 87H.

## SERIAL PORT IN MODE 3:

The baud rate in mode 3 is variable and sets up exactly the same as in mode 1.

# MCS®-51 INSTRUCTION SET

**Table 10. 8051 Instruction Set Summary**

Interrupt Response Time: Refer to Hardware Description Chapter.

**Instructions that Affect Flag Settings(1)**

| Instruction | Flag | | | Instruction | Flag | | |
|---|---|---|---|---|---|---|---|
| | C | OV | AC | | C | OV | AC |
| ADD | X | X | X | CLR C | O | | |
| ADDC | X | X | X | CPL C | X | | |
| SUBB | X | X | X | ANL C,bit | X | | |
| MUL | O | X | | ANL C,/bit | X | | |
| DIV | O | X | | ORL C,bit | X | | |
| DA | X | | | ORL C,bit | X | | |
| RRC | X | | | MOV C,bit | X | | |
| RLC | X | | | CJNE | X | | |
| SETB C | 1 | | | | | | |

(1)Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

**Note on instruction set and addressing modes:**

Rn — Register R7–R0 of the currently selected Register Bank.

direct — 8-bit internal data location's address. This could be an Internal Data RAM location (0–127) or a SFR [i.e., I/O port, control register, status register, etc. (128–255)].

@Ri — 8-bit internal data RAM location (0–255) addressed indirectly through register R1 or R0.

#data — 8-bit constant included in instruction.

#data 16 — 16-bit constant included in instruction.

addr 16 — 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.

addr 11 — 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.

rel — Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to +127 bytes relative to first byte of the following instruction.

bit — Direct Addressed bit in Internal Data RAM or Special Function Register.

| Mnemonic | | Description | Byte | Oscillator Period |
|---|---|---|---|---|
| **ARITHMETIC OPERATIONS** | | | | |
| ADD | A,Rn | Add register to Accumulator | 1 | 12 |
| ADD | A,direct | Add direct byte to Accumulator | 2 | 12 |
| ADD | A,@Ri | Add indirect RAM to Accumulator | 1 | 12 |
| ADD | A,#data | Add immediate data to Accumulator | 2 | 12 |
| ADDC | A,Rn | Add register to Accumulator with Carry | 1 | 12 |
| ADDC | A,direct | Add direct byte to Accumulator with Carry | 2 | 12 |
| ADDC | A,@Ri | Add indirect RAM to Accumulator with Carry | 1 | 12 |
| ADDC | A,#data | Add immediate data to Acc with Carry | 2 | 12 |
| SUBB | A,Rn | Subtract Register from Acc with borrow | 1 | 12 |
| SUBB | A,direct | Subtract direct byte from Acc with borrow | 2 | 12 |
| SUBB | A,@Ri | Subtract indirect RAM from ACC with borrow | 1 | 12 |
| SUBB | A,#data | Subtract immediate data from Acc with borrow | 2 | 12 |
| INC | A | Increment Accumulator | 1 | 12 |
| INC | Rn | Increment register | 1 | 12 |
| INC | direct | Increment direct byte | 2 | 12 |
| INC | @Ri | Increment direct RAM | 1 | 12 |
| DEC | A | Decrement Accumulator | 1 | 12 |
| DEC | Rn | Decrement Register | 1 | 12 |
| DEC | direct | Decrement direct byte | 2 | 12 |
| DEC | @Ri | Decrement indirect RAM | 1 | 12 |

**Table 10. 8051 Instruction Set Summary** (Continued)

| Mnemonic | | Description | Byte | Oscillator Period |
|---|---|---|---|---|
| **ARITHMETIC OPERATIONS** (Continued) | | | | |
| INC | DPTR | Increment Data Pointer | 1 | 24 |
| MUL | AB | Multiply A & B | 1 | 48 |
| DIV | AB | Divide A by B | 1 | 48 |
| DA | A | Decimal Adjust Accumulator | 1 | 12 |
| **LOGICAL OPERATIONS** | | | | |
| ANL | A,Rn | AND Register to Accumulator | 1 | 12 |
| ANL | A,direct | AND direct byte to Accumulator | 2 | 12 |
| ANL | A,@Ri | AND indirect RAM to Accumulator | 1 | 12 |
| ANL | A,#data | AND immediate data to Accumulator | 2 | 12 |
| ANL | direct,A | AND Accumulator to direct byte | 2 | 12 |
| ANL | direct,#data | AND immediate data to direct byte | 3 | 24 |
| ORL | A,Rn | OR register to Accumulator | 1 | 12 |
| ORL | A,direct | OR direct byte to Accumulator | 2 | 12 |
| ORL | A,@Ri | OR indirect RAM to Accumulator | 1 | 12 |
| ORL | A,#data | OR immediate data to Accumulator | 2 | 12 |
| ORL | direct,A | OR Accumulator to direct byte | 2 | 12 |
| ORL | direct,#data | OR immediate data to direct byte | 3 | 24 |
| XRL | A,Rn | Exclusive-OR register to Accumulator | 1 | 12 |
| XRL | A,direct | Exclusive-OR direct byte to Accumulator | 2 | 12 |
| XRL | A,@Ri | Exclusive-OR indirect RAM to Accumulator | 1 | 12 |
| XRL | A,#data | Exclusive-OR immediate data to Accumulator | 2 | 12 |
| XRL | direct,A | Exclusive-OR Accumulator to direct byte | 2 | 12 |
| XRL | direct,#data | Exclusive-OR immediate data to direct byte | 3 | 24 |
| CLR | A | Clear Accumulator | 1 | 12 |
| CPL | A | Complement Accumulator | 1 | 12 |

| Mnemonic | | Description | Byte | Oscillator Period |
|---|---|---|---|---|
| **LOGICAL OPERATIONS** (Continued) | | | | |
| RL | A | Rotate Accumulator Left | 1 | 12 |
| RLC | A | Rotate Accumulator Left through the Carry | 1 | 12 |
| RR | A | Rotate Accumulator Right | 1 | 12 |
| RRC | A | Rotate Accumulator Right through the Carry | 1 | 12 |
| SWAP | A | Swap nibbles within the Accumulator | 1 | 12 |
| **DATA TRANSFER** | | | | |
| MOV | A,Rn | Move register to Accumulator | 1 | 12 |
| MOV | A,direct | Move direct byte to Accumulator | 2 | 12 |
| MOV | A,@Ri | Move indirect RAM to Accumulator | 1 | 12 |
| MOV | A,#data | Move immediate data to Accumulator | 2 | 12 |
| MOV | Rn,A | Move Accumulator to register | 1 | 12 |
| MOV | Rn,direct | Move direct byte to register | 2 | 24 |
| MOV | Rn,#data | Move immediate data to register | 2 | 12 |
| MOV | direct,A | Move Accumulator to direct byte | 2 | 12 |
| MOV | direct,Rn | Move register to direct byte | 2 | 24 |
| MOV | direct,direct | Move direct byte to direct | 3 | 24 |
| MOV | direct,@Ri | Move indirect RAM to direct byte | 2 | 24 |
| MOV | direct,#data | Move immediate data to direct byte | 3 | 24 |
| MOV | @Ri,A | Move Accumulator to indirect RAM | 1 | 12 |

**Table 10. 8051 Instruction Set Summary** (Continued)

| Mnemonic | | Description | Byte | Oscillator Period |
|---|---|---|---|---|
| **DATA TRANSFER** (Continued) | | | | |
| MOV | @Ri,direct | Move direct byte to indirect RAM | 2 | 24 |
| MOV | @Ri,#data | Move immediate data to indirect RAM | 2 | 12 |
| MOV | DPTR,#data16 | Load Data Pointer with a 16-bit constant | 3 | 24 |
| MOVC | A,@A+DPTR | Move Code byte relative to DPTR to Acc | 1 | 24 |
| MOVC | A,@A+PC | Move Code byte relative to PC to Acc | 1 | 24 |
| MOVX | A,@Ri | Move External RAM (8-bit addr) to Acc | 1 | 24 |
| MOVX | A,@DPTR | Move External RAM (16-bit addr) to Acc | 1 | 24 |
| MOVX | @Ri,A | Move Acc to External RAM (8-bit addr) | 1 | 24 |
| MOVX | @DPTR,A | Move Acc to External RAM (16-bit addr) | 1 | 24 |
| PUSH | direct | Push direct byte onto stack | 2 | 24 |
| POP | direct | Pop direct byte from stack | 2 | 24 |
| XCH | A,Rn | Exchange register with Accumulator | 1 | 12 |
| XCH | A,direct | Exchange direct byte with Accumulator | 2 | 12 |
| XCH | A,@Ri | Exchange indirect RAM with Accumulator | 1 | 12 |
| XCHD | A,@Ri | Exchange low-order Digit indirect RAM with Acc | 1 | 12 |
| **BOOLEAN VARIABLE MANIPULATION** | | | | |
| CLR | C | Clear Carry | 1 | 12 |
| CLR | bit | Clear direct bit | 2 | 12 |
| SETB | C | Set Carry | 1 | 12 |
| SETB | bit | Set direct bit | 2 | 12 |
| CPL | C | Complement Carry | 1 | 12 |
| CPL | bit | Complement direct bit | 2 | 12 |
| ANL | C,bit | AND direct bit to CARRY | 2 | 24 |
| ANL | C,/bit | AND complement of direct bit to Carry | 2 | 24 |
| ORL | C,bit | OR direct bit to Carry | 2 | 24 |
| ORL | C,/bit | OR complement of direct bit to Carry | 2 | 24 |
| MOV | C,bit | Move direct bit to Carry | 2 | 12 |
| MOV | bit,C | Move Carry to direct bit | 2 | 24 |
| JC | rel | Jump if Carry is set | 2 | 24 |
| JNC | rel | Jump if Carry not set | 2 | 24 |
| JB | bit,rel | Jump if direct Bit is set | 3 | 24 |
| JNB | bit,rel | Jump if direct Bit is Not set | 3 | 24 |
| JBC | bit,rel | Jump if direct Bit is set & clear bit | 3 | 24 |
| **PROGRAM BRANCHING** | | | | |
| ACALL | addr11 | Absolute Subroutine Call | 2 | 24 |
| LCALL | addr16 | Long Subroutine Call | 3 | 24 |
| RET | | Return from Subroutine | 1 | 24 |
| RETI | | Return from interrupt | 1 | 24 |
| AJMP | addr11 | Absolute Jump | 2 | 24 |
| LJMP | addr16 | Long Jump | 3 | 24 |
| SJMP | rel | Short Jump (relative addr) | 2 | 24 |

**Table 10. 8051 Instruction Set Summary** (Continued)

| Mnemonic | | Description | Byte | Oscillator Period |
|---|---|---|---|---|
| **PROGRAM BRANCHING** (Continued) | | | | |
| JMP | @A+DPTR | Jump indirect relative to the DPTR | 1 | 24 |
| JZ | rel | Jump if Accumulator is Zero | 2 | 24 |
| JNZ | rel | Jump if Accumulator is Not Zero | 2 | 24 |
| CJNE | A,direct,rel | Compare direct byte to Acc and Jump if Not Equal | 3 | 24 |
| CJNE | A,#data,rel | Compare immediate to Acc and Jump if Not Equal | 3 | 24 |
| CJNE | Rn,#data,rel | Compare immediate to register and Jump if Not Equal | 3 | 24 |
| CJNE | @Ri,#data,rel | Compare immediate to indirect and Jump if Not Equal | 3 | 24 |
| DJNZ | Rn,rel | Decrement register and Jump if Not Zero | 2 | 24 |
| DJNZ | direct,rel | Decrement direct byte and Jump if Not Zero | 3 | 24 |
| NOP | | No Operation | 1 | 12 |

**Table 11. Instruction Opcodes in Hexadecimal Order**

| Hex Code | Number of Bytes | Mnemonic | Operands |
|---|---|---|---|
| 00 | 1 | NOP | |
| 01 | 2 | AJMP | code addr |
| 02 | 3 | LJMP | code addr |
| 03 | 1 | RR | A |
| 04 | 1 | INC | A |
| 05 | 2 | INC | data addr |
| 06 | 1 | INC | @R0 |
| 07 | 1 | INC | @R1 |
| 08 | 1 | INC | R0 |
| 09 | 1 | INC | R1 |
| 0A | 1 | INC | R2 |
| 0B | 1 | INC | R3 |
| 0C | 1 | INC | R4 |
| 0D | 1 | INC | R5 |
| 0E | 1 | INC | R6 |
| 0F | 1 | INC | R7 |
| 10 | 3 | JBC | bit addr, code addr |
| 11 | 2 | ACALL | code addr |
| 12 | 3 | LCALL | code addr |
| 13 | 1 | RRC | A |
| 14 | 1 | DEC | A |
| 15 | 2 | DEC | data addr |
| 16 | 1 | DEC | @R0 |
| 17 | 1 | DEC | @R1 |
| 18 | 1 | DEC | R0 |
| 19 | 1 | DEC | R1 |
| 1A | 1 | DEC | R2 |
| 1B | 1 | DEC | R3 |
| 1C | 1 | DEC | R4 |
| 1D | 1 | DEC | R5 |
| 1E | 1 | DEC | R6 |
| 1F | 1 | DEC | R7 |
| 20 | 3 | JB | bit addr, code addr |
| 21 | 2 | AJMP | code addr |
| 22 | 1 | RET | |
| 23 | 1 | RL | A |
| 24 | 2 | ADD | A, #data |
| 25 | 2 | ADD | A,data addr |
| 26 | 1 | ADD | A,@R0 |
| 27 | 1 | ADD | A,@R1 |
| 28 | 1 | ADD | A,R0 |
| 29 | 1 | ADD | A,R1 |
| 2A | 1 | ADD | A,R2 |
| 2B | 1 | ADD | A,R3 |
| 2C | 1 | ADD | A,R4 |
| 2D | 1 | ADD | A,R5 |
| 2E | 1 | ADD | A,R6 |
| 2F | 1 | ADD | A,R7 |
| 30 | 3 | JNB | bit addr, code addr |
| 31 | 2 | ACALL | code addr |
| 32 | 1 | RETI | |
| 33 | 1 | RLC | A |
| 34 | 2 | ADDC | A, #data |
| 35 | 2 | ADDC | A,data addr |
| 36 | 1 | ADDC | A,@R0 |
| 37 | 1 | ADDC | A,@R1 |
| 38 | 1 | ADDC | A,R0 |
| 39 | 1 | ADDC | A,R1 |
| 3A | 1 | ADDC | A,R2 |
| 3B | 1 | ADDC | A,R3 |
| 3C | 1 | ADDC | A,R4 |
| 3D | 1 | ADDC | A,R5 |
| 3E | 1 | ADDC | A,R6 |
| 3F | 1 | ADDC | A,R7 |
| 40 | 2 | JC | code addr |
| 41 | 2 | AJMP | code addr |
| 42 | 2 | ORL | data addr,A |
| 43 | 3 | ORL | data addr, #data |
| 44 | 2 | ORL | A, #data |
| 45 | 2 | ORL | A,data addr |
| 46 | 1 | ORL | A,@R0 |
| 47 | 1 | ORL | A,@R1 |
| 48 | 1 | ORL | A,R0 |
| 49 | 1 | ORL | A,R1 |
| 4A | 1 | ORL | A,R2 |
| 4B | 1 | ORL | A,R3 |
| 4C | 1 | ORL | A,R4 |
| 4D | 1 | ORL | A,R5 |
| 4E | 1 | ORL | A,R6 |
| 4F | 1 | ORL | A,R7 |
| 50 | 2 | JNC | code addr |
| 51 | 2 | ACALL | code addr |
| 52 | 2 | ANL | data addr,A |
| 53 | 3 | ANL | data addr, #data |
| 54 | 2 | ANL | A, #data |
| 55 | 2 | ANL | A,data addr |
| 56 | 1 | ANL | A,@R0 |
| 57 | 1 | ANL | A,@R1 |
| 58 | 1 | ANL | A,R0 |
| 59 | 1 | ANL | A,R1 |
| 5A | 1 | ANL | A,R2 |
| 5B | 1 | ANL | A,R3 |
| 5C | 1 | ANL | A,R4 |
| 5D | 1 | ANL | A,R5 |
| 5E | 1 | ANL | A,R6 |
| 5F | 1 | ANL | A,R7 |
| 60 | 2 | JZ | code addr |
| 61 | 2 | AJMP | code addr |
| 62 | 2 | XRL | data addr,A |
| 63 | 3 | XRL | data addr, #data |
| 64 | 2 | XRL | A, #data |
| 65 | 2 | XRL | A,data addr |

**Table 11. Instruction Opcodes in Hexadecimal Order** (Continued)

| Hex Code | Number of Bytes | Mnemonic | Operands |
|---|---|---|---|
| 66 | 1 | XRL | A,@R0 |
| 67 | 1 | XRL | A,@R1 |
| 68 | 1 | XRL | A,R0 |
| 69 | 1 | XRL | A,R1 |
| 6A | 1 | XRL | A,R2 |
| 6B | 1 | XRL | A,R3 |
| 6C | 1 | XRL | A,R4 |
| 6D | 1 | XRL | A,R5 |
| 6E | 1 | XRL | A,R6 |
| 6F | 1 | XRL | A,R7 |
| 70 | 2 | JNZ | code addr |
| 71 | 2 | ACALL | code addr |
| 72 | 2 | ORL | C,bit addr |
| 73 | 1 | JMP | @A+DPTR |
| 74 | 2 | MOV | A,#data |
| 75 | 3 | MOV | data addr,#data |
| 76 | 2 | MOV | @R0,#data |
| 77 | 2 | MOV | @R1,#data |
| 78 | 2 | MOV | R0,#data |
| 79 | 2 | MOV | R1,#data |
| 7A | 2 | MOV | R2,#data |
| 7B | 2 | MOV | R3,#data |
| 7C | 2 | MOV | R4,#data |
| 7D | 2 | MOV | R5,#data |
| 7E | 2 | MOV | R6,#data |
| 7F | 2 | MOV | R7,#data |
| 80 | 2 | SJMP | code addr |
| 81 | 2 | AJMP | code addr |
| 82 | 2 | ANL | C,bit addr |
| 83 | 1 | MOVC | A,@A+PC |
| 84 | 1 | DIV | AB |
| 85 | 3 | MOV | data addr, data addr |
| 86 | 2 | MOV | data addr,@R0 |
| 87 | 2 | MOV | data addr,@R1 |
| 88 | 2 | MOV | data addr,R0 |
| 89 | 2 | MOV | data addr,R1 |
| 8A | 2 | MOV | data addr,R2 |
| 8B | 2 | MOV | data addr,R3 |
| 8C | 2 | MOV | data addr,R4 |
| 8D | 2 | MOV | data addr,R5 |
| 8E | 2 | MOV | data addr,R6 |
| 8F | 2 | MOV | data addr,R7 |
| 90 | 3 | MOV | DPTR,#data |
| 91 | 2 | ACALL | code addr |
| 92 | 2 | MOV | bit addr,C |
| 93 | 1 | MOVC | A,@A+DPTR |
| 94 | 2 | SUBB | A,#data |
| 95 | 2 | SUBB | A,data addr |
| 96 | 1 | SUBB | A,@R0 |
| 97 | 1 | SUBB | A,@R1 |
| 98 | 1 | SUBB | A,R0 |
| 99 | 1 | SUBB | A,R1 |
| 9A | 1 | SUBB | A,R2 |
| 9B | 1 | SUBB | A,R3 |
| 9C | 1 | SUBB | A,R4 |
| 9D | 1 | SUBB | A,R5 |
| 9E | 1 | SUBB | A,R6 |
| 9F | 1 | SUBB | A,R7 |
| A0 | 2 | ORL | C,/bit addr |
| A1 | 2 | AJMP | code addr |
| A2 | 2 | MOV | C,bit addr |
| A3 | 1 | INC | DPTR |
| A4 | 1 | MUL | AB |
| A5 | | reserved | |
| A6 | 2 | MOV | @R0,data addr |
| A7 | 2 | MOV | @R1,data addr |
| A8 | 2 | MOV | R0,data addr |
| A9 | 2 | MOV | R1,data addr |
| AA | 2 | MOV | R2,data addr |
| AB | 2 | MOV | R3,data addr |
| AC | 2 | MOV | R4,data addr |
| AD | 2 | MOV | R5,data addr |
| AE | 2 | MOV | R6,data addr |
| AF | 2 | MOV | R7,data addr |
| B0 | 2 | ANL | C,/bit addr |
| B1 | 2 | ACALL | code addr |
| B2 | 2 | CPL | bit addr |
| B3 | 1 | CPL | C |
| B4 | 3 | CJNE | A,#data,code addr |
| B5 | 3 | CJNE | A,data addr,code addr |
| B6 | 3 | CJNE | @R0,#data,code addr |
| B7 | 3 | CJNE | @R1,#data,code addr |
| B8 | 3 | CJNE | R0,#data,code addr |
| B9 | 3 | CJNE | R1,#data,code addr |
| BA | 3 | CJNE | R2,#data,code addr |
| BB | 3 | CJNE | R3,#data,code addr |
| BC | 3 | CJNE | R4,#data,code addr |
| BD | 3 | CJNE | R5,#data,code addr |
| BE | 3 | CJNE | R6,#data,code addr |
| BF | 3 | CJNE | R7,#data,code addr |
| C0 | 2 | PUSH | data addr |
| C1 | 2 | AJMP | code addr |
| C2 | 2 | CLR | bit addr |
| C3 | 1 | CLR | C |
| C4 | 1 | SWAP | A |
| C5 | 2 | XCH | A,data addr |
| C6 | 1 | XCH | A,@R0 |
| C7 | 1 | XCH | A,@R1 |
| C8 | 1 | XCH | A,R0 |
| C9 | 1 | XCH | A,R1 |
| CA | 1 | XCH | A,R2 |
| CB | 1 | XCH | A,R3 |

**Table 11. Instruction Opcodes in Hexadecimal Order** (Continued)

| Hex Code | Number of Bytes | Mnemonic | Operands |
|---|---|---|---|
| CC | 1 | XCH | A,R4 |
| CD | 1 | XCH | A,R5 |
| CE | 1 | XCH | A,R6 |
| CF | 1 | XCH | A,R7 |
| D0 | 2 | POP | data addr |
| D1 | 2 | ACALL | code addr |
| D2 | 2 | SETB | bit addr |
| D3 | 1 | SETB | C |
| D4 | 1 | DA | A |
| D5 | 3 | DJNZ | data addr,code addr |
| D6 | 1 | XCHD | A,@R0 |
| D7 | 1 | XCHD | A,@R1 |
| D8 | 2 | DJNZ | R0,code addr |
| D9 | 2 | DJNZ | R1,code addr |
| DA | 2 | DJNZ | R2,code addr |
| DB | 2 | DJNZ | R3,code addr |
| DC | 2 | DJNZ | R4,code addr |
| DD | 2 | DJNZ | R5,code addr |
| DE | 2 | DJNZ | R6,code addr |
| DF | 2 | DJNZ | R7,code addr |
| E0 | 1 | MOVX | A,@DPTR |
| E1 | 2 | AJMP | code addr |
| E2 | 1 | MOVX | A,@R0 |
| E3 | 1 | MOVX | A,@R1 |
| E4 | 1 | CLR | A |
| E5 | 2 | MOV | A,data addr |
| E6 | 1 | MOV | A,@R0 |
| E7 | 1 | MOV | A,@R1 |
| E8 | 1 | MOV | A,R0 |
| E9 | 1 | MOV | A,R1 |
| EA | 1 | MOV | A,R2 |
| EB | 1 | MOV | A,R3 |
| EC | 1 | MOV | A,R4 |
| ED | 1 | MOV | A,R5 |
| EE | 1 | MOV | A,R6 |
| EF | 1 | MOV | A,R7 |
| F0 | 1 | MOVX | @DPTR,A |
| F1 | 2 | ACALL | code addr |
| F2 | 1 | MOVX | @R0,A |
| F3 | 1 | MOVX | @R1,A |
| F4 | 1 | CPL | A |
| F5 | 2 | MOV | data addr,A |
| F6 | 1 | MOV | @R0,A |
| F7 | 1 | MOV | @R1,A |
| F8 | 1 | MOV | R0,A |
| F9 | 1 | MOV | R1,A |
| FA | 1 | MOV | R2,A |
| FB | 1 | MOV | R3,A |
| FC | 1 | MOV | R4,A |
| FD | 1 | MOV | R5,A |
| FE | 1 | MOV | R6,A |
| FF | 1 | MOV | R7,A |

# INSTRUCTION DEFINITIONS

## ACALL addr11

**Function:** Absolute Call

**Description:** ACALL unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction, then pushes the 16-bit result onto the stack (low-order byte first) and increments the Stack Pointer twice. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, opcode bits 7-5, and the second byte of the instruction. The subroutine called must therefore start within the same 2K block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

**Example:** Initially SP equals 07H. The label "SUBRTN" is at program memory location 0345 H. After executing the instruction,

ACALL SUBRTN

at location 0123H, SP will contain 09H, internal RAM locations 08H and 09H will contain 25H and 01H, respectively, and the PC will contain 0345H.

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| a10 a9 a8 1 | 0 0 0 1 | a7 a6 a5 a4 | a3 a2 a1 a0 |
|---|---|---|---|

**Operation:** ACALL
$(PC) \leftarrow (PC) + 2$
$(SP) \leftarrow (SP) + 1$
$((SP)) \leftarrow (PC_{7\text{-}0})$
$(SP) \leftarrow (SP) + 1$
$((SP)) \leftarrow (PC_{15\text{-}8})$
$(PC_{10\text{-}0}) \leftarrow$ page address

## ADD A,<src-byte>

**Function:** Add

**Description:** ADD adds the byte variable indicated to the Accumulator, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occured.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

**Example:** The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B). The instruction,

ADD A,R0

will leave 6DH (01101101B) in the Accumulator with the AC flag cleared and both the carry flag and OV set to 1.

### ADD A,Rn

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 1 0 | 1 r r r |
|---|---|

**Operation:** ADD
(A) ← (A) + (Rn)

### ADD A,direct

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 0 1 0 | 0 1 0 1 | direct address |
|---|---|---|

**Operation:** ADD
(A) ← (A) + (direct)

**ADD A,@Ri**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 1 0 | 0 1 1 i |
|---|---|

**Operation:** ADD
(A) ← (A) + ((Ri))

**ADD A,#data**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 0 1 0 | 0 1 0 0 | immediate data |
|---|---|---|

**Operation:** ADD
(A) ← (A) + #data

**ADDC A,<src-byte>**

**Function:** Add with Carry

**Description:** ADDC simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occured.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

**Example:** The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) with the carry flag set. The instruction,

ADDC A,R0

will leave 6EH (01101110B) in the Accumulator with AC cleared and both the Carry flag and OV set to 1.

**ADDC A,Rn**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 1 1 | 1 r r r |
|---|---|

**Operation:** ADDC
(A) ← (A) + (C) + ($R_n$)

**ADDC A,direct**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 0 1 1 | 0 1 0 1 | direct address |
|---|---|---|

**Operation:** ADDC
(A) ← (A) + (C) + (direct)

**ADDC A,@Ri**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 1 1 | 0 1 1 i |
|---|---|

**Operation:** ADDC
(A) ← (A) + (C) + (($R_i$))

**ADDC A,#data**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 0 1 1 | 0 1 0 0 | immediate data |
|---|---|---|

**Operation:** ADDC
(A) ← (A) + (C) + #data

## AJMP addr11

**Function:** Absolute Jump

**Description:** AJMP transfers program execution to the indicated address, which is formed at run-time by concatenating the high-order five bits of the PC (*after* incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must therefore be within the same 2K block of program memory as the first byte of the instruction following AJMP.

**Example:** The label "JMPADR" is at program memory location 0123H. The instruction,

```
AJMP   JMPADR
```

is at location 0345H and will load the PC with 0123H.

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| a10 a9 a8 0 | 0 0 0 1 | a7 a6 a5 a4 | a3 a2 a1 a0 |
|---|---|---|---|

**Operation:** AJMP
(PC) ← (PC) + 2
($PC_{10\text{-}0}$) ← page address

## ANL <dest-byte>,<src-byte>

**Function:** Logical-AND for byte variables

**Description:** ANL performs the bitwise logical-AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

*Note:* When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

**Example:** If the Accumulator holds 0C3H (11000011B) and register 0 holds 55H (01010101B) then the instruction,

```
ANL   A,R0
```

will leave 41H (01000001B) in the Accumulator.

When the destination is a directly addressed byte, this instruction will clear combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be a constant contained in the instruction or a value computed in the Accumulator at run-time. The instruction,

```
ANL   P1,#01110011B
```

will clear bits 7, 3, and 2 of output port 1.

**ANL A,Rn**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 1 0 1 | 1 r r r |
|---|---|

**Operation:** ANL
(A) ← (A) ∧ (Rn)

**ANL A,direct**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 0 1 | 0 1 0 1 | direct address |
|---|---|---|

**Operation:** ANL

(A) ← (A) ∧ (direct)

**ANL A,@Ri**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 1 0 1 | 0 1 1 i |
|---|---|

**Operation:** ANL

(A) ← (A) ∧ ((Ri))

**ANL A,#data**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 0 1 | 0 1 0 0 | immediate data |
|---|---|---|

**Operation:** ANL
(A) ← (A) ∧ #data

**ANL direct,A**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 0 1 | 0 0 1 0 | direct address |
|---|---|---|

**Operation:** ANL
(direct) ← (direct) ∧ (A)

**ANL direct,#data**

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 0 1 0 1 | 0 0 1 1 | direct address | immediate data |
|---|---|---|---|

**Operation:** ANL
(direct) ← (direct) ∧ #data

**ANL C,<src-bit>**

**Function:** Logical-AND for bit variables

**Description:** If the Boolean value of the source bit is a logical 0 then clear the carry flag; otherwise leave the carry flag in its current state. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, *but the source bit itself is not affected.* No other flags are affected.

Only direct addressing is allowed for the source operand.

**Example:** Set the carry flag if, and only if, P1.0 = 1, ACC. 7 = 1, and OV = 0:

```
MOV  C,P1.0     ;LOAD CARRY WITH INPUT PIN STATE
ANL  C,ACC.7    ;AND CARRY WITH ACCUM. BIT 7
ANL  C,/OV      ;AND WITH INVERSE OF OVERFLOW FLAG
```

**ANL C,bit**

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 1 0 0 0 | 0 0 1 0 | bit address |
|---|---|---|

**Operation:** ANL
(C) ← (C) ∧ (bit)

**ANL C,/bit**

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 1 0 1 1 | 0 0 0 0 | bit address |
|---|---|---|

**Operation:** ANL
(C) ← (C) ∧ ¬ (bit)

## CJNE <dest-byte>,<src-byte>, rel

**Function:** Compare and Jump if Not Equal.

**Description:** CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The branch destination is computed by adding the signed relative-displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction. The carry flag is set if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations: the Accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

**Example:** The Accumulator contains 34H. Register 7 contains 56H. The first instruction in the sequence,

```
                CJNE    R7,#60H, NOT_EQ
;               ...     .....           ; R7 = 60H.
NOT_EQ:         JC      REQ_LOW         ; IF R7 < 60H.
;               ...     .....           ; R7 > 60H.
```

sets the carry flag and branches to the instruction at label NOT_EQ. By testing the carry flag, this instruction determines whether R7 is greater or less than 60H.

If the data being presented to Port 1 is also 34H, then the instruction,

```
WAIT:  CJNE  A,P1,WAIT
```

clears the carry flag and continues with the next instruction in sequence, since the Accumulator does equal the data read from P1. (If some other value was being input on P1, the program will loop at this point until the P1 data changes to 34H.)

## CJNE A,direct,rel

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 1 0 1 1 | 0 1 0 1 | direct address | rel. address |
|---|---|---|---|

**Operation:**

```
(PC) ← (PC) + 3
IF (A) <> (direct)
THEN
        (PC) ← (PC) + relative offset

IF (A) < (direct)
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

**CJNE A,#data,rel**

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 1 0 1 1 | 0 1 0 0 | immediate data | rel. address |
|---|---|---|---|

**Operation:**

```
(PC) ← (PC) + 3
IF (A) <> data
THEN
        (PC) ← (PC) + relative offset

IF (A) < data
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

**CJNE Rn,#data,rel**

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 1 0 1 1 | 1 r r r | immediate data | rel. address |
|---|---|---|---|

**Operation:**

```
(PC) ← (PC) + 3
IF (Rn) <> data
THEN
        (PC) ← (PC) + relative offset

IF (Rn) < data
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

**CJNE @Ri,#data,rel**

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 1 0 1 1 | 0 1 1 i | immediate data | rel. address |
|---|---|---|---|

**Operation:**

```
(PC) ← (PC) + 3
IF ((Ri)) <> data
THEN
        (PC) ← (PC) + relative offset

IF ((Ri)) < data
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

### CLR A

**Function:** Clear Accumulator

**Description:** The Accumulator is cleared (all bits set on zero). No flags are affected.

**Example:** The Accumulator contains 5CH (01011100B). The instruction,

CLR A

will leave the Accumulator set to 00H (00000000B).

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 1 0 | 0 1 0 0 |
|---|---|

**Operation:**
CLR
(A) ← 0

### CLR bit

**Function:** Clear bit

**Description:** The indicated bit is cleared (reset to zero). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

**Example:** Port 1 has previously been written with 5DH (01011101B). The instruction,

CLR P1.2

will leave the port set to 59H (01011001B).

### CLR C

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 0 0 | 0 0 1 1 |
|---|---|

**Operation:**
CLR
(C) ← 0

### CLR bit

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 1 1 0 0 | 0 0 1 0 | bit address |
|---|---|---|

**Operation:**
CLR
(bit) ← 0

## CPL A

**Function:** Complement Accumulator

**Description:** Each bit of the Accumulator is logically complemented (one's complement). Bits which previously contained a one are changed to a zero and vice-versa. No flags are affected.

**Example:** The Accumulator contains 5CH (01011100B). The instruction,

CPL A

will leave the Accumulator set to 0A3H (10100011B).

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 1 1 | 0 1 0 0 |
|---|---|

**Operation:**
CPL
(A) ← ¬ (A)

## CPL bit

**Function:** Complement bit

**Description:** The bit variable specified is complemented. A bit which had been a one is changed to zero and vice-versa. No other flags are affected. CLR can operate on the carry or any directly addressable bit.

*Note:* When this instruction is used to modify an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

**Example:** Port 1 has previously been written with 5BH (01011101B). The instruction sequence,

CPL P1.1

CPL P1.2

will leave the port set to 5BH (01011011B).

## CPL C

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 0 1 1 | 0 0 1 1 |
|---|---|

**Operation:**
CPL
(C) ← ¬ (C)

**CPL bit**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 1 0 1 1 | 0 0 1 0 | bit address |
|---|---|---|

**Operation:** CPL
(bit) ← ¬ (bit)

**DA A**

**Function:** Decimal-adjust Accumulator for Addition

**Description:** DA A adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variables (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If Accumulator bits 3-0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag is one, six is added to the Accumulator producing the proper BCD digit in the low-order nibble. This internal addition would set the carry flag if a carry-out of the low-order four-bit field propagated through all high-order bits, but it would not clear the carry flag otherwise.

If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxxx-111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this would set the carry flag if there was a carry-out of the high-order bits, but wouldn't clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00H, 06H, 60H, or 66H to the Accumulator, depending on initial Accumulator and PSW conditions.

*Note:* DA A *cannot* simply convert a hexadecimal number in the Accumulator to BCD notation, nor does DA A apply to decimal subtraction.

**Example:** The Accumulator holds the value 56H (01010110B) representing the packed BCD digits of the decimal number 56. Register 3 contains the value 67H (01100111B) representing the packed BCD digits of the decimal number 67. The carry flag is set. The instruction sequence.

```
ADDC  A,R3
DA    A
```

will first perform a standard twos-complement binary addition, resulting in the value 0BEH (10111110) in the Accumulator. The carry and auxiliary carry flags will be cleared.

The Decimal Adjust instruction will then alter the Accumulator to the value 24H (00100100B), indicating the packed BCD digits of the decimal number 24, the low-order two digits of the decimal sum of 56, 67, and the carry-in. The carry flag will be set by the Decimal Adjust instruction, indicating that a decimal overflow occurred. The true sum 56, 67, and 1 is 124.

BCD variables can be incremented or decremented by adding 01H or 99H. If the Accumulator initially holds 30H (representing the digits of 30 decimal), then the instruction sequence,

```
ADD   A,#99H

DA    A
```

will leave the carry set and 29H in the Accumulator, since 30 + 99 = 129. The low-order byte of the sum can be interpreted to mean 30 − 1 = 29.

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 0 1 | 0 1 0 0 |
|---|---|

**Operation:** DA

-contents of Accumulator are BCD

IF $[[(A_{3-0}) > 9] \vee [(AC) = 1]]$

THEN $(A_{3-0}) \leftarrow (A_{3-0}) + 6$

AND

IF $[[(A_{7-4}) > 9] \vee [(C) = 1]]$

THEN $(A_{7-4}) \leftarrow (A_{7-4}) + 6$

## DEC byte

**Function:** Decrement

**Description:** The variable indicated is decremented by 1. An original value of 00H will underflow to 0FFH. No flags are affected. Four operand addressing modes are allowed: accumulator, register, direct, or register-indirect.

*Note:* When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

**Example:** Register 0 contains 7FH (01111111B). Internal RAM locations 7EH and 7FH contain 00H and 40H, respectively. The instruction sequence,

```
DEC  @R0
DEC  R0
DEC  @R0
```

will leave register 0 set to 7EH and internal RAM locations 7EH and 7FH set to 0FFH and 3FH.

## DEC A

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 0 1 | 0 1 0 0 |
|---|---|

**Operation:** DEC
(A) ← (A) − 1

## DEC Rn

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 0 1 | 1 r r r |
|---|---|

**Operation:** DEC
(Rn) ← (Rn) − 1

**DEC direct**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 0 0 1 | 0 1 0 1 | direct address |
|---|---|---|

**Operation:** DEC
(direct) ← (direct) − 1

**DEC @Ri**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 0 1 | 0 1 1 i |
|---|---|

**Operation:** DEC
((Ri)) ← ((Ri)) − 1

**DIV AB**

**Function:** Divide

**Description:** DIV AB divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B. The Accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and OV flags will be cleared.

*Exception:* if B had originally contained 00H, the values returned in the Accumulator and B-register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

**Example:** The Accumulator contains 251 (0FBH or 11111011B) and B contains 18 (12H or 00010010B). The instruction,

DIV AB

will leave 13 in the Accumulator (0DH or 00001101B) and the value 17 (11H or 00010001B) in B, since 251 = (13 X 18) + 17. Carry and OV will both be cleared.

**Bytes:** 1

**Cycles:** 4

**Encoding:**

| 1 0 0 0 | 0 1 0 0 |
|---|---|

**Operation:** DIV
$(A)_{15\text{-}8}$
$(B)_{7\text{-}0}$ ← (A)/(B)

**DJNZ <byte>,<rel-addr>**

**Function:** Decrement and Jump if Not Zero

**Description:** DJNZ decrements the location indicated by 1, and branches to the address indicated by the second operand if the resulting value is not zero. An original value of 00H will underflow to 0FFH. No flags are affected. The branch destination would be computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

*Note:* When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

**Example:** Internal RAM locations 40H, 50H, and 60H contain the values 01H, 70H, and 15H, respectively. The instruction sequence,

```
DJNZ  40H,LABEL_1
DJNZ  50H,LABEL_2
DJNZ  60H,LABEL_3
```

will cause a jump to the instruction at label LABEL_2 with the values 00H, 6FH, and 15H in the three RAM locations. The first jump was *not* taken because the result was zero.

This instruction provides a simple way of executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. The instruction sequence,

```
        MOV     R2,#8
TOGGLE: CPL     P1.7
        DJNZ    R2,TOGGLE
```

will toggle P1.7 eight times, causing four output pulses to appear at bit 7 of output Port 1. Each pulse will last three machine cycles; two for DJNZ and one to alter the pin.

**DJNZ Rn,rel**

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 1 1 0 1 | 1 r r r | rel. address |
|---|---|---|

**Operation:**

```
DJNZ
(PC) ← (PC) + 2
(Rn) ← (Rn) − 1
IF  (Rn) > 0 or (Rn) < 0
    THEN
          (PC) ← (PC) + rel
```

**DJNZ direct,rel**

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 1 1 0 1 | 0 1 0 1 | direct address | rel. address |
|---|---|---|---|

**Operation:**

```
DJNZ
(PC) ← (PC) + 2
(direct) ← (direct) − 1
IF (direct) > 0 or (direct) < 0
   THEN
         (PC) ← (PC) + rel
```

**INC <byte>**

**Function:** Increment

**Description:** INC increments the indicated variable by 1. An original value of 0FFH will overflow to 00H. No flags are affected. Three addressing modes are allowed: register, direct, or register-indirect.

*Note:* When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

**Example:** Register 0 contains 7EH (011111110B). Internal RAM locations 7EH and 7FH contain 0FFH and 40H, respectively. The instruction sequence,

```
INC  @R0
INC  R0
INC  @R0
```

will leave register 0 set to 7FH and internal RAM locations 7EH and 7FH holding (respectively) 00H and 41H.

**INC A**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 0 0 | 0 1 0 0 |
|---|---|

**Operation:**

```
INC
(A) ← (A) + 1
```

**INC Rn**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 0 0 | 1 r r r |
|---|---|

**Operation:** INC
(Rn) ← (Rn) + 1

**INC direct**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 0 0 0 | 0 1 0 1 | direct address |
|---|---|---|

**Operation:** INC
(direct) ← (direct) + 1

**INC @Ri**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 0 0 | 0 1 1 i |
|---|---|

**Operation:** INC
((Ri)) ← ((Ri)) + 1

**INC DPTR**

**Function:** Increment Data Pointer

**Description:** Increment the 16-bit data pointer by 1. A 16-bit increment (modulo $2^{16}$) is performed; an overflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H will increment the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be incremented.

**Example:** Registers DPH and DPL contain 12H and 0FEH, respectively. The instruction sequence,

```
INC  DPTR
INC  DPTR
INC  DPTR
```

will change DPH and DPL to 13H and 01H.

**Bytes:** 1

**Cycles:** 2

**Encoding:**

| 1 0 1 0 | 0 0 1 1 |
|---|---|

**Operation:** INC
(DPTR) ← (DPTR) + 1

## JB bit,rel

**Function:** Jump if Bit set

**Description:** If the indicated bit is a one, jump to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

**Example:** The data present at input port 1 is 11001010B. The Accumulator holds 56 (01010110B). The instruction sequence,

```
JB   P1.2,LABEL1

JB   ACC.2,LABEL2
```

will cause program execution to branch to the instruction at label LABEL2.

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 0 0 1 0 | 0 0 0 0 | bit address | rel. address |
|---|---|---|---|

**Operation:**

```
JB
(PC) ← (PC) + 3
IF  (bit) = 1
    THEN
            (PC) ← (PC) + rel
```

## JBC bit,rel

**Function:** Jump if Bit is set and Clear bit

**Description:** If the indicated bit is one, branch to the address indicated; otherwise proceed with the next instruction. *The bit will not be cleared if it is already a zero.* The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected.

*Note:* When this instruction is used to test an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

**Example:** The Accumulator holds 56H (01010110B). The instruction sequence,

```
JBC  ACC.3,LABEL1
JBC  ACC.2,LABEL2
```

will cause program execution to continue at the instruction identified by the label LABEL2, with the Accumulator modified to 52H (01010010B).

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 0 0 0 1 | 0 0 0 0 | bit address | rel. address |
|---|---|---|---|

**Operation:**

```
JBC
(PC) ← (PC) + 3
IF  (bit) = 1
    THEN
            (bit) ← 0
            (PC) ← (PC) + rel
```

## JC rel

**Function:** Jump if Carry is set

**Description:** If the carry flag is set, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. No flags are affected.

**Example:** The carry flag is cleared. The instruction sequence,

```
JC    LABEL1
CPL   C
JC    LABEL 2
```

will set the carry and cause program execution to continue at the instruction identified by the label LABEL2.

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 0 1 0 0 | 0 0 0 0 | rel. address |
|---|---|---|

**Operation:**

```
JC
(PC) ← (PC) + 2
IF  (C) = 1
    THEN
            (PC) ← (PC) + rel
```

## JMP @A+DPTR

**Function:** Jump indirect

**Description:** Add the eight-bit unsigned contents of the Accumulator with the sixteen-bit data pointer, and load the resulting sum to the program counter. This will be the address for subsequent instruction fetches. Sixteen-bit addition is performed (modulo $2^{16}$): a carry-out from the low-order eight bits propagates through the higher-order bits. Neither the Accumulator nor the Data Pointer is altered. No flags are affected.

**Example:** An even number from 0 to 6 is in the Accumulator. The following sequence of instructions will branch to one of four AJMP instructions in a jump table starting at JMP_TBL:

```
        MOV   DPTR,#JMP_TBL
        JMP   @A+DPTR
JMP_TBL: AJMP  LABEL0
        AJMP  LABEL1
        AJMP  LABEL2
        AJMP  LABEL3
```

If the Accumulator equals 04H when starting this sequence, execution will jump to label LABEL2. Remember that AJMP is a two-byte instruction, so the jump instructions start at every other address.

**Bytes:** 1

**Cycles:** 2

**Encoding:**

| 0 1 1 1 | 0 0 1 1 |
|---|---|

**Operation:** JMP
(PC) ← (A) + (DPTR)

## JNB bit,rel

**Function:** Jump if Bit Not set

**Description:** If the indicated bit is a zero, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

**Example:** The data present at input port 1 is 11001010B. The Accumulator holds 56H (01010110B). The instruction sequence,

```
JNB  P1.3,LABEL1
JNB  ACC.3,LABEL2
```

will cause program execution to continue at the instruction at label LABEL2.

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 0 0 1 1 | 0 0 0 0 | bit address | rel. address |
|---|---|---|---|

**Operation:**

```
JNB
(PC) ← (PC) + 3
IF  (bit) = 0
    THEN (PC) ← (PC) + rel.
```

## JNC rel

**Function:** Jump if Carry not set

**Description:** If the carry flag is a zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The carry flag is not modified.

**Example:** The carry flag is set. The instruction sequence,

```
JNC  LABEL1
CPL  C
JNC  LABEL2
```

will clear the carry and cause program execution to continue at the instruction identified by the label LABEL2.

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 0 1 0 1 | 0 0 0 0 | rel. address |
|---|---|---|

**Operation:**

```
JNC
(PC) ← (PC) + 2
IF  (C) = 0
    THEN  (PC) ← (PC) + rel
```

## JNZ rel

**Function:** Jump if Accumulator Not Zero

**Description:** If any bit of the Accumulator is a one, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

**Example:** The Accumulator originally holds 00H. The instruction sequence,

```
JNZ  LABEL1
INC  A
JNZ  LABEL2
```

will set the Accumulator to 01H and continue at label LABEL2.

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 0 1 1 1 | 0 0 0 0 | rel. address |
|---|---|---|

**Operation:**

```
JNZ
(PC) ← (PC) + 2
IF   (A) ≠ 0
     THEN  (PC) ← (PC) + rel
```

## JZ rel

**Function:** Jump if Accumulator Zero

**Description:** If all bits of the Accumulator are zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

**Example:** The Accumulator originally contains 01H. The instruction sequence,

```
JZ   LABEL1
DEC  A
JZ   LABEL2
```

will change the Accumulator to 00H and cause program execution to continue at the instruction identified by the label LABEL2.

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 0 1 1 0 | 0 0 0 0 | rel. address |
|---|---|---|

**Operation:**

```
JZ
(PC) ← (PC) + 2
IF   (A) = 0
     THEN  (PC) ← (PC) + rel
```

## LCALL addr16

**Function:** Long call

**Description:** LCALL calls a subroutine located at the indicated address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first), incrementing the Stack Pointer by two. The high-order and low-order bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 64K-byte program memory address space. No flags are affected.

**Example:** Initially the Stack Pointer equals 07H. The label "SUBRTN" is assigned to program memory location 1234H. After executing the instruction,

```
LCALL  SUBRTN
```

at location 0123H, the Stack Pointer will contain 09H, internal RAM locations 08H and 09H will contain 26H and 01H, and the PC will contain 1234H.

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 0 0 0 1 \| 0 0 1 0 | addr15-addr8 | addr7-addr0 |
|---|---|---|

**Operation:**
LCALL
(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← ($PC_{7-0}$)
(SP) ← (SP) + 1
((SP)) ← ($PC_{15-8}$)
(PC) ← $addr_{15-0}$

## LJMP addr16

**Function:** Long Jump

**Description:** LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64K program memory address space. No flags are affected.

**Example:** The label "JMPADR" is assigned to the instruction at program memory location 1234H. The instruction,

```
LJMP  JMPADR
```

at location 0123H will load the program counter with 1234H.

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 0 0 0 0 \| 0 0 1 0 | addr15-addr8 | addr7-addr0 |
|---|---|---|

**Operation:**
LJMP
(PC) ← $addr_{15-0}$

## MOV <dest-byte>,<src-byte>

**Function:** Move byte variable

**Description:** The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

This is by far the most flexible operation. Fifteen combinations of source and destination addressing modes are allowed.

**Example:** Internal RAM location 30H holds 40H. The value of RAM location 40H is 10H. The data present at input port 1 is 11001010B (0CAH).

```
MOV   R0,#30H   ;R0 <= 30H
MOV   A,@R0     ;A <= 40H
MOV   R1,A      ;R1 <= 40H
MOV   B,@R1     ;B <= 10H
MOV   @R1,P1    ;RAM (40H) <= 0CAH
MOV   P2,P1     ;P2 #0CAH
```

leaves the value 30H in register 0, 40H in both the Accumulator and register 1, 10H in register B, and 0CAH (11001010B) both in RAM location 40H and output on port 2.

### MOV A,Rn

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 1 0 | 1 r r r |
|---|---|

**Operation:** MOV
(A) ← (Rn)

### *MOV A,direct

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 1 1 1 0 | 0 1 0 1 | direct address |
|---|---|---|

**Operation:** MOV
(A) ← (direct)

**MOV A,ACC is not a valid instruction.**

**MOV A,@Ri**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 1 0 | 0 1 1 i |
|---|---|

**Operation:**
MOV
(A) ← ((Ri))

**MOV A,#data**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 1 1 | 0 1 0 0 | immediate data |
|---|---|---|

**Operation:**
MOV
(A) ← #data

**MOV Rn,A**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 1 1 | 1 r r r |
|---|---|

**Operation:**
MOV
(Rn) ← (A)

**MOV Rn,direct**

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 1 0 1 0 | 1 r r r | direct addr. |
|---|---|---|

**Operation:**
MOV
(Rn) ← (direct)

**MOV Rn,#data**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 1 1 | 1 r r r | immediate data |
|---|---|---|

**Operation:**
MOV
(Rn) ← #data

**MOV direct,A**

**Bytes:** 2

**Cycles:** 1

**Encoding:**


**Operation:** MOV
(direct) ← (A)

**MOV direct,Rn**

**Bytes:** 2

**Cycles:** 2

**Encoding:**


**Operation:** MOV
(direct) ← (Rn)

**MOV direct,direct**

**Bytes:** 3

**Cycles:** 2

**Encoding:**


**Operation:** MOV
(direct) ← (direct)

**MOV direct,@Ri**

**Bytes:** 2

**Cycles:** 2

**Encoding:**


**Operation:** MOV
(direct) ← ((Ri))

**MOV direct,#data**

**Bytes:** 3

**Cycles:** 2

**Encoding:**


**Operation:** MOV
(direct) ← #data

**MOV @Ri,A**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 1 1 | 0 1 1 i |
|---|---|

**Operation:** MOV
((Ri)) ← (A)

**MOV @Ri,direct**

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 1 0 1 0 | 0 1 1 i | direct addr. |
|---|---|---|

**Operation:** MOV
((Ri)) ← (direct)

**MOV @Ri,#data**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 1 1 | 0 1 1 i | immediate data |
|---|---|---|

**Operation:** MOV
((RI)) ← #data

**MOV <dest-bit>,<src-bit>**

**Function:** Move bit data

**Description:** The Boolean variable indicated by the second operand is copied into the location specified by the first operand. One of the operands must be the carry flag; the other may be any directly addressable bit. No other register or flag is affected.

**Example:** The carry flag is originally set. The data present at input Port 3 is 11000101B. The data previously written to output Port 1 is 35H (00110101B).

```
MOV  P1.3,C
MOV  C,P3.3
MOV  P1.2,C
```

will leave the carry cleared and change Port 1 to 39H (00111001B).

**MOV C,bit**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 1 0 1 0 | 0 0 1 0 | bit address |
|---|---|---|

**Operation:** MOV
(C) ← (bit)

**MOV bit,C**

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 1 0 0 1 | 0 0 1 0 | bit address |
|---|---|---|

**Operation:** MOV
(bit) ← (C)

**MOV DPTR,#data16**

**Function:** Load Data Pointer with a 16-bit constant

**Description:** The Data Pointer is loaded with the 16-bit constant indicated. The 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the low-order byte. No flags are affected.

This is the only instruction which moves 16 bits of data at once.

**Example:** The instruction,

MOV DPTR,#1234H

will load the value 1234H into the Data Pointer: DPH will hold 12H and DPL will hold 34H.

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 1 0 0 1 | 0 0 0 0 | immed. data15-8 | immed. data7-0 |
|---|---|---|---|

**Operation:** MOV
(DPTR) ← #data$_{15-0}$
DPH □ DPL ← #data$_{15-8}$ □ #data$_{7-0}$

## MOVC A,@A+ <base-reg>

**Function:** Move Code byte

**Description:** The MOVC instructions load the Accumulator with a code byte, or constant from program memory. The address of the byte fetched is the sum of the original unsigned eight-bit Accumulator contents and the contents of a sixteen-bit base register, which may be either the Data Pointer or the PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the Accumulator; otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.

**Example:** A value between 0 and 3 is in the Accumulator. The following instructions will translate the value in the Accumulator to one of four values defined by the DB (define byte) directive.

```
REL_PC: INC   A
        MOVC  A,@A+PC
        RET
        DB    66H
        DB    77H
        DB    88H
        DB    99H
```

If the subroutine is called with the Accumulator equal to 01H, it will return with 77H in the Accumulator. The INC A before the MOVC instruction is needed to "get around" the RET instruction above the table. If several bytes of code separated the MOVC from the table, the corresponding number would be added to the Accumulator instead.

## MOVC A,@A+DPTR

**Bytes:** 1

**Cycles:** 2

**Encoding:**

| 1 0 0 1 | 0 0 1 1 |
|---|---|

**Operation:** MOVC
(A) ← ((A) + (DPTR))

## MOVC A,@A + PC

**Bytes:** 1

**Cycles:** 2

**Encoding:**

| 1 0 0 0 | 0 0 1 1 |
|---|---|

**Operation:** MOVC
(PC) ← (PC) + 1
(A) ← ((A) + (PC))

**MOVX <dest-byte>,<src-byte>**

**Function:** Move External

**Description:** The MOVX instructions transfer data between the Accumulator and a byte of external data memory, hence the "X" appended to MOV. There are two types of instructions, differing in whether they provide an eight-bit or sixteen-bit indirect address to the external data RAM.

In the first type, the contents of R0 or R1 in the current register bank provide an eight-bit address multiplexed with data on P0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array. For somewhat larger arrays, any output port pins can be used to output higher-order address bits. These pins would be controlled by an output instruction preceding the MOVX.

In the second type of MOVX instruction, the Data Pointer generates a sixteen-bit address. P2 outputs the high-order eight address bits (the contents of DPH) while P0 multiplexes the low-order eight bits (DPL) with data. The P2 Special Function Register retains its previous contents while the P2 output buffers are emitting the contents of DPH. This form is faster and more efficient when accessing very large data arrays (up to 64K bytes), since no additional instructions are needed to set up the output ports.

It is possible in some situations to mix the two MOVX types. A large RAM array with its high-order address lines driven by P2 can be addressed via the Data Pointer, or with code to output high-order address bits to P2 followed by a MOVX instruction using R0 or R1.

**Example:** An external 256 byte RAM using multiplexed address/data lines (e.g., an Intel 8155 RAM/I/O/Timer) is connected to the 8051 Port 0. Port 3 provides control lines for the external RAM. Ports 1 and 2 are used for normal I/O. Registers 0 and 1 contain 12H and 34H. Location 34H of the external RAM holds the value 56H. The instruction sequence,

```
MOVX  A,@R1

MOVX  @R0,A
```

copies the value 56H into both the Accumulator and external RAM location 12H.

**MOVX A,@Ri**

**Bytes:** 1

**Cycles:** 2

**Encoding:**

| 1 1 1 0 | 0 0 1 i |
|---|---|

**Operation:** MOVX
(A) ← ((Ri))

**MOVX A,@DPTR**

**Bytes:** 1

**Cycles:** 2

**Encoding:**

| 1 1 1 0 | 0 0 0 0 |
|---|---|

**Operation:** MOVX
(A) ← ((DPTR))

**MOVX @Ri,A**

**Bytes:** 1

**Cycles:** 2

**Encoding:**

| 1 1 1 1 | 0 0 1 i |
|---|---|

**Operation:** MOVX
((Ri)) ← (A)

**MOVX @DPTR,A**

**Bytes:** 1

**Cycles:** 2

**Encoding:**

| 1 1 1 1 | 0 0 0 0 |
|---|---|

**Operation:** MOVX
(DPTR) ← (A)

## MUL AB

**Function:** Multiply

**Description:** MUL AB multiplies the unsigned eight-bit integers in the Accumulator and register B. The low-order byte of the sixteen-bit product is left in the Accumulator, and the high-order byte in B. If the product is greater than 255 (0FFH) the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.

**Example:** Originally the Accumulator holds the value 80 (50H). Register B holds the value 160 (0A0H). The instruction,

```
MUL   AB
```

will give the product 12,800 (3200H), so B is changed to 32H (00110010B) and the Accumulator is cleared. The overflow flag is set, carry is cleared.

**Bytes:** 1

**Cycles:** 4

**Encoding:**

| 1 0 1 0 | 0 1 0 0 |
|---|---|

**Operation:**

MUL
$(A)_{7\text{-}0} \leftarrow (A) \times (B)$
$(B)_{15\text{-}8}$

## NOP

**Function:** No Operation

**Description:** Execution continues at the following instruction. Other than the PC, no registers or flags are affected.

**Example:** It is desired to produce a low-going output pulse on bit 7 of Port 2 lasting exactly 5 cycles. A simple SETB/CLR sequence would generate a one-cycle pulse, so four additional cycles must be inserted. This may be done (assuming no interrupts are enabled) with the instruction sequence,

```
CLR     P2.7
NOP
NOP
NOP
NOP
SETB    P2.7
```

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 0 0 | 0 0 0 0 |
|---|---|

**Operation:**

NOP
$(PC) \leftarrow (PC) + 1$

**ORL <dest-byte> <src-byte>**

**Function:** Logical-OR for byte variables

**Description:** ORL performs the bitwise logical-OR operation between the indicated variables, storing the results in the destination byte. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

*Note:* When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

**Example:** If the Accumulator holds 0C3H (11000011B) and R0 holds 55H (01010101B) then the instruction,

```
ORL  A,R0
```

will leave the Accumulator holding the value 0D7H (11010111B).

When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable computed in the Accumulator at run-time. The instruction,

```
ORL  P1,#00110010B
```

will set bits 5, 4, and 1 of output Port 1.

**ORL A,Rn**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 1 0 0 | 1 r r r |
|---|---|

**Operation:** ORL
(A) ← (A) ∨ (Rn)

**ORL A,direct**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 0 0 | 0 1 0 1 | direct address |
|---|---|---|

**Operation:** ORL
(A) ← (A) ∨ (direct)

**ORL A,@Ri**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 1 0 0 | 0 1 1 i |
|---|---|

**Operation:** ORL
(A) ← (A) ∨ ((Ri))

**ORL A,#data**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 0 0 | 0 1 0 0 | immediate data |
|---|---|---|

**Operation:** ORL
(A) ← (A) ∨ #data

**ORL direct,A**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 0 0 | 0 0 1 0 | direct address |
|---|---|---|

**Operation:** ORL
(direct) ← (direct) ∨ (A)

**ORL direct,#data**

**Bytes:** 3

**Cycles:** 2

**Encoding:**

| 0 1 0 0 | 0 0 1 1 | direct addr. | immediate data |
|---|---|---|---|

**Operation:** ORL
(direct) ← (direct) ∨ #data

## ORL C,<src-bit>

**Function:** Logical-OR for bit variables

**Description:** Set the carry flag if the Boolean value is a logical 1; leave the carry in its current state otherwise . A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

**Example:** Set the carry flag if and only if P1.0 = 1, ACC. 7 = 1, or OV = 0:

```
MOV  C,P1.0     ;LOAD CARRY WITH INPUT PIN P10
ORL  C,ACC.7    ;OR CARRY WITH THE ACC. BIT 7
ORL  C,/OV      ;OR CARRY WITH THE INVERSE OF OV.
```

## ORL C,bit

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 0 1 1 1 | 0 0 1 0 | bit address |
|---|---|---|

**Operation:** ORL
(C) ← (C) ∨ (bit)

## ORL C,/bit

**Bytes:** 2

**Cycles:** 2

**Encoding:**

| 1 0 1 0 | 0 0 0 0 | bit address |
|---|---|---|

**Operation:** ORL
(C) ← (C) ∨ $(\overline{bit})$

**POP direct**

**Function:** Pop from stack.

**Description:** The contents of the internal RAM location addressed by the Stack Pointer is read, and the Stack Pointer is decremented by one. The value read is then transferred to the directly addressed byte indicated. No flags are affected.

**Example:** The Stack Pointer originally contains the value 32H, and internal RAM locations 30H through 32H contain the values 20H, 23H, and 01H, respectively. The instruction sequence,

```
POP   DPH
POP   DPL
```

will leave the Stack Pointer equal to the value 30H and the Data Pointer set to 0123H. At this point the instruction,

```
POP   SP
```

will leave the Stack Pointer set to 20H. Note that in this special case the Stack Pointer was decremented to 2FH before being loaded with the value popped (20H).

**Bytes:** 2

**Cycles:** 2

**Encoding:**


**Operation:**

```
POP
(direct) ← ((SP))
(SP) ← (SP) − 1
```

**PUSH direct**

**Function:** Push onto stack

**Description:** The Stack Pointer is incremented by one. The contents of the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. Otherwise no flags are affected.

**Example:** On entering an interrupt routine the Stack Pointer contains 09H. The Data Pointer holds the value 0123H. The instruction sequence,

```
PUSH   DPL
PUSH   DPH
```

will leave the Stack Pointer set to 0BH and store 23H and 01H in internal RAM locations 0AH and 0BH, respectively.

**Bytes:** 2

**Cycles:** 2

**Encoding:**


**Operation:**

```
PUSH
(SP) ← (SP) + 1
((SP)) ← (direct)
```

## RET

**Function:** Return from subroutine

**Description:** RET pops the high- and low-order bytes of the PC successively from the stack, decrementing the Stack Pointer by two. Program execution continues at the resulting address, generally the instruction immediately following an ACALL or LCALL. No flags are affected.

**Example:** The Stack Pointer originally contains the value 0BH. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

RET

will leave the Stack Pointer equal to the value 09H. Program execution will continue at location 0123H.

**Bytes:** 1

**Cycles:** 2

**Encoding:**

| 0 0 1 0 | 0 0 1 0 |
|---|---|

**Operation:**
RET
$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$
$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$

## RETI

**Function:** Return from interrupt

**Description:** RETI pops the high- and low-order bytes of the PC successively from the stack, and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. The Stack Pointer is left decremented by two. No other registers are affected; the PSW is *not* automatically restored to its pre-interrupt status. Program execution continues at the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower- or same-level interrupt had been pending when the RETI instruction is executed, that one instruction will be executed before the pending interrupt is processed.

**Example:** The Stack Pointer originally contains the value 0BH. An interrupt was detected during the instruction ending at location 0122H. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

RETI

will leave the Stack Pointer equal to 09H and return program execution to location 0123H.

**Bytes:** 1

**Cycles:** 2

**Encoding:**

| 0 0 1 1 | 0 0 1 0 |
|---|---|

**Operation:**
RETI
$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$
$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$

## RL A

**Function:** Rotate Accumulator Left

**Description:** The eight bits in the Accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position. No flags are affected.

**Example:** The Accumulator holds the value 0C5H (11000101B). The instruction,

```
RL   A
```

leaves the Accumulator holding the value 8BH (10001011B) with the carry unaffected.

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 1 0 | 0 0 1 1 |
|---|---|

**Operation:**

RL
$(A_n + 1) \leftarrow (An) \quad n = 0 - 6$
$(A0) \leftarrow (A7)$

## RLC A

**Function:** Rotate Accumulator Left through the Carry flag

**Description:** The eight bits in the Accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

**Example:** The Accumulator holds the value 0C5H (11000101B), and the carry is zero. The instruction,

```
RLC  A
```

leaves the Accumulator holding the value 8BH (10001010B) with the carry set.

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 1 1 | 0 0 1 1 |
|---|---|

**Operation:**

RLC
$(An + 1) \leftarrow (An) \quad n = 0 - 6$
$(A0) \leftarrow (C)$
$(C) \leftarrow (A7)$

## RR A

**Function:** Rotate Accumulator Right

**Description:** The eight bits in the Accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.

**Example:** The Accumulator holds the value 0C5H (11000101B). The instruction,

RR A

leaves the Accumulator holding the value 0E2H (11100010B) with the carry unaffected.

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 0 0 | 0 0 1 1 |
|---|---|

**Operation:** RR
(An) ← ($A_n$ + 1) n = 0 − 6
(A7) ← (A0)

## RRC A

**Function:** Rotate Accumulator Right through Carry flag

**Description:** The eight bits in the Accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag; the original value of the carry flag moves into the bit 7 position. No other flags are affected.

**Example:** The Accumulator holds the value 0C5H (11000101B), the carry is zero. The instruction,

RRC A

leaves the Accumulator holding the value 62 (01100010B) with the carry set.

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 0 0 1 | 0 0 1 1 |
|---|---|

**Operation:** RRC
(An) ← (An + 1) n = 0 − 6
(A7) ← (C)
(C) ← (A0)

## SETB <bit>

**Function:** Set Bit

**Description:** SETB sets the indicated bit to one. SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

**Example:** The carry flag is cleared. Output Port 1 has been written with the value 34H (00110100B). The instructions,

SETB C

SETB P1.0

will leave the carry flag set to 1 and change the data output on Port 1 to 35H (00110101B).

### SETB C

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 0 1 | 0 0 1 1 |
|---|---|

**Operation:** SETB
(C) ← 1

### SETB bit

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 1 1 0 1 | 0 0 1 0 | bit address |
|---|---|---|

**Operation:** SETB
(bit) ← 1

## SJMP rel

**Function:** Short Jump

**Description:** Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

**Example:** The label "RELADR" is assigned to an instruction at program memory location 0123H. The instruction,

SJMP RELADR

will assemble into location 0100H. After the instruction is executed, the PC will contain the value 0123H.

(*Note:* Under the above conditions the instruction following SJMP will be at 102H. Therefore, the displacement byte of the instruction will be the relative offset (0123H-0102H) = 21H. Put another way, an SJMP with a displacement of 0FEH would be a one-instruction infinite loop.)

**Bytes:** 2

**Cycles:** 2

**Encoding:**


**Operation:** SJMP
(PC) ← (PC) + 2
(PC) ← (PC) + rel

**SUBB A,<src-byte>**

**Function:** Subtract with borrow

**Description:** SUBB subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7, and clears C otherwise. (If C was set *before* executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision subtraction, so the carry is subtracted from the Accumulator along with the source operand.) AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

When subtracting signed integers OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: register, direct, register-indirect, or immediate.

**Example:** The Accumulator holds 0C9H (11001001B), register 2 holds 54H (01010100B), and the carry flag is set. The instruction,

SUBB A,R2

will leave the value 74H (01110100B) in the accumulator, with the carry flag and AC cleared but OV set.

Notice that 0C9H minus 54H is 75H. The difference between this and the above result is due to the carry (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision subtraction, it should be explicitly cleared by a CLR C instruction.

**SUBB A,Rn**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 0 0 1 | 1 r r r |
|---|---|

**Operation:** SUBB
(A) ← (A) − (C) − (Rn)

**SUBB A,direct**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 1 0 0 1 | 0 1 0 1 | direct address |
|---|---|---|

**Operation:** SUBB
(A) ← (A) − (C) − (direct)

**SUBB A,@Ri**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 0 0 1 | 0 1 1 i |
|---|---|

**Operation:** SUBB
(A) ← (A) − (C) − ((Ri))

**SUBB A,#data**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 1 0 0 1 | 0 1 0 0 | immediate data |
|---|---|---|

**Operation:** SUBB
(A) ← (A) − (C) − #data

**SWAP A**

**Function:** Swap nibbles within the Accumulator

**Description:** SWAP A interchanges the low- and high-order nibbles (four-bit fields) of the Accumulator (bits 3-0 and bits 7-4). The operation can also be thought of as a four-bit rotate instruction. No flags are affected.

**Example:** The Accumulator holds the value 0C5H (11000101B). The instruction,

SWAP A

leaves the Accumulator holding the value 5CH (01011100B).

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 0 0 | 0 1 0 0 |
|---|---|

**Operation:** SWAP
$(A_{3-0}) \rightleftarrows (A_{7-4})$

## XCH A,<byte>

**Function:** Exchange Accumulator with byte variable

**Description:** XCH loads the Accumulator with the contents of the indicated variable, at the same time writing the original Accumulator contents to the indicated variable. The source/destination operand can use register, direct, or register-indirect addressing.

**Example:** R0 contains the address 20H. The Accumulator holds the value 3FH (00111111B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

```
XCH   A,@R0
```

will leave RAM location 20H holding the values 3FH (00111111B) and 75H (01110101B) in the accumulator.

## XCH A,Rn

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 0 0 | 1 r r r |
|---|---|

**Operation:**
XCH
(A) $\rightleftarrows$ (Rn)

## XCH A,direct

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 1 1 0 0 | 0 1 0 1 | direct address |
|---|---|---|

**Operation:**
XCH
(A) $\rightleftarrows$ (direct)

## XCH A,@Ri

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 0 0 | 0 1 1 i |
|---|---|

**Operation:**
XCH
(A) $\rightleftarrows$ ((Ri))

## XCHD A,@Ri

**Function:** Exchange Digit

**Description:** XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (bits 7-4) of each register are not affected. No flags are affected.

**Example:** R0 contains the address 20H. The Accumulator holds the value 36H (00110110B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

```
XCHD  A,@R0
```

will leave RAM location 20H holding the value 76H (01110110B) and 35H (00110101B) in the Accumulator.

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 1 1 0 1 | 0 1 1 i |
|---|---|

**Operation:** XCHD
$(A_{3-0}) \rightleftarrows ((Ri_{3-0}))$

## XRL <dest-byte>,<src-byte>

**Function:** Logical Exclusive-OR for byte variables

**Description:** XRL performs the bitwise logical Exclusive-OR operation between the indicated variables, storing the results in the destination. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

(*Note:* When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.)

**Example:** If the Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) then the instruction,

```
XRL  A,R0
```

will leave the Accumulator holding the value 69H (01101001B).

When the destination is a directly addressed byte, this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte, either a constant contained in the instruction or a variable computed in the Accumulator at run-time. The instruction,

```
XRL  P1,#00110001B
```

will complement bits 5, 4, and 0 of output Port 1.

**XRL A,Rn**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 1 1 0 | 1 r r r |
|---|---|

**Operation:** XRL
(A) ← (A) ∀ (Rn)

**XRL A,direct**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 1 0 | 0 1 0 1 | direct address |
|---|---|---|

**Operation:** XRL
(A) ← (A) ∀ (direct)

**XRL A,@Ri**

**Bytes:** 1

**Cycles:** 1

**Encoding:**

| 0 1 1 0 | 0 1 1 i |
|---|---|

**Operation:** XRL
(A) ← (A) ∀ ((Ri))

**XRL A,#data**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 1 0 | 0 1 0 0 | immediate data |
|---|---|---|

**Operation:** XRL
(A) ← (A) ∀ #data

**XRL direct,A**

**Bytes:** 2

**Cycles:** 1

**Encoding:**

| 0 1 1 0 | 0 0 1 0 | direct address |
|---|---|---|

**Operation:** XRL
(direct) ← (direct) ∀ (A)

**XRL direct,#data**

**Bytes:** 3

**Cycles:** 2

**Encoding:**


**Operation:** XRL
(direct) ← (direct) ⊻ #data

# 8051, 8052 and 80C51 Hardware Description

3

# 8051, 8052 and 80C51 Hardware Description

## CONTENTS



## CONTENTS

intel

# 8051, 8052 AND 80C51 HARDWARE DESCRIPTION

## INTRODUCTION

This chapter presents a comprehensive description of the on-chip hardware features of the MCS®-51 microcontrollers. Included in this description are

- The port drivers and how they function both as ports and, for Ports 0 and 2, in bus operations
- The Timer/Counters
- The Serial Interface
- The Interrupt System
- Reset
- The Reduced Power Modes in the CHMOS devices
- The EPROM versions of the 8051AH, 8052AH and 80C51BH

The devices under consideration are listed in Table 1. As it becomes unwieldy to be constantly referring to each of these devices by their individual names, we will adopt a convention of referring to them generically as 8051s and 8052s, unless a specific member of the group is being referred to, in which case it will be specifically named. The "8051s" include the 8051AH, 80C51BH, and their ROMless and EPROM versions. The "8052s" are the 8052AH, 8032AH and 8752BH.

Figure 1 shows a functional block diagram of the 8051s and 8052s.

**Table 1. The MCS-51 Family of Microcontrollers**

| Device Name | ROMless Version | EPROM Version | ROM Bytes | RAM Bytes | 16-bit Timers | Ckt Type |
|---|---|---|---|---|---|---|
| 8051AH | 8031AH | 8751H, 8751BH | 4K | 128 | 2 | HMOS |
| 8052AH | 8032AH | 8752BH | 8K | 256 | 3 | HMOS |
| 80C51BH | 80C31BH | 87C51 | 4K | 128 | 2 | CHMOS |

## Special Function Registers

A map of the on-chip memory area called SFR (Special Function Register) space is shown in Figure 2. SFRs marked by parentheses are resident in the 8052s but not in the 8051s.

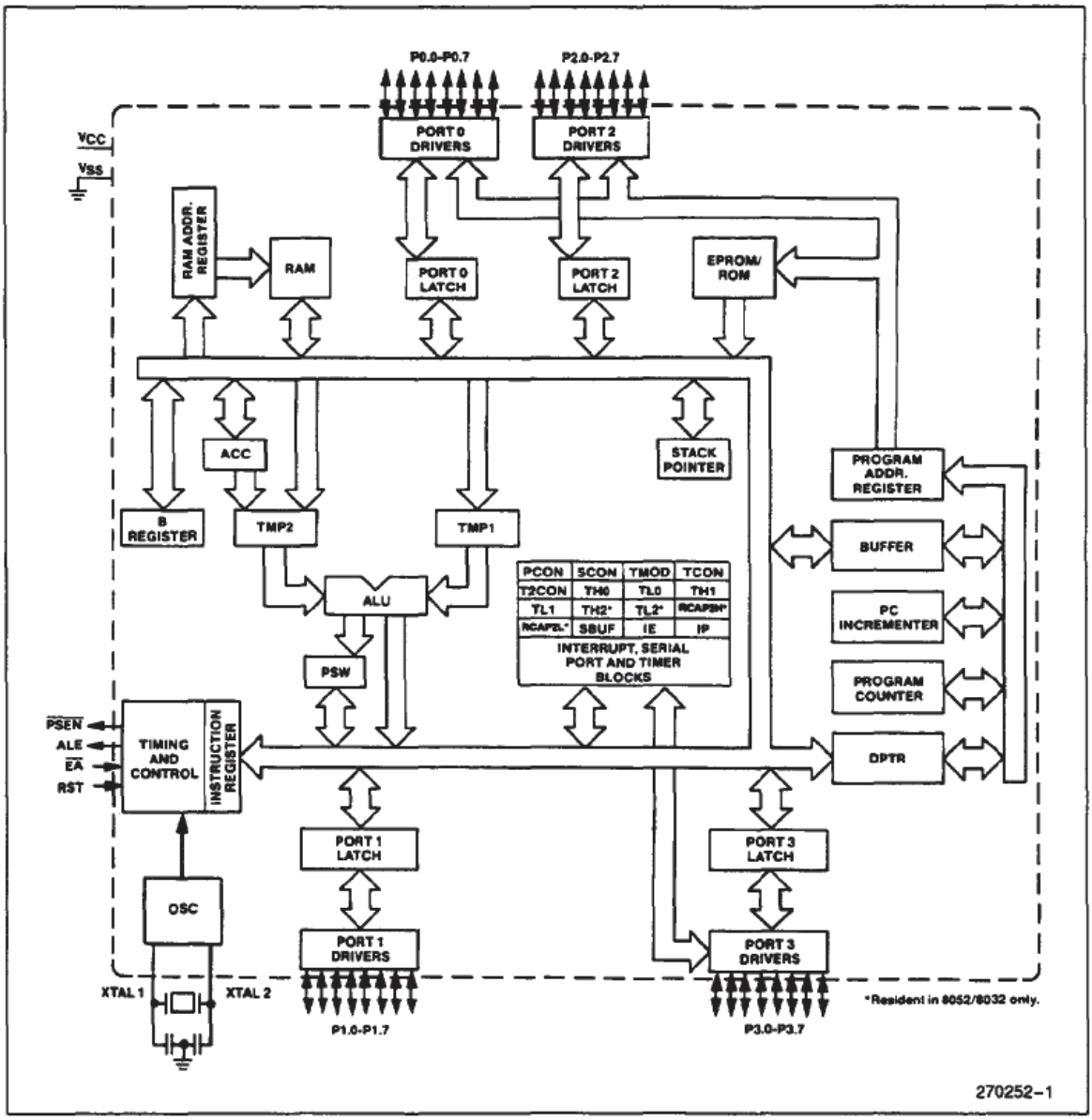



**Figure 1. MCS-51 Architectural Block Diagram**

| | 8 Bytes | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| F8 | | | | | | | | | FF |
| F0 | B | | | | | | | | F7 |
| E8 | | | | | | | | | EF |
| E0 | ACC | | | | | | | | E7 |
| D8 | | | | | | | | | DF |
| D0 | PSW | | | | | | | | D7 |
| C8 | (T2CON) | | (RCAP2L) | (RCAP2H) | (TL2) | (TH2) | | | CF |
| C0 | | | | | | | | | C7 |
| B8 | IP | | | | | | | | BF |
| B0 | P3 | | | | | | | | B7 |
| A8 | IE | | | | | | | | AF |
| A0 | P2 | | | | | | | | A7 |
| 98 | SCON | SBUF | | | | | | | 9F |
| 90 | P1 | | | | | | | | 97 |
| 88 | TCON | TMOD | TL0 | TL1 | TH0 | TH1 | | | 8F |
| 80 | P0 | SP | DPL | DPH | | | | PCON | 87 |

**Figure 2. SFR Map. ( . . . ) Indicates Resident in 8052s, not in 8051s**

Note that not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

User software should not write 1s to these unimplemented locations, since they may be used in future MCS-51 products to invoke new features. In that case the reset or inactive values of the new bits will always be 0, and their active values will be 1.

The functions of the SFRs are outlined below.

### ACCUMULATOR

ACC is the Accumulator register. The mnemonics for Accumulator-Specific instructions, however, refer to the Accumulator simply as A.

### B REGISTER

The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

### PROGRAM STATUS WORD

The PSW register contains program status information as detailed in Figure 3.

### STACK POINTER

The Stack Pointer Register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

### DATA POINTER

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

### PORTS 0 TO 3

P0, P1, P2 and P3 are the SFR latches of Ports 0, 1, 2 and 3, respectively.

### SERIAL DATA BUFFER

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

### TIMER REGISTERS

Register pairs (TH0, TL0), (TH1, TL1), and (TH2, TL2) are the 16-bit Counting registers for Timer/Counters 0, 1, and 2, respectively.

### CAPTURE REGISTERS

The register pair (RCAP2H, RCAP2L) are the Capture registers for the Timer 2 "Capture Mode." In this mode, in response to a transition at the 8052's T2EX pin, TH2 and TL2 are copied into RCAP2H and RCAP2L. Timer 2 also has a 16-bit auto-reload mode, and RCAP2H and RCAP2L hold the reload value for this mode. More about Timer 2's features in a later section.

### CONTROL REGISTERS

Special Function Registers IP, IE, TMOD, TCON, T2CON, SCON, and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port. They are described in later sections.

| (MSB) | | | | | | | (LSB) |
|---|---|---|---|---|---|---|---|
| CY | AC | F0 | RS1 | RS0 | OV | — | P |

| Symbol | Position | Name and Significance |
|---|---|---|
| CY | PSW.7 | Carry flag. |
| AC | PSW.6 | Auxiliary Carry flag. (For BCD operations.) |
| F0 | PSW.5 | Flag 0 (Available to the user for general purposes.) |
| RS1 | PSW.4 | Register bank select control bits 1 & |
| RS0 | PSW.3 | 0. Set/cleared by software to determine working register bank (see Note). |
| OV | PSW.2 | Overflow flag. |
| — | PSW.1 | User definable flag. |
| P | PSW.0 | Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the Accumulator, i.e., even parity. |

**NOTE:**
The contents of (RS1, RS0) enable the working register banks as follows:

(0.0)—Bank 0 (00H–07H)
(0.1)—Bank 1 (08H–0FH)
(1.0)—Bank 2 (10H–17H)
(1.1)—Bank 3 (18H–1FH)

**Figure 3. PSW: Program Status Word Register**

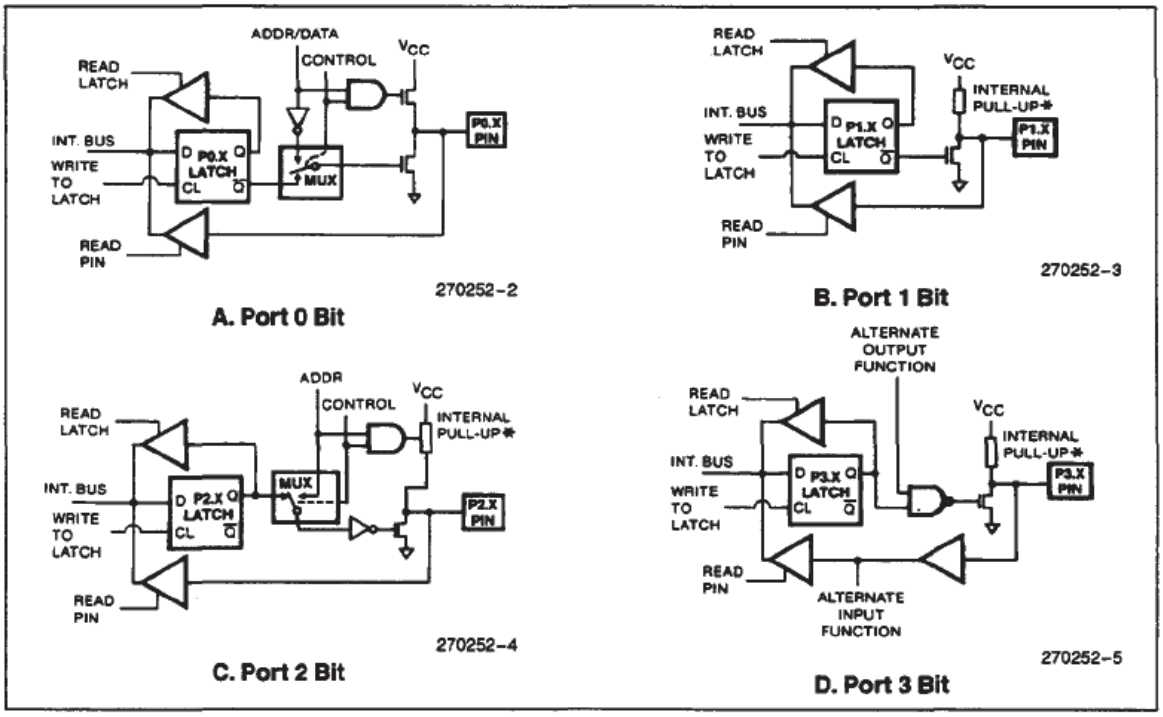



**Figure 4. 8051 Port Bit Latches and I/O Buffers**
*See Figure 5 for details of the internal pullup.

## PORT STRUCTURES AND OPERATION

All four ports in the 8051 are bidirectional. Each consists of a latch (Special Function Registers P0 through P3), an output driver, and an input buffer.

The output drivers of Ports 0 and 2, and the input buffers of Port 0, are used in accesses to external memory. In this application, Port 0 outputs the low byte of the external memory address, time-multiplexed with the byte being written or read. Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise the Port 2 pins continue to emit the P2 SFR content.

All the Port 3 pins, and (in the 8052) two Port 1 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed on the following page.

| Port Pin | Alternate Function |
|---|---|
| *P1.0 | T2 (Timer/Counter 2 external input) |
| *P1.1 | T2EX (Timer/Counter 2 Capture/Reload trigger) |
| P3.0 | RXD (serial input port) |
| P3.1 | TXD (serial output port) |
| P3.2 | $\overline{INT0}$ (external interrupt) |
| P3.3 | $\overline{INT1}$ (external interrupt) |
| P3.4 | T0 (Timer/Counter 0 external input) |
| P3.5 | T1 (Timer/Counter 1 external input) |
| P3.6 | $\overline{WR}$ (external Data Memory write strobe) |
| P3.7 | $\overline{RD}$ (external Data Memory read strobe) |

*P1.0 and P1.1 serve these alternate functions only on the 8052.

The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise the port pin is stuck at 0.

## I/O Configurations

Figure 4 shows a functional diagram of a typical bit latch and I/O buffer in each of the four ports. The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which will clock in a value from the internal bus in response to a "write to latch" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in response to a "read latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal. More about that later.

As shown in Figure 4, the output drivers of Ports 0 and 2 are switchable to an internal ADDR and ADDR/DATA bus by an internal CONTROL signal for use in external memory accesses. During external memory accesses, the P2 SFR remains unchanged, but the P0 SFR gets 1s written to it.

Also shown in Figure 4, is that if a P3 bit latch contains a 1, then the output level is controlled by the signal labeled "alternate output function." The actual P3.X pin level is always available to the pin's alternate input function, if any.

Ports 1, 2, and 3 have internal pullups. Port 0 has open drain outputs. Each I/O line can be independently used as an input or an output. (Ports 0 and 2 may not be used as general purpose I/O when being used as the ADDR/DATA BUS). To be used as an input, the port bit latch must contain a 1, which turns off the output driver FET. Then, for Ports 1, 2, and 3, the pin is pulled high by the internal pullup, but can be pulled low by an external source.

Port 0 differs in not having internal pullups. The pullup FET in the P0 output driver (see Figure 4) is used only when the Port is emitting 1s during external memory accesses. Otherwise the pullup FET is off. Consequently P0 lines that are being used as output port lines are open drain. Writing a 1 to the bit latch leaves both output FETs off, so the pin floats. In that condition it can be used a high-impedance input.

Because Ports 1, 2, and 3 have fixed internal pullups they are sometimes called "quasi-bidirectional" ports. When configured as inputs they pull high and will source current (IIL, in the data sheets) when externally pulled low. Port 0, on the other hand, is considered "true" bidirectional, because when configured as an input it floats.

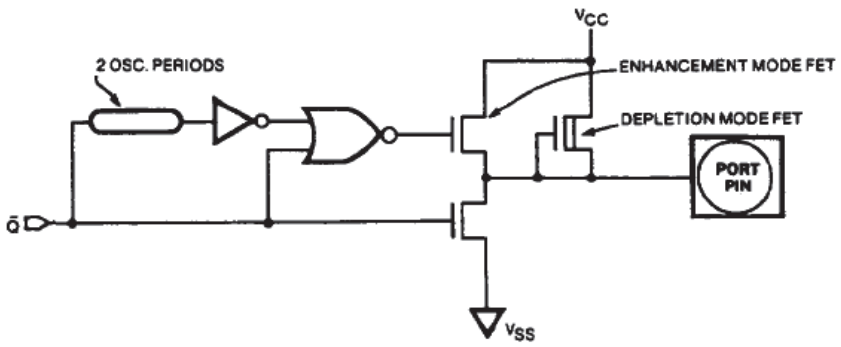
All the port latches in the 8051 have 1s written to them by the reset function. If a 0 is subsequently written to a port latch, it can be reconfigured as an input by writing a 1 to it.

## Writing to a Port

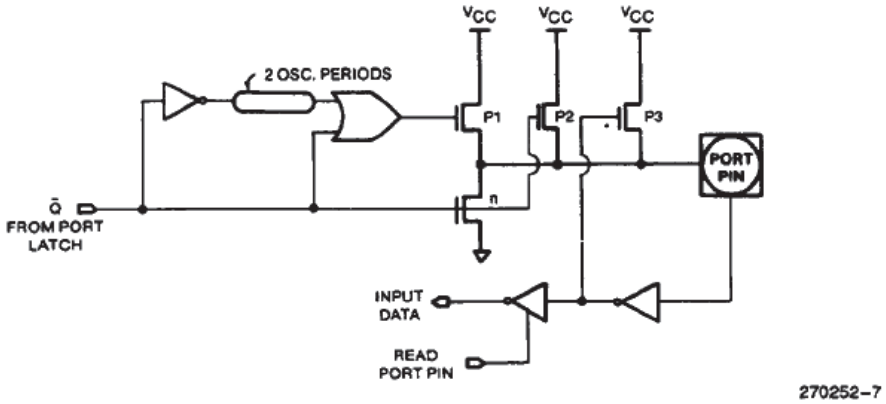
In the execution of an instruction that changes the value in a port latch, the new value arrives at the latch during S6P2 of the final cycle of the instruction. However, port latches are in fact sampled by their output buffers only during Phase 1 of any clock period. (During Phase 2 the output buffer holds the value it saw during the previous Phase 1). Consequently, the new value in the port latch won't actually appear at the output pin until the next Phase 1, which will be at S1P1 of the next machine cycle. See Figure 39 in the Internal Timing section.

If the change requires a 0-to-1 transition in Port 1, 2, or 3, an additional pullup is turned on during S1P1 and S1P2 of the cycle in which the transition occurs. This is done to increase the transition speed. The extra pullup can source about 100 times the current that the normal pullup can. It should be noted that the internal pullups are field-effect transistors, not linear resistors. The pullup arrangements are shown in Figure 5.

In HMOS versions of the 8051, the fixed part of the pullup is a depletion-mode transistor with the gate wired to the source. This transistor will allow the pin to source about 0.25 mA when shorted to ground. In parallel with the fixed pullup is an enhancement-mode transistor, which is activated during S1 whenever the port bit does a 0-to-1 transition. During this interval, if the port pin is shorted to ground, this extra transistor will allow the pin to source an additional 30 mA.




**A. HMOS Configuration. The enhancement mode transistor is turned on for 2 osc. periods after $\overline{Q}$ makes a 0-to-1 transition.**




**B. CHMOS Configuration. pFET 1 is turned on for 2 osc. periods after $\overline{Q}$ makes a 0-to-1 transition. During this time, pFET 1 also turns on pFET 3 through the inverter to form a latch which holds the 1. pFET 2 is also on.**

**Figure 5. Ports 1 And 3 HMOS And CHMOS Internal Pullup Configurations. Port 2 is Similar Except That It Holds The Strong Pullup On While Emitting 1s That Are Address Bits. (See Text, "Accessing External Memory".)**

In the CHMOS versions, the pullup consists of three pFETs. It should be noted that an n-channel FET (nFET) is turned on when a logical 1 is applied to its gate, and is turned off when a logical 0 is applied to its gate. A p-channel FET (pFET) is the opposite: it is on when its gate sees a 0, and off when its gate sees a 1.

pFET1 in Figure 5 is the transistor that is turned on for 2 oscillator periods after a 0-to-1 transition in the port latch. While it's on, it turns on pFET3 (a weak pullup), through the inverter. This inverter and pFET form a latch which hold the 1.

Note that if the pin is emitting a 1, a negative glitch on the pin from some external source can turn off pFET3, causing the pin to go into a float state. pFET2 is a very weak pullup which is on whenever the nFET is off, in traditional CMOS style. It's only about $1/10$ the strength of pFET3. Its function is to restore a 1 to the pin in the event the pin *had* a 1 and lost it to a glitch.

## Port Loading and Interfacing

The output buffers of Ports 1, 2, and 3 can each drive 4 LS TTL inputs. These ports on HMOS versions can be driven in a normal manner by any TTL or NMOS circuit. Both HMOS and CHMOS pins can be driven by open-collector and open-drain outputs, but note that 0-to-1 transitions will not be fast. In the HMOS device, if the pin is driven by an open-collector output, a 0-to-1 transition will have to be driven by the relatively weak depletion mode FET in Figure 5(A). In the CHMOS device, an input 0 turns off pullup pFET3, leaving only the very weak pullup pFET2 to drive the transition.

In external bus mode, Port 0 output buffers can each drive 8 LS TTL inputs. As port pins, they require external pullups to drive any inputs.

## Read-Modify-Write Feature

Some instructions that read a port read the latch and others read the pin. Which ones do which? The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write" instructions. The instructions listed below are read-modify-write instructions. When the destination operand is a port, or a port bit, these instructions read the latch rather than the pin:

| | |
|---|---|
| ANL | (logical AND, e.g., ANL P1, A) |
| ORL | (logical OR, e.g., ORL P2, A) |
| XRL | (logical EX-OR, e.g., XRL P3, A) |
| JBC | (jump if bit = 1 and clear bit, e.g., JBC P1.1, LABEL) |
| CPL | (complement bit, e.g., CPL P3.0) |
| INC | (increment, e.g., INC P2) |
| DEC | (decrement, e.g., DEC P2) |
| DJNZ | (decrement and jump if not zero, e.g., DJNZ P3, LABEL) |
| MOV, PX.Y, C | (move carry bit to bit Y of Port X) |
| CLR PX.Y | (clear bit Y of Port X) |
| SETB PX.Y | (set bit Y of Port X) |

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of a transistor. When a 1 is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as a 0. Reading the latch rather than the pin will return the correct value of 1.

## ACCESSING EXTERNAL MEMORY

Accesses to external memory are of two types: accesses to external Program Memory and accesses to external Data Memory. Accesses to external Program Memory use signal $\overline{PSEN}$ (program store enable) as the read strobe. Accesses to external Data Memory use $\overline{RD}$ or $\overline{WR}$ (alternate functions of P3.7 and P3.6) to strobe the memory. Refer to Figures 36 through 38 in the Internal Timing section.

Fetches from external Program Memory always use a 16-bit address. Accesses to external Data Memory can use either a 16-bit address (MOVX @DPTR) or an 8-bit address (MOVX @Ri).

Whenever a 16-bit address is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle. Note that the Port 2 drivers use the strong pullups during the entire time that they are emitting address bits that are 1s. This is during the execution of a MOVX @DPTR instruction. During this time the Port 2 latch (the Special Function Register) does not have to contain 1s, and the contents of the Port 2 SFR are not modified. If the external memory cycle is not immediately followed by another external memory cycle, the undisturbed contents of the Port 2 SFR will reappear in the next cycle.

If an 8-bit address is being used (MOVX @Ri), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging.

In any case, the low byte of the address is time-multiplexed with the data byte on Port 0. The ADDR/DATA signal drives both FETs in the Port 0 output buffers. Thus, in this application the Port 0 pins are not open-drain outputs, and do not require external pullups. Signal ALE (Address Latch Enable) should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before $\overline{WR}$ is activated, and remains there until after $\overline{WR}$ is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe is deactivated.

During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating whatever information the Port 0 SFR may have been holding. If the user writes to Port 0 during an external memory fetch, the incoming code byte is corrupted. Therefore, do not write to Port 0 if external program memory is used.

External Program Memory is accessed under two conditions:

1) Whenever signal $\overline{EA}$ is active; or
2) Whenever the program counter (PC) contains a number that is larger than 0FFFH (1FFFH for the 8052).

This requires that the ROMless versions have $\overline{EA}$ wired low to enable the lower 4K (8K for the 8032) program bytes to be fetched from external memory.

When the CPU is executing out of external Program Memory, all 8 bits of Port 2 are dedicated to an output function and may not be used for general purpose I/O. During external program fetches they output the high byte of the PC. During this time the Port 2 drivers use the strong pullups to emit PC bits that are 1s.

## TIMER/COUNTERS

The 8051 has two 16-bit Timer/Counter registers: Timer 0 and Timer 1. The 8052 has these two plus one

more: Timer 2. All three can be configured to operate either as timers or event counters.

In the "Timer" function, the register is incremented every machine cycle. Thus, one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is $1/12$ of the oscillator frequency.

In the "Counter" function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0, T1 or (in the 8052) T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is $1/24$ of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full machine cycle.

In addition to the "Timer" or "Counter" selection, Timer 0 and Timer 1 have four operating modes from which to select. Timer 2, in the 8052, has three modes of operation: "Capture," "Auto-Reload" and "baud rate generator."

## Timer 0 and Timer 1

These Timer/Counters are present in both the 8051 and the 8052. The "Timer" or "Counter" function is selected by control bits C/$\overline{T}$ in the Special Function Register TMOD (Figure 6). These two Timer/Counters have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for both Timer/Counters. Mode 3 is different. The four operating modes are described in the following text.

### MODE 0

Either Timer in Mode 0 is an 8-bit Counter with a divide-by-32 prescaler. This 13-bit timer is MCS-48 compatible. Figure 7 shows the Mode 0 operation as it applies to Timer 1.

In this mode, the Timer register is configured as a 13-Bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or $\overline{INT1}$ = 1. (Setting GATE = 1 allows the Timer to be controlled by external input $\overline{INT1}$, to facilitate pulse width measurements.) TR1 is a control bit in the Special Function Register TCON (Figure 8). GATE is in TMOD.

The 13-Bit register consists of all 8 bits of TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored. Setting the run flag (TR1) does not clear the registers.

Mode 0 operation is the same for Timer 0 as for Timer 1. Substitute TR0, TF0 and $\overline{INT0}$ for the corresponding Timer 1 signals in Figure 7. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

### MODE 1

Mode 1 is the same as Mode 0, except that the Timer register is being run with all 16 bits.


GATE — Gating control when set. Timer/Counter "x" is enabled only while "$\overline{INTx}$" pin is high and "TRx" control pin is set. When cleared Timer "x" is enabled whenever "TRx" control bit is set.

C/$\overline{T}$ — Timer or Counter Selector cleared for Timer operation (input from internal system clock). Set for Counter operation (input from "Tx" input pin).

| M1 | M0 | Operating Mode |
|---|---|---|
| 0 | 0 | 8-bit Timer/Counter "THx" with "TLx" as 5-bit prescaler. |
| 0 | 1 | 16-bit Timer/Counter "THx" and "TLx" are cascaded; there is no prescaler. |
| 1 | 0 | 8-bit auto-reload Timer/Counter "THx" holds a value which is to be reloaded into "TLx" each time it overflows. |
| 1 | 1 | (Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits. |
| 1 | 1 | (Timer 1) Timer/Counter 1 stopped. |

**Figure 6. TMOD: Timer/Counter Mode Control Register**

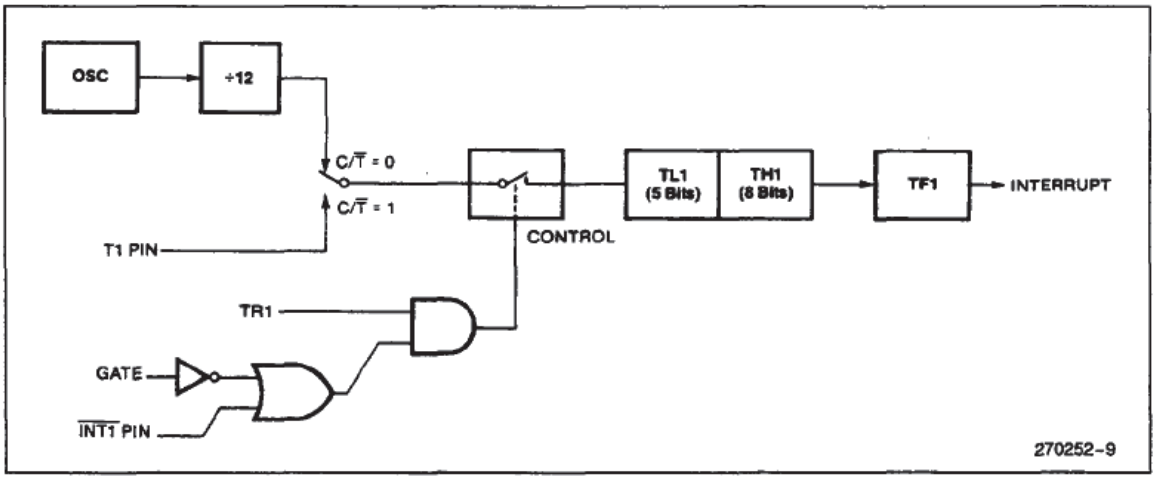
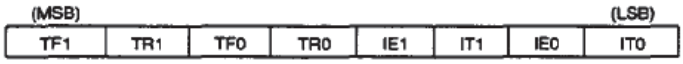



**Figure 7. Timer/Counter 1 Mode 0: 13-Bit Counter**




| Symbol | Position | Name and Significance |
|---|---|---|
| TF1 | TCON.7 | Timer 1 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine. |
| TR1 | TCON.6 | Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off. |
| TF0 | TCON.5 | Timer 0 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine. |
| TR0 | TCON.4 | Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off. |
| IE1 | TCON.3 | Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed. |
| IT1 | TCON.2 | Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts. |
| IE0 | TCON.1 | Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed. |
| IT0 | TCON.0 | Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts. |

**Figure 8.TCON: Timer/Counter Control Register**

### MODE 2

Mode 2 configures the Timer register as an 8-bit Counter (TL1) with automatic reload, as shown in Figure 9. Overflow from TL1 not only sets TF1, but also reloads TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.

Mode 2 operation is the same for Timer/Counter 0.

### MODE 3

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0.

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 10. TL0 uses the Timer 0 control bits: C/$\overline{T}$, GATE, TR0, $\overline{INT0}$, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the "Timer 1" interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer or counter. With Timer 0 in Mode 3, an 8051 can look like it has three Timer/Counters, and an 8052, like it has four. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3, or can still be used by the serial port as a baud rate generator, or in fact, in any application not requiring an interrupt.

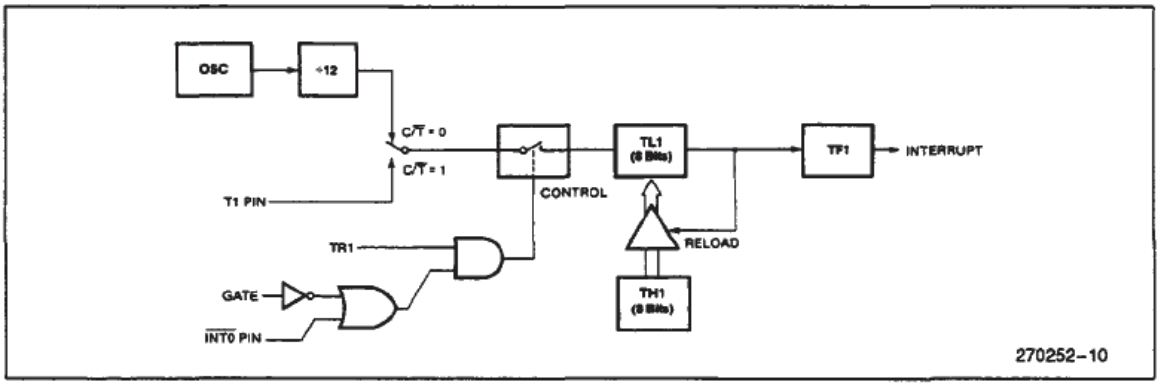



**Figure 9. Timer/Counter 1 Mode 2: 8-Bit Auto-Reload**

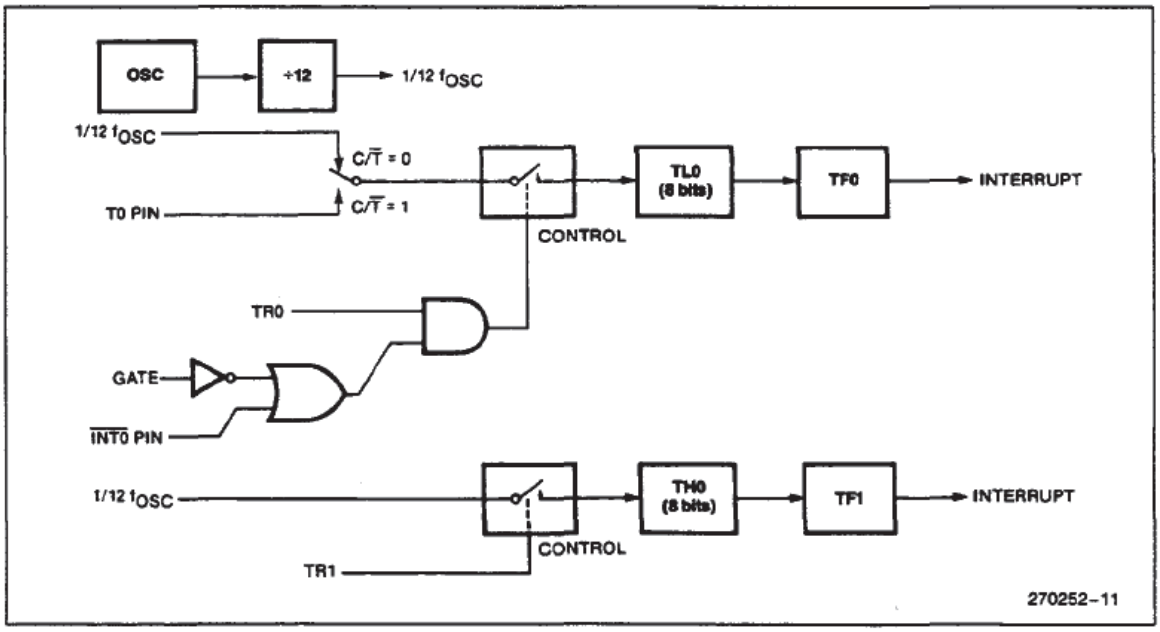


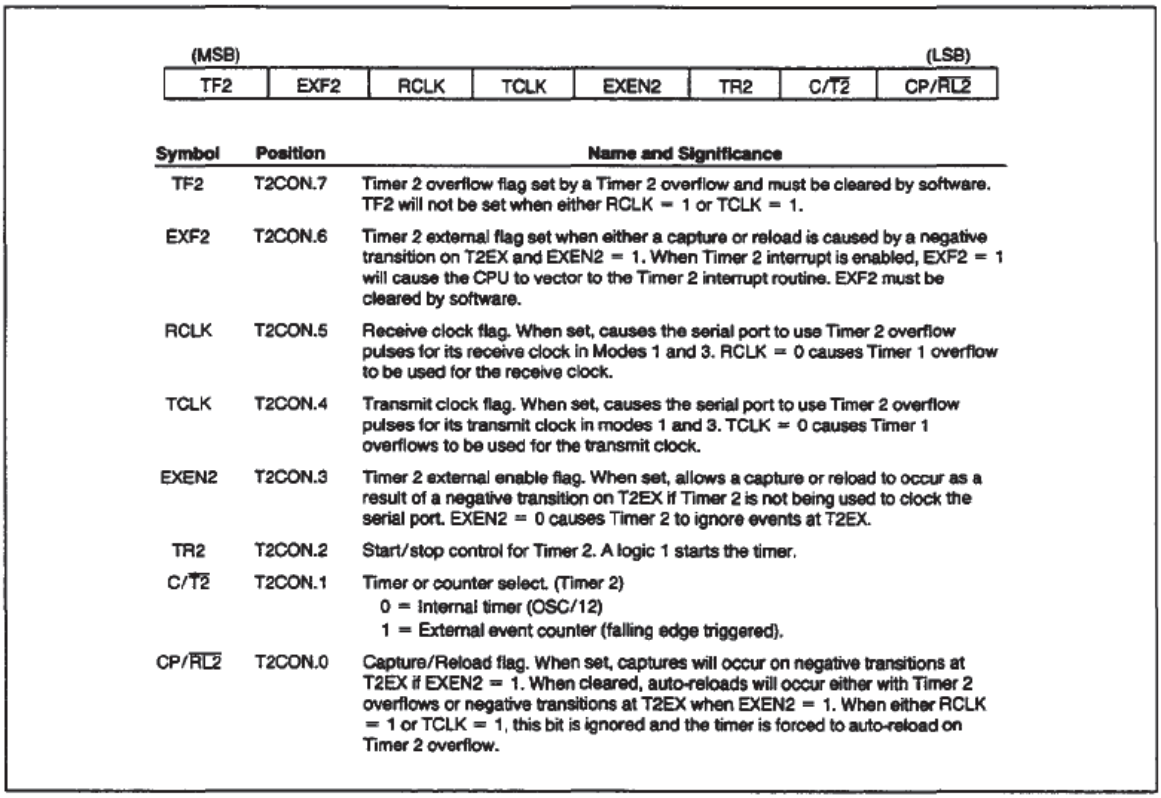

**Figure 10. Timer/Counter 0 Mode 3: Two 8-Bit Counters**

## Timer 2

Timer 2 is a 16-bit Timer/Counter which is present only in the 8052. Like Timers 0 and 1, it can operate either as a timer or as an event counter. This is selected by bit C/$\overline{T2}$ in the Special Function Register T2CON (Figure 11). It has three operating modes: "capture," "auto-load" and "baud rate generator," which are selected by bits in T2CON as shown in Table 2.

**Table 2. Timer 2 Operating Modes**

| RCLK + TCLK | CP/$\overline{RL2}$ | TR2 | Mode |
|---|---|---|---|
| 0 | 0 | 1 | 16-bit Auto-Reload |
| 0 | 1 | 1 | 16-bit Capture |
| 1 | X | 1 | Baud Rate Generator |
| X | X | 0 | (off) |



| (MSB) | | | | | | | (LSB) |
|---|---|---|---|---|---|---|---|
| TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/$\overline{T2}$ | CP/$\overline{RL2}$ |

| Symbol | Position | Name and Significance |
|---|---|---|
| TF2 | T2CON.7 | Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1. |
| EXF2 | T2CON.6 | Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. |
| RCLK | T2CON.5 | Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock. |
| TCLK | T2CON.4 | Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock. |
| EXEN2 | T2CON.3 | Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX. |
| TR2 | T2CON.2 | Start/stop control for Timer 2. A logic 1 starts the timer. |
| C/$\overline{T2}$ | T2CON.1 | Timer or counter select. (Timer 2)<br>0 = Internal timer (OSC/12)<br>1 = External event counter (falling edge triggered). |
| CP/$\overline{RL2}$ | T2CON.0 | Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow. |

**Figure 11. T2CON: Timer/Counter 2 Control Register**

In the Capture Mode there are two options which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then Timer 2 is a 16-bit timer or counter which upon overflowing sets bit TF2, the Timer 2 overflow bit, which can be used to generate an interrupt. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in the Timer 2 registers, TL2 and TH2, to be captured into registers RCAP2L and RCAP2H, respectively. (RCAP2L and RCAP2H are new Special Function Registers in the 8052.) In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and EXF2, like TF2, can generate an interrupt.

The Capture Mode is illustrated in Figure 12.

In the auto-reload mode there are again two options, which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then when Timer 2 rolls over it not only sets TF2 but also causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2L and RCAP2H, which are preset by software. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX will also trigger the 16-bit reload and set EXF2.

The auto-reload mode is illustrated in Figure 13.

The baud rate generator mode is selected by RCLK = 1 and/or TCLK = 1. It will be described in conjunction with the serial port.

## SERIAL INTERFACE

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register. (However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost). The serial port receive and transmit registers are both accessed at Special Function Register SBUF. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

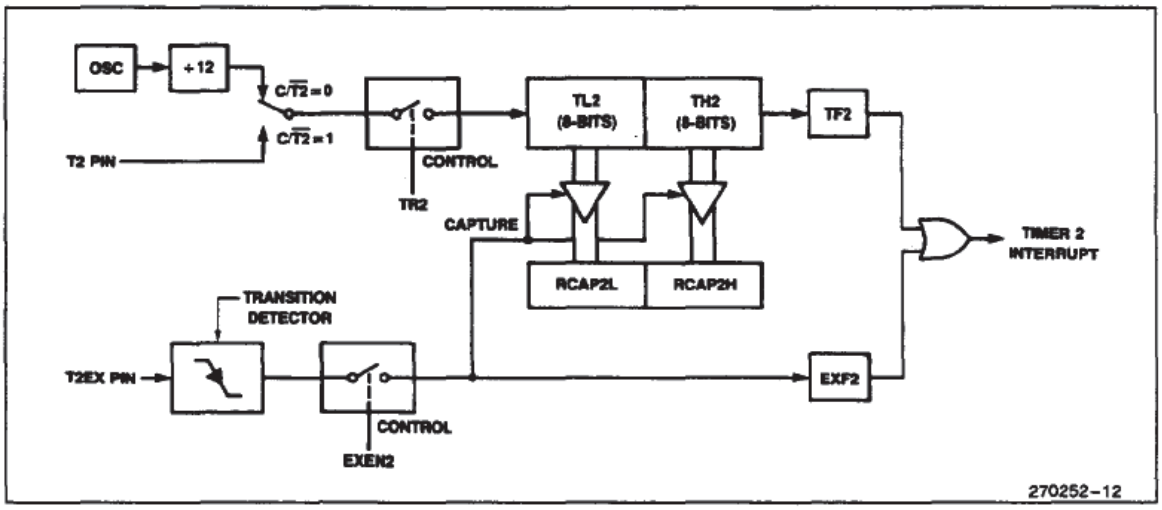



**Figure 12. Timer 2 in Capture Mode**

The serial port can operate in 4 modes:

**Mode 0:** Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at 1/12 the oscillator frequency.

**Mode 1:** 10 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in Special Function Register SCON. The baud rate is variable.

**Mode 2:** 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On Transmit, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On receive, the 9th data bit goes into RB8 in Special Functon Register SCON, while the stop bit is ignored. The baud rate is programmable to either $1/32$ or $1/64$ the oscillator frequency.

**Mode 3:** 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except the baud rate. The baud rate in Mode 3 is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1.

## Multiprocessor Communications

Modes 2 and 3 have a special provision for multiprocessor communications. In these modes, 9 data bits are received. The 9th one goes into RB8. Then comes a stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8 = 1. This feature is enabled by setting bit SM2 in SCON. A way to use this feature in multiprocessor systems is as follows.

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM2s set and go on about their business, ignoring the coming data bytes.

SM2 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. In a Mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

## Serial Port Control Register

The serial port control and status register is the Special Function Register SCON, shown in Figure 14. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8), and the serial port interrupt bits (TI and RI).

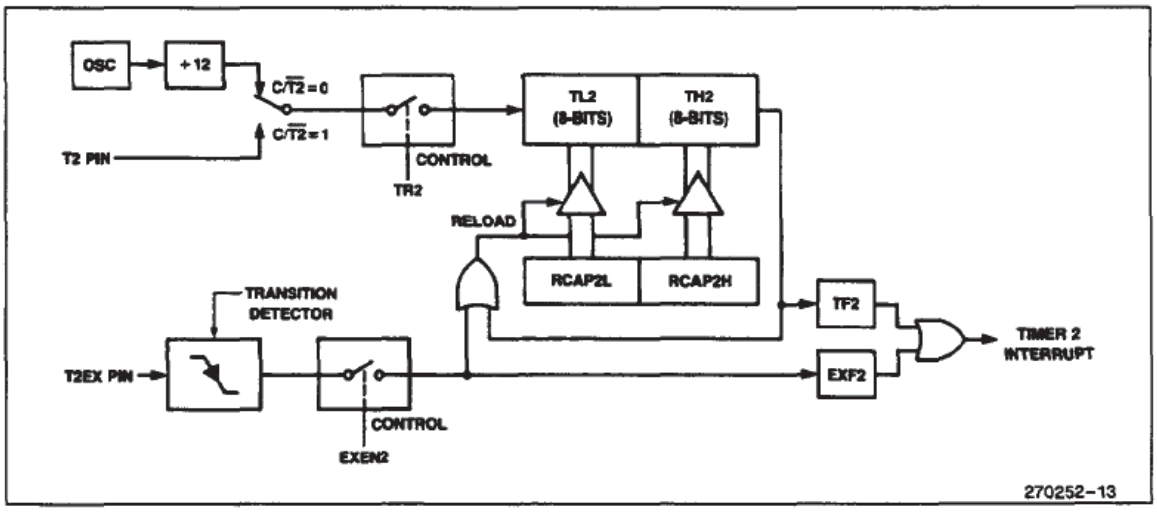



**Figure 13. Timer 2 in Auto-Reload Mode**

| (MSB) | | | | | | | (LSB) |
|---|---|---|---|---|---|---|---|
| SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |

Where SM0, SM1 specify the serial port mode, as follows:

| SM0 | SM1 | Mode | Description | Baud Rate |
|---|---|---|---|---|
| 0 | 0 | 0 | shift register | $f_{osc.}/12$ |
| 0 | 1 | 1 | 8-bit UART | variable |
| 1 | 0 | 2 | 9-bit UART | $f_{osc.}/64$ or $f_{osc.}/32$ |
| 1 | 1 | 3 | 9-bit UART | variable |

- SM2 enables the multiprocessor communication feature in Modes 2 and 3. In Mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In Mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In Mode 0, SM2 should be 0.
- REN enables serial reception. Set by software to enable reception. Clear by software to disable reception.
- TB8 is the 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.
- RB8 in Modes 2 and 3, is the 9th data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.
- TI is transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.
- RI is receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

**Figure 14. SCON: Serial Port Control Register**

## Baud Rates

The baud rate in Mode 0 is fixed:

$$\text{Mode 0 Baud Rate} = \frac{\text{Oscillator Frequency}}{12}$$

The baud rate in Mode 2 depends on the value of bit SMOD in Special Function Register PCON. If SMOD = 0 (which is the value on reset), the baud rate $1/64$ the oscillator frequency. If SMOD = 1, the baud rate is $1/32$ the oscillator frequency.

$$\text{Mode 2 Baud Rate} = \frac{2^{\text{SMOD}}}{64} \times (\text{Oscillator Frequency})$$

In the 8051, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate. In the 8052, these baud rates can be determined by Timer 1, or by Timer 2, or by both (one for transmit and the other for receive).

### Using Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

$$\text{Modes 1, 3 Baud Rate} = \frac{2^{SMOD}}{32} \times \text{(Timer 1 Overflow Rate)}$$

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either "timer" or "counter" operation, and in any of its 3 running modes. In the most typical applications, it is configured for "timer" operation, in the auto-reload mode (high nibble of TMOD = 0010B). In that case, the baud rate is given by the formula

$$\text{Modes 1, 3 Baud Rate} = \frac{2^{SMOD}}{32} \times \frac{\text{Oscillator Frequency}}{12x\,[256 - (TH1)]}$$

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the Timer 1 interrupt to do a 16-bit software reload.

Figure 15 lists various commonly used baud rates and how they can be obtained from Timer 1.

| Baud Rate | $f_{OSC}$ | SMOD | Timer 1 | | |
|---|---|---|---|---|---|
| | | | C/T | Mode | Reload Value |
| Mode 0 Max: 1 MHZ | 12 MHZ | X | X | X | X |
| Mode 2 Max: 375K | 12 MHZ | 1 | X | X | X |
| Modes 1, 3: 62.5K | 12 MHZ | 1 | 0 | 2 | FFH |
| 19.2K | 11.059 MHZ | 1 | 0 | 2 | FDH |
| 9.6K | 11.059 MHZ | 0 | 0 | 2 | FDH |
| 4.8K | 11.059 MHZ | 0 | 0 | 2 | FAH |
| 2.4K | 11.059 MHZ | 0 | 0 | 2 | F4H |
| 1.2K | 11.059 MHZ | 0 | 0 | 2 | E8H |
| 137.5 | 11.986 MHZ | 0 | 0 | 2 | 1DH |
| 110 | 6 MHZ | 0 | 0 | 2 | 72H |
| 110 | 12 MHZ | 0 | 0 | 1 | FEEBH |

**Figure 15. Timer 1 Generated Commonly Used Baud Rates**

### Using Timer 2 to Generate Baud Rates

In the 8052, Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Figure 11). Note then the baud rates for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 16.

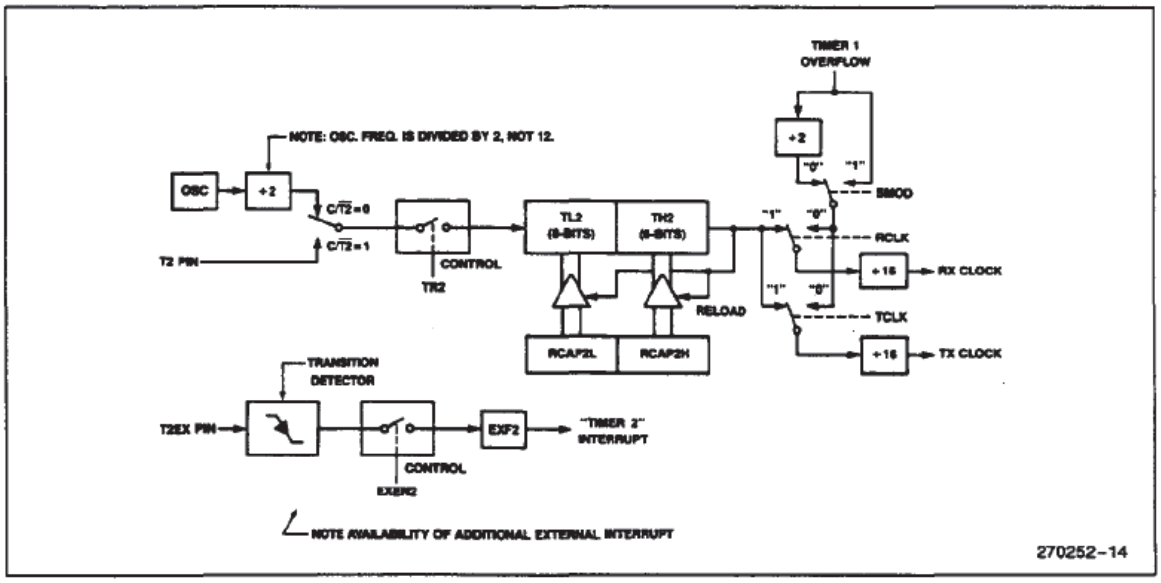



**Figure 16. Timer 2 in Baud Rate Generator Mode**

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

Now, the baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate as follows:

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either "timer" or "counter" operation. In the most typical applications, it is configured for "timer" operation (C/T2 = 0). "Timer" operation is a little different for Timer 2 when it's being used as a baud rate generator. Normally, as a timer it would increment every machine cycle (thus at $1/12$ the oscillator frequency). As a baud rate generator, however, it increments every state time (thus at $1/2$ the oscillator frequency). In that case the baud rate is given by the formula

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Oscillator Frequency}}{32x\,[65536 - (\text{RCAP2H, RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 16. This Figure is valid only if RCLK + TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Therefore, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

It should be noted that when Timer 2 is running (TR2 = 1) in "timer" function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the Timer is being incremented every state time, and the results of a read or write may not be accurate. The RCAP registers may be read, but shouldn't be written to, because a write might overlap a reload and cause write and/or reload errors. Turn the Timer off (clear TR2) before accessing the Timer 2 or RCAP registers, in this case.

## More About Mode 0

Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at $1/12$ the oscillator frequency.

Figure 17 shows a simplified functional diagram of the serial port in Mode 0, and associated timing.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal at S6P2 also loads a 1 into the 9th position of the transmit shift register and tells the TX Control block to commence a transmission. The internal timing is such that one full machine cycle will elapse between "write to SBUF," and activation of SEND.

SEND enables the output of the shift register to the alternate output function line of P3.0, and also enables SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK is low during S3, S4, and S5 of every machine cycle, and high during S6, S1 and S2. At S6P2 of every machine cycle in which SEND is active, the contents of the transmit shift register are shifted to the right one position.

As data bits shift out to the right, zeroes come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position, is just to the left of the MSB, and all positions to the left of that contain zeroes. This condition flags the TX Control block to do one last shift and then deactivate SEND and set TI. Both of these actions occur at S1P1 of the 10th machine cycle after "write to SBUF."

Reception is initiated by the condition REN = 1 and R1 = 0. At S6P2 of the next machine cycle, the RX Control unit writes the bits 11111110 to the receive shift register, and in the next clock phase activates RECEIVE.

RECEIVE enables SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK makes transitions at S3P1 and S6P1 of every machine cycle. At S6P2 of every machine cycle in which RECEIVE is active, the contents of the receive shift register are shifted to the left one position. The value that comes in from the right is the value that was sampled at the P3.0 pin at S5P2 of the same machine cycle.

As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX Control block to do one last shift and load SBUF. At S1P1 of the 10th machine cycle after the write to SCON that cleared RI, RECEIVE is cleared and RI is set.

## More About Mode 1

Ten bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in SCON. In the 8051 the baud rate is determined by the Timer 1 overflow rate. In the 8052 it is determined either by the Timer 1 overflow rate, or the Timer 2 overflow rate, or both (one for transmit and the other for receive).

Figure 18 shows a simplified functional diagram of the serial port in Mode 1, and associated timings for transmit receive.

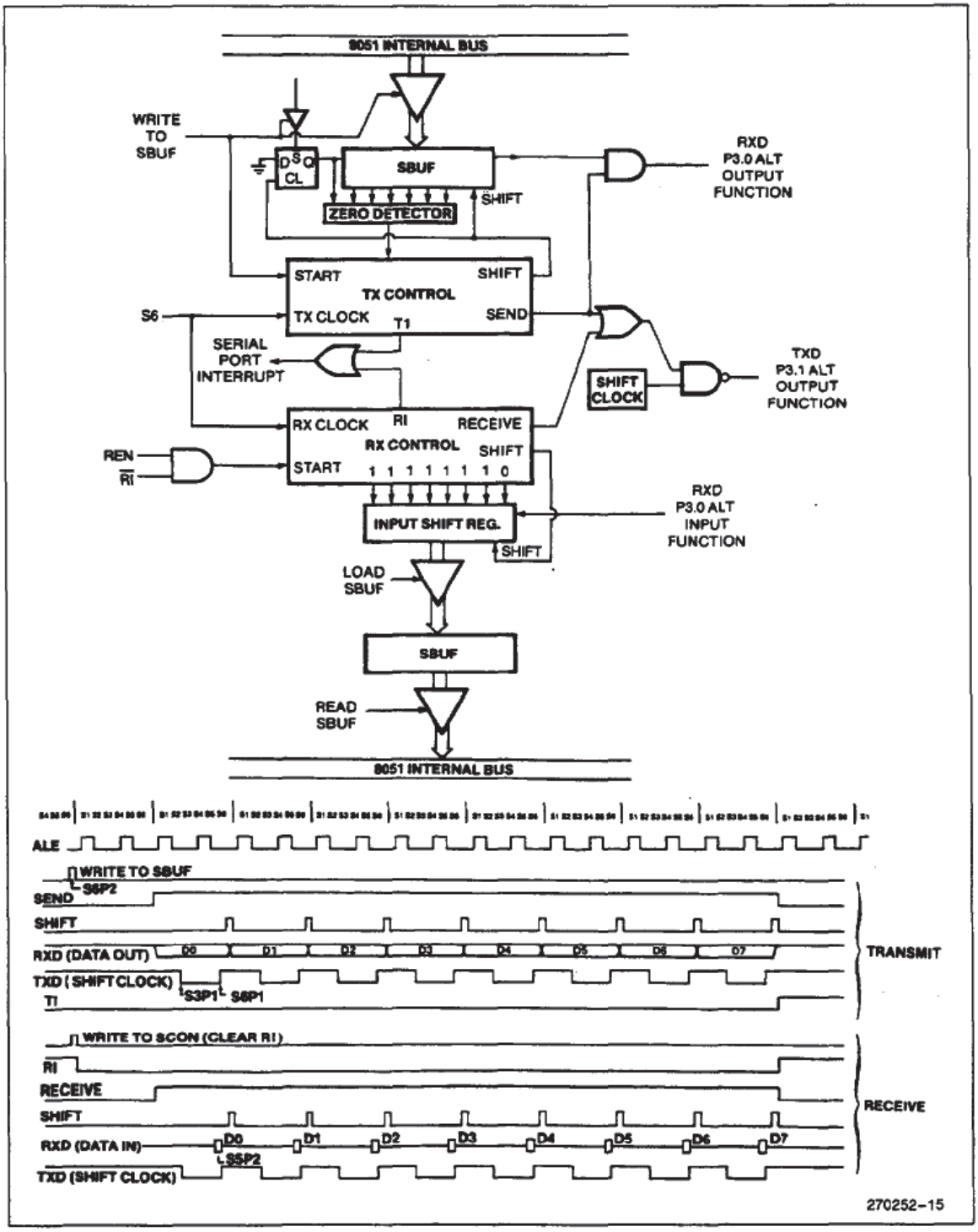



**Figure 17. Serial Port Mode 0**

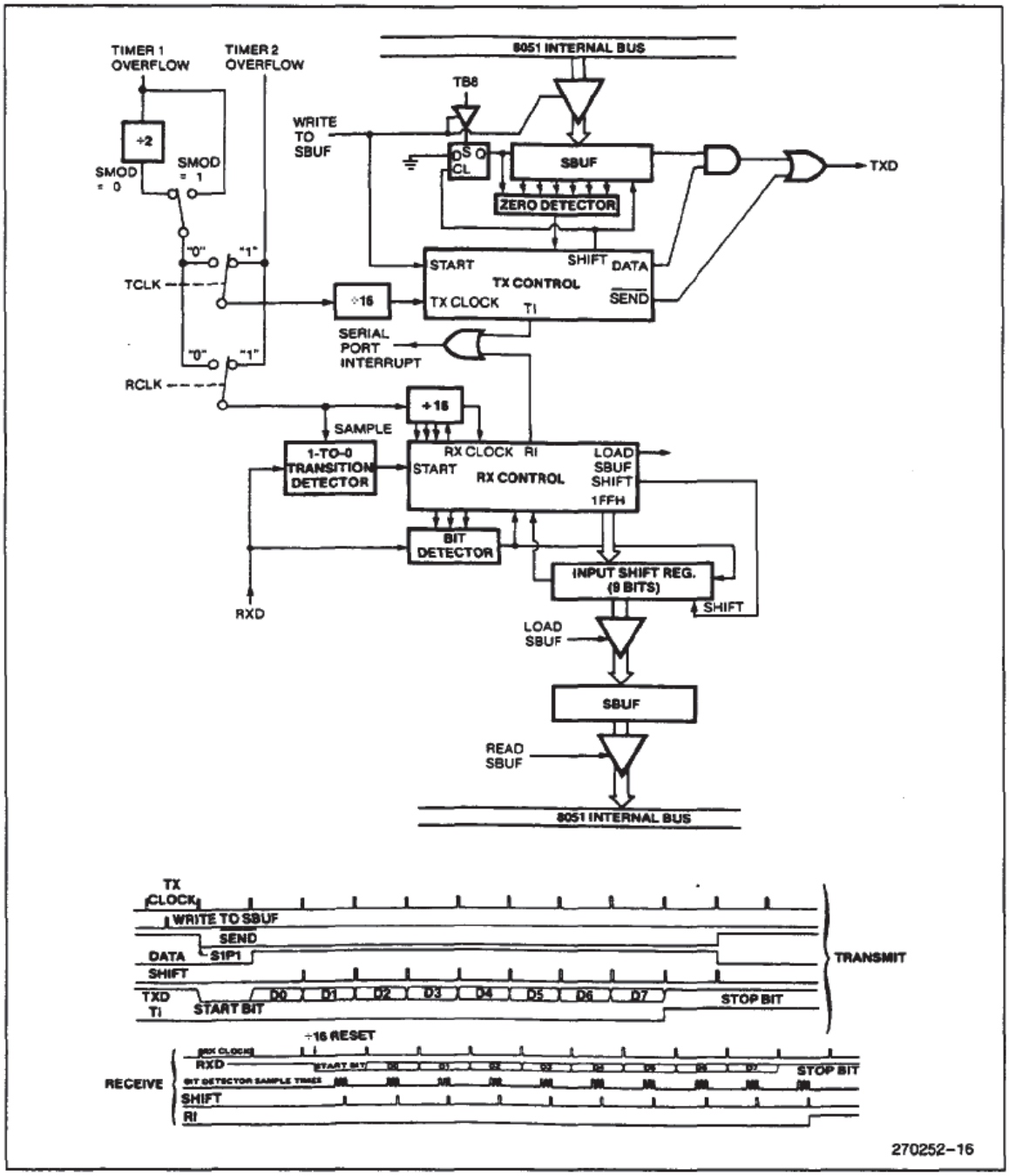



**Figure 18. Serial Port Mode 1. TCLK, RCLK and Timer 2 are Present in the 8052/8032 Only.**

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads a 1 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission actually commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal).

The transmission begins with activation of $\overline{\text{SEND}}$, which puts the start bit at TXD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that.

As data bits shift out to the right, zeroes are clocked in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position is just to the left of the MSB, and all positions to the left of that contain zeroes. This condition flags the TX Control unit to do one last shift and then deactivate $\overline{\text{SEND}}$ and set TI. This occurs at the 10th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written into the input shift register. Resetting the divide-by-16 counter aligns its rollovers with the boundaries of the incoming bit times.

The 16 states of the counter divide each bit time into 16ths. At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register, (which in mode 1 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated.

1) RI = 0, and
2) Either SM2 = 0, or the received stop bit = 1

If either of these two conditions is not met, the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB8, the 8 data bits go into SBUF, and RI is activated. At this time, whether the above conditions are met or not, the unit goes back to looking for a 1-to-0 transition in RXD.

## More About Modes 2 and 3

Eleven bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmit, the 9th data bit (TB8) can be assigned the value of 0 or 1. On receive, the 9th data bit goes into RB8 in SCON. The baud rate is programmable to either $1/32$ or $1/64$ the oscillator frequency in Mode 2. Mode 3 may have a variable baud rate generated from either Timer 1 or 2 depending on the state of TCLK and RCLK.

Figures 19 and 20 show a functional diagram of the serial port in Modes 2 and 3. The receive portion is exactly the same as in Mode 1. The transmit portion differs from Mode 1 only in the 9th bit of the transmit shift register.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads TB8 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal.)

The transmission begins with activation of SEND, which puts the start bit at TXD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeroes are clocked in. Thus, as data bits shift out to the right, zeroes are clocked in from the left. When TB8 is at the output position of the shift register, then the stop bit is just to the left of TB8, and all positions to the left of that contain zeroes. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI. This occurs at the 11th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written to the input shift register.

At the 7th, 8th and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

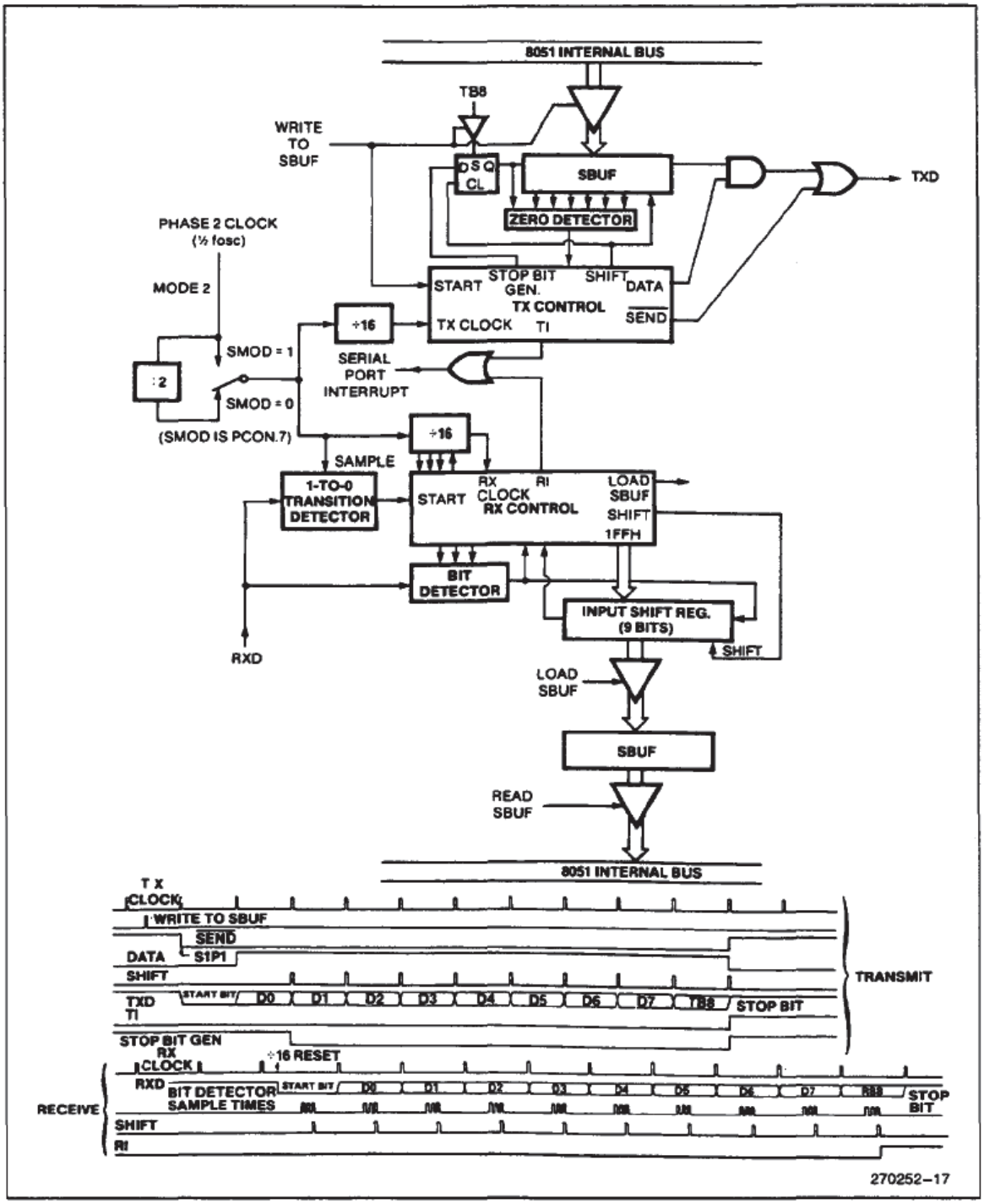



**Figure 19. Serial Port Mode 2**

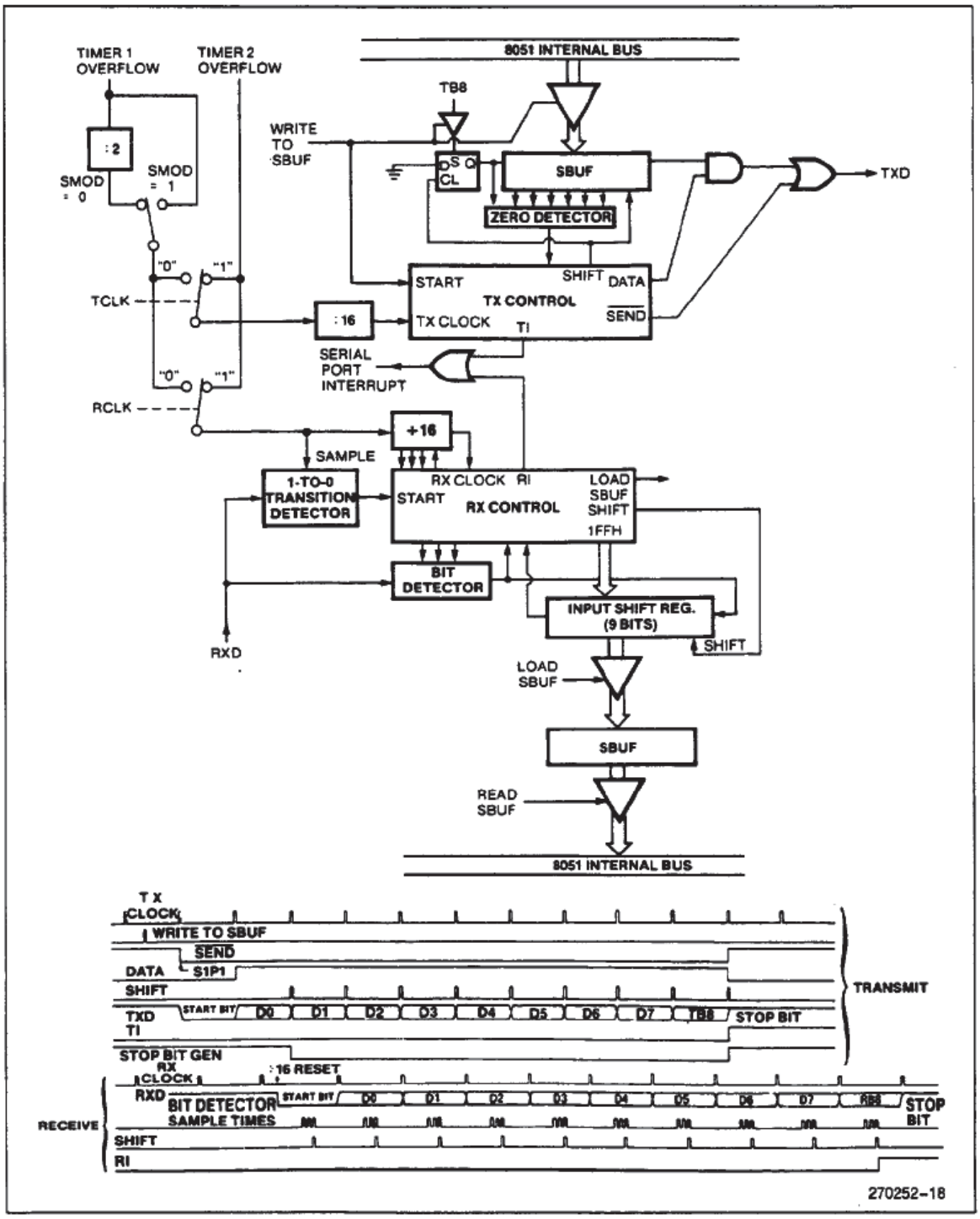



**Figure 20. Serial Port Mode 3. TCLK, RCLK, and Timer 2 are Present in the 8052/8032 Only.**

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in Modes 2 and 3 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

1) RI = 0, and
2) Either SM2 = 0 or the received 9th data bit = 1

If either of these conditions is not met, the received frame is irretrievably lost, and RI is not set. If both conditions are met, the received 9th data bit goes into RB8, and the first 8 data bits go into SBUF. One bit time later, whether the above conditions were met or not, the unit goes back to looking for a 1-to-0 transition at the RXD input.

Note that the value of the received stop bit is irrelevant to SBUF, RB8, or RI.

## INTERRUPTS

The 8051 provides 5 interrupt sources. The 8052 provides 6. These are shown in Figure 21.

The External Interrupts $\overline{INT0}$ and $\overline{INT1}$ can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in Register TCON. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt was transition-activated. If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

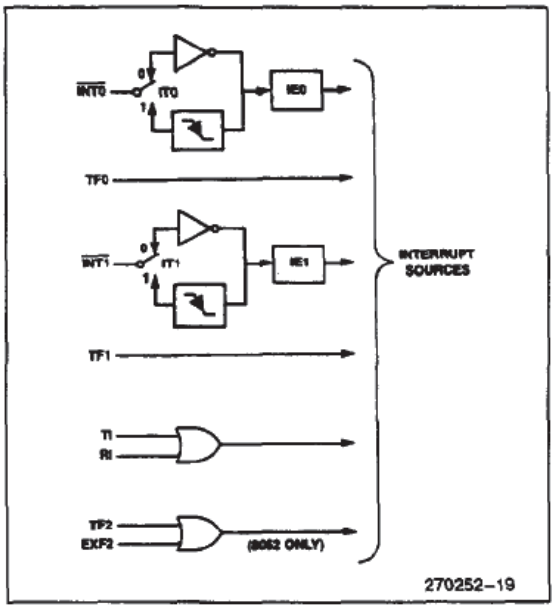



**Figure 21. MCS®-51 Interrupt Sources**

The Timer 0 and Timer 1 Interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter registers (except see Timer 0 in Mode 3). When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The Serial Port Interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine will normally have to determine whether it was RI or TI that generated the interrupt, and the bit will have to be cleared in software.

In the 8052, the Timer 2 Interrupt is generated by the logical OR of TF2 and EXF2. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared in software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

(MSB) (LSB)

| EA | — | ET2 | ES | ET1 | EX1 | ET0 | EX0 |
|---|---|---|---|---|---|---|---|

Enable Bit = 1 enables the interrupt.
Enable Bit = 0 disables it.

| Symbol | Position | Function |
|---|---|---|
| EA | IE.7 | disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit. |
| — | IE.6 | reserved. |
| ET2 | IE.5 | Timer 2 interrupt enable bit. |
| ES | IE.4 | Serial Port interrupt enable bit. |
| ET1 | IE.3 | Timer 1 interrupt enable bit. |
| EX1 | IE.2 | External interrupt 1 enable bit. |
| ET0 | IE.1 | Timer 0 interrupt enable bit. |
| EX0 | IE.0 | External interrupt 0 enable bit. |

User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

**Figure 22. IE: Interrupt Enable Register**

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE (Figure 22). IE contains also a global disable bit, EA, which disables all interrupts at once.

Note in Figure 22 that bit position IE.6 is unimplemented. In the 8051s, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future MCS-51 products.

## Priority Level Structure

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in Special Function Register IP (Figure 23). A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

(MSB) (LSB)

| — | — | PT2 | PS | PT1 | PX1 | PT0 | PX0 |
|---|---|---|---|---|---|---|---|

Priority bit = 1 assigns high priority.
Priority bit = 0 assigns low priority.

| Symbol | Position | Function |
|---|---|---|
| — | IP.7 | reserved |
| — | IP.6 | reserved |
| PT2 | IP.5 | Timer 2 interrupt priority bit. |
| PS | IP.4 | Serial Port interrupt priority bit. |
| PT1 | IP.3 | Timer 1 interrupt priority bit. |
| PX1 | IP.2 | External interrupt 1 priority bit. |
| PT0 | IP.1 | Timer 0 interrupt priority bit. |
| PX0 | IP.0 | External interrupt 0 priority bit. |

User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

**Figure 23. IP: Interrupt Priority Register**

If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence, as follows:

| | Source | Priority Within Level |
|---|---|---|
| 1. | IE0 | (highest) |
| 2. | TF0 | |
| 3. | IE1 | |
| 4. | TF1 | |
| 5. | RI + TI | |
| 6. | TF2 + EXF2 | (lowest) |

Note that the "priority within level" structure is only used to resolve *simultaneous requests of the same priority level.*

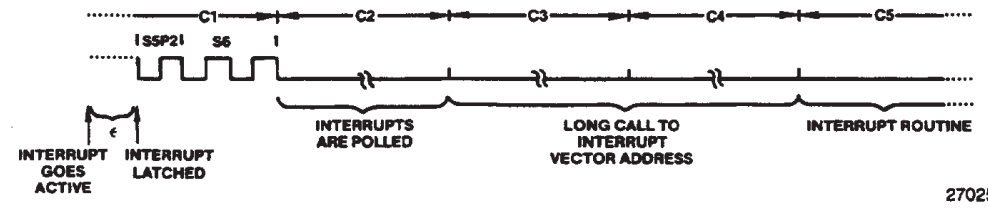
The IP register contains a number of unimplemented bits. IP.7 and IP.6 are vacant in the 8052s, and in the 8051s these and IP.5 are vacant. User software should not write 1s to these bit positions, since they may be used in future MCS-51 products.

## How Interrupts Are Handled

The interrupt flags are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. The 8052's Timer 2 interrupt cycle is different, as described in the Response Time Section. If one of the flags was in a set condition at S5P2 of the preceding 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 equal or higher priority level is already in progress.
2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE or IP registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be




This is the fastest possible response when C2 is the final cycle of an instruction other than RETI or an access to IE or IP.

**Figure 24. Interrupt Response Timing Diagram**

completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IE or IP, then at least *one more* instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at S5P2 of the previous machine cycle. Note then that if an interrupt flag is active but not being responded to for one of the above conditions, and is not *still* active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The polling cycle/LCALL sequence is illustrated in Figure 24.

Note that if an interrupt of higher priority level goes active prior to S5P2 of the machine cycle labeled C3 in Figure 24, then in accordance with the above rules it will be vectored to during C5 and C6, without any instruction of the lower priority routine having been executed.

Thus the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine. In some cases it also clears the flag that generated the interrupt, and in other cases it doesn't. It never clears the Serial Port or Timer 2 flags. This has to be done in the user's software. It clears an external interrupt flag (IE0 or IE1) only if it was transition-activated. The hardware-generated LCALL pushes the contents of the Program Counter onto 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 below.

| Source | Vector Address |
|---|---|
| IE0 | 0003H |
| TF0 | 000BH |
| IE1 | 0013H |
| TF1 | 001BH |
| RI + TI | 0023H |
| TF2 + EXF2 | 002BH |

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stack and reloads the Program Counter. Execution of the interrupted program continues from where it left off.

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking an interrupt was still in progress.

## External Interrupts

The external sources can be programmed to be level-activated or transition-activated by setting or clearing bit IT1 or IT0 in Register TCON. If ITx = 0, external interrupt x is triggered by a detected low at the $\overline{INTx}$ pin. If ITx = 1, external interrupt x is edge-triggered. In this mode if successive samples of the $\overline{INTx}$ pin show a high in one cycle and a low in the next cycle, interrupt request flag IEx in TCON is set. Flag bit IEx then requests the interrupt.

Since the external interrupt pins are sampled once each machine cycle, an input high or low should hold for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one machine cycle, and then hold it low for at least one machine cycle to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.

If the external interrupt is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

## Response Time

The $\overline{INT0}$ and $\overline{INT1}$ levels are inverted and latched into the interrupt flags IE0 and IE1 at S5P2 of every machine cycle. Similarly, the Timer 2 flag EXF2 and the Serial Port flags RI and TI are set at S5P2. The values are not actually polled by the circuitry until the next machine cycle.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag TF2 is set at S2P2 and is polled in the same cycle in which the timer overflows.

If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine. Figure 24 shows interrupt response timings.

A longer response time would result if the request is blocked by one of the 3 previously listed conditions. If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is not in its final cycle, the additional wait time cannot be more than 3 cycles, since the longest instructions (MUL and DIV) are only 4

cycles long, and if the instruction in progress is RETI or an access to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV).

Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 9 cycles.

## SINGLE-STEP OPERATION

The 8051 interrupt structure allows single-step execution with very little software overhead. As previously noted, an interrupt request will not be responded to while an interrupt of equal priority level is still in progress, nor will it be responded to after RETI until at least one other instruction has been executed. Thus, once an interrupt routine has been entered, it cannot be re-entered until at least one instruction of the interrupted program is executed. One way to use this feature for single-stop operation is to program one of the external interrupts (say, $\overline{INT0}$) to be level-activated. The service routine for the interrupt will terminate with the following code:

```
JNB   P3.2,$  ;Wait Here Till INT0 Goes High
JB    P3.2,$  ;Now Wait Here Till it Goes Low
RETI          :Go Back and Execute One Instruction
```

Now if the $\overline{INT0}$ pin, which is also the P3.2 pin, is held normally low, the CPU will go right into the External Interrupt 0 routine and stay there until $\overline{INT0}$ is pulsed (from low to high to low). Then it will execute RETI, go back to the task program, execute one instruction, and immediately re-enter the External Interrupt 0 routine to await the next pulsing of P3.2. One step of the task program is executed each time P3.2 is pulsed.

## RESET

The reset input is the RST pin, which is the input to a Schmitt Trigger.

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 generating an internal reset, with the timing shown in Figure 25.

The external reset signal is asynchronous to the internal clock. The RST pin is sampled during State 5 Phase 2 of every machine cycle. The port pins will maintain their current activities for 19 oscillator periods after a logic 1 has been sampled at the RST pin; that is, for 19 to 31 oscillator periods after the external reset signal has been applied to the RST pin.

While the RST pin is high, ALE and PSEN are weakly pulled high. After RST is pulled low, it will take 1 to 2 machine cycles for ALE and PSEN to start clocking. For this reason, other devices can not be synchronized to the internal timings of the 8051.

Driving the ALE and $\overline{PSEN}$ pins to 0 while reset is active could cause the device to go into an indeterminate state.

The internal reset algorithm writes 0s to all the SFRs except the port latches, the Stack Pointer, and SBUF. The port latches are initialized to FFH, the Stack Pointer to 07H, and SBUF is indeterminate. Table 3 lists the SFRs and their reset values.

The internal RAM is not affected by reset. On power up the RAM content is indeterminate.

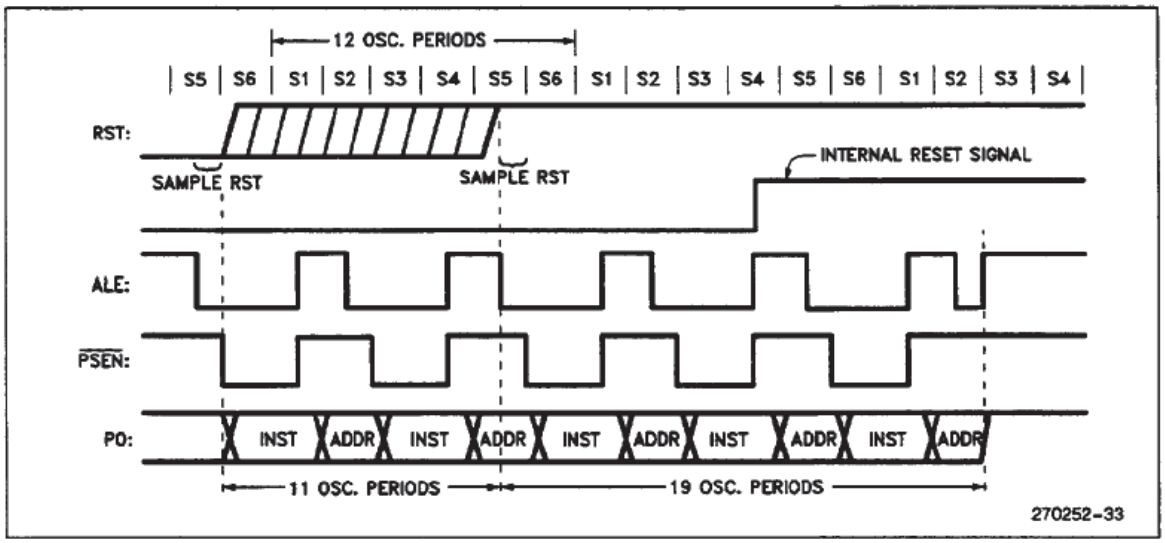



**Figure 25. Reset Timing**

**Table 3. Reset Values of the SFRs**

| SFR Name | Reset Value |
|---|---|
| PC | 0000H |
| ACC | 00H |
| B | 00H |
| PSW | 00H |
| SP | 07H |
| DPTR | 0000H |
| P0–P3 | FFH |
| IP (8051) | XXX00000B |
| IP (8052) | XX000000B |
| IE (8051) | 0XX00000B |
| IE (8052) | 0X000000B |
| TMOD | 00H |
| TCON | 00H |
| TH0 | 00H |
| TL0 | 00H |
| TH1 | 00H |
| TL1 | 00H |
| TH2 (8052) | 00H |
| TL2 (8052) | 00H |
| RCAP2H (8052) | 00H |
| RCAP2L (8052) | 00H |
| SCON | 00H |
| SBUF | Indeterminate |
| PCON (HMOS) | 0XXXXXXXB |
| PCON (CHMOS) | 0XXX0000B |

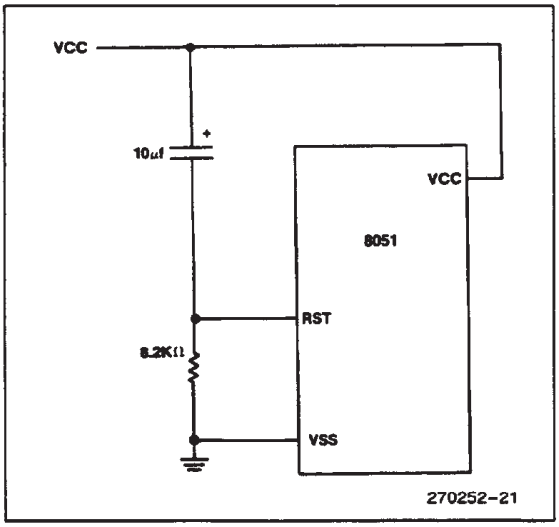



**Figure 26. Power on Reset Circuit**

## POWER-ON RESET

For HMOS devices when $V_{CC}$ is turned on an automatic reset can be obtained by connecting the RST pin to $V_{CC}$ through a 10 μF capacitor and to $V_{SS}$ through an 8.2 KΩ resistor (Figure 26). The CHMOS devices do not require this resistor although its presence does no harm. In fact, for CHMOS devices the external resistor can be removed because they have an internal pulldown on the RST pin. The capacitor value could then be reduced to 1 μF.

When power is turned on, the circuit holds the RST pin high for an amount of time that depends on the capacitor value and the rate at which it charges. To ensure a valid reset the RST pin must be held high long enough to allow the oscillator to start up plus two machine cycles.

On power up, $V_{CC}$ should rise within approximately ten milliseconds. The oscillator start-up time will depend on the oscillator frequency. For a 10 MHz crystal, the start-up time is typically 1 ms. For a 1 MHz crystal, the start-up time is typically 10 ms.

With the given circuit, reducing $V_{CC}$ quickly to 0 causes the RST pin voltage to momentarily fall below 0V. However, this voltage is internally limited and will not harm the device.

**NOTE:**
The port pins will be in a random state until the oscillator has started and the internal reset algorithm has written 1s to them.

Powering up the device without a valid reset could cause the CPU to start executing instructions from an indeterminate location. This is because the SFRs, specifically the Program Counter, may not get properly initialized.

## POWER-SAVING MODES OF OPERATION

For applications where power consumption is critical the CHMOS version provides power reduced modes of operation as a standard feature. The power down mode in HMOS devices is no longer a standard feature and is being phased out.

### CHMOS Power Reduction Modes

CHMOS versions have two power-reducing modes, Idle and Power Down. The input through which back-up power is supplied during these operations is VCC. Figure 27 shows the internal circuitry which implements these features. In the Idle mode (IDL = 1), the oscillator continues to run and the Interrupt, Serial Port, and Timer blocks continue to be clocked, but the

clock signal is gated off to the CPU. In Power Down (PD = 1), the oscillator is frozen. The Idle and Power Down modes are activated by setting bits in Special Function Register PCON. The address of this register is 87H. Figure 26 details its contents.

In the HMOS devices the PCON register only contains SMOD. The other four bits are implemented only in the CHMOS devices. User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

## IDLE MODE

An instruction that sets PCON.0 causes that to be the last instruction executed before going into the Idle mode. In the Idle mode, the internal clock signal is gated off to the CPU, but not to the Interrupt, Timer, and Serial Port functions. The CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical states they had at the time Idle was activated. ALE and $\overline{PSEN}$ hold at logic high levels.

There are two ways to terminate the Idle. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that put the device into Idle.

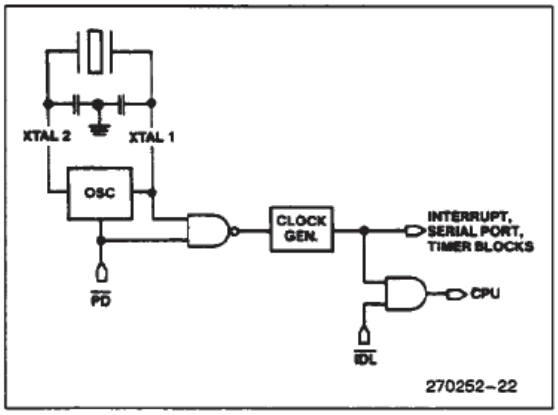



**Figure 27. Idle and Power Down Hardware**

| (MSB) | | | | | | | (LSB) |
|---|---|---|---|---|---|---|---|
| SMOD | - | - | - | GF1 | GF0 | PD | IDL |

| Symbol | Position | Name and Function |
|---|---|---|
| SMOD | PCON.7 | Double Baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the Serial Port is used in modes 1, 2, or 3. |
| — | PCON.6 | (Reserved) |
| — | PCON.5 | (Reserved) |
| — | PCON.4 | (Reserved) |
| GF1 | PCON.3 | General-purpose flag bit. |
| GF0 | PCON.2 | General-purpose flag bit. |
| PD | PCON.1 | Power Down bit. Setting this bit activates power down operation. |
| IDL | PCON.0 | Idle mode bit. Setting this bit activates idle mode operation. |

If 1s are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XXX0000).
In the HMOS devices the PCON register only contains SMOD. The other four bits are implemented only in the CHMOS devices. User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

**Figure 28. PCON: Power Control Register**

The flag bits GF0 and GF1 can be used to give an indication if an interrupt occurred during normal operation or during an Idle. For example, an instruction that activates Idle can also set one or both flag bits. When Idle is terminated by an interrupt, the interrupt service routine can examine the flag bits.

The other way of terminating the Idle mode is with a hardware reset. Since the clock oscillator is still running, the hardware reset needs to be held active for only two machine cycles (24 oscillator periods) to complete the reset.

The signal at the RST pin clears the IDL bit directly and asynchronously. At this time the CPU resumes program execution from where it left off; that is, at the instruction following the one that invoked the Idle Mode. As shown in Figure 25, two or three machine cycles of program execution may take place before the internal reset algorithm takes control. On-chip hardware inhibits access to the internal RAM during this time, but access to the port pins is not inhibited. To eliminate the possibility of unexpected outputs at the port pins, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external Data RAM.

## POWER DOWN MODE

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the Power Down mode. In the Power Down mode, the on-chip oscillator is stopped. With the clock frozen, all func-

**Table 4. EPROM Versions of the 8051 and 8052**

| Device Name | EPROM Version | EPROM Bytes | Ckt Type | VPP | Time Required to Program Entire Array |
|---|---|---|---|---|---|
| 8051AH | 8751H/8751BH | 4K | HMOS | 21.0V/12.75V | 4 minutes |
| 80C51BH | 87C51 | 4K | CHMOS | 12.75V | 13 seconds |
| 8052AH | 8752BH | 8K | HMOS | 12.75V | 26 seconds |

tions are stopped, but the on-chip RAM and Special Function Registers are held. The port pins output the values held by their respective SFRs. ALE and $\overline{PSEN}$ output lows.

The only exit from Power Down for the 80C51 is a hardware reset. Reset redefines all the SFRs, but does not change the on-chip RAM.

In the Power Down mode of operation, VCC can be reduced to as low as 2V. Care must be taken, however, to ensure that VCC is not reduced before the Power Down mode is invoked, and that VCC is restored to its normal operating level, before the Power Down mode is terminated. The reset that terminates Power Down also frees the oscillator. The reset should not be activated before VCC is restored to its normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize (normally less than 10 msec).

## EPROM VERSIONS

The EPROM versions of these devices are listed in Table 4. The 8751H programs at VPP = 21V using one 50 msec $\overline{PROG}$ pulse per byte programmed. This results in a total programming time (4K bytes) of approximately 4 minutes.

The 8751BH, 8752BH and 87C51 use the faster "Quick-Pulse" programming™ algorithm. These devices program at $\overline{VPP}$ = 12.75V using a series of twenty-five 100 $\mu$s $\overline{PROG}$ pulses per byte programmed. This results in a total programming time of approximately 26 seconds for the 8752BH (8 Kbytes) and 13 seconds for the 87C51 (4 Kbytes).

Detailed procedures for programming and verifying each device are given in the data sheets.

## Exposure to Light

It is good practice to cover the EPROM window with an opaque label when the device is in operation. This is not so much to protect the EPROM array from inadvertent erasure, but to protect the RAM and other on-chip logic. Allowing light to impinge on the silicon die while the device is operating can cause logical malfunction.

## Program Memory Locks

In some microcontroller applications it is desirable that the Program Memory be secure from software piracy. Intel has responded to this need by implementing a Program Memory locking scheme in some of the MCS-51 devices. While it is impossible for anyone to guarantee absolute security against all levels of technological sophistication, the Program Memory locks in the MCS-51 devices will present a substantial barrier against illegal readout of protected software.

### One Lock Bit Scheme on 8751H

The 8751H contains a lock bit which, once programmed, denies electrical access by any external means to the on-chip Program Memory. The effect of this lock bit is that while it is programmed the internal Program Memory can not be read out, the device can not be further programmed, and it *can not execute external Program Memory.* Erasing the EPROM array deactivates the lock bit and restores the device's full functionality. It can then be re-programmed.

The procedure for programming the lock bit is detailed in the 8751H data sheet.

### Two Program Memory Lock Schemes

The 8751BH, 8752BH and 87C51 contain two Program Memory locking schemes: Encrypted Verify and Lock Bits.

**Encryption Array:** Within the EPROM is an array of encryption bytes that are initially unprogrammed (all 1's). The user can program the array to encrypt the code bytes during EPROM verification. The verification procedure sequentially XNORs each code byte with one of the key bytes. When the last key byte in the array is reached, the verify routine starts over with the first byte of the array for the next code byte. If the key bytes are unprogrammed, the XNOR process leaves the code byte unchanged. With the key bytes programmed, the code bytes are encrypted and can be read correctly only if the key bytes are known in their proper order. Table 6 lists the number of encryption bytes available on the various products.

When using the encryption array, one important factor should be considered. If a code byte has the value

0FFH, verifying the byte will produce the encryption byte value. If a large block of code is left unprogrammed, a verification routine will display the encryption array contents. For this reason all unused code bytes should be programmed with some value other than 0FFH, and not all of them the same value. This will ensure maximum program protection.

**Program Lock Bits:** Also included in the Program Lock scheme are Lock Bits which can be enabled to provide varying degrees of protection. Table 5 lists the Lock Bits and their corresponding effect on the microcontroller. Refer to Table 6 for the Lock Bits available on the various products.

Erasing the EPROM also erases the Encryption Array and the Lock Bits, returning the part to full functionality.

**Table 5. Program Lock Bits and their Features**

| | Program Lock Bits | | | Protection Type |
|---|---|---|---|---|
| | LB1 | LB2 | LB3 | |
| 1 | U | U | U | No program lock features enabled. (Code verify will still be encrypted by the encryption array if programmed.) |
| 2 | P | U | U | MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the EPROM is disabled. |
| 3 | P | P | U | Same as 2, also verify is disabled. |
| 4 | P | P | P | Same as 3, also external execution is disabled. |

P-Programmed
U-Unprogrammed

Any other combination of the Lock Bits is not defined.

**Table 6. Program Protection**

| Device | Lock Bits | Encrypt Array |
|---|---|---|
| 8751BH | LB1, LB2 | 32 Bytes |
| 8752BH | LB1, LB2 | 32 Bytes |
| 87C51 | LB1, LB2, LB3 | 64 Bytes |

When Lock Bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

### ROM PROTECTION

The 8051AHP and 80C51BHP are ROM Protected versions of the 8051AH and 80C51BH, respectively. To incorporate this Protection Feature, program verification has been disabled and external memory accesses have been limited to 4K. Refer to the data sheets on these parts for more information.

## ONCE™ Mode

The ONCE ("on-circuit emulation") mode facilitates testing and debugging of systems using the device without the device having to be removed from the circuit. The ONCE mode is invoked by:

1. Pull ALE low while the device is in reset and $\overline{PSEN}$ is high;
2. Hold ALE low as RST is deactivated.

While the device is in ONCE mode, the Port 0 pins go into a float state, and the other port pins and ALE and $\overline{PSEN}$ are weakly pulled high. The oscillator circuit remains active. While the device is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored after a normal reset is applied.

## THE ON-CHIP OSCILLATORS

### HMOS Versions

The on-chip oscillator circuitry for the HMOS (HMOS-I and HMOS-II) members of the MCS-51 family is a single stage linear inverter (Figure 29), intended for use as a crystal-controlled, positive reactance oscillator (Figure 30). In this application the crystal is operated in its fundamental response mode as an inductive reactance in parallel resonance with capacitance external to the crystal.

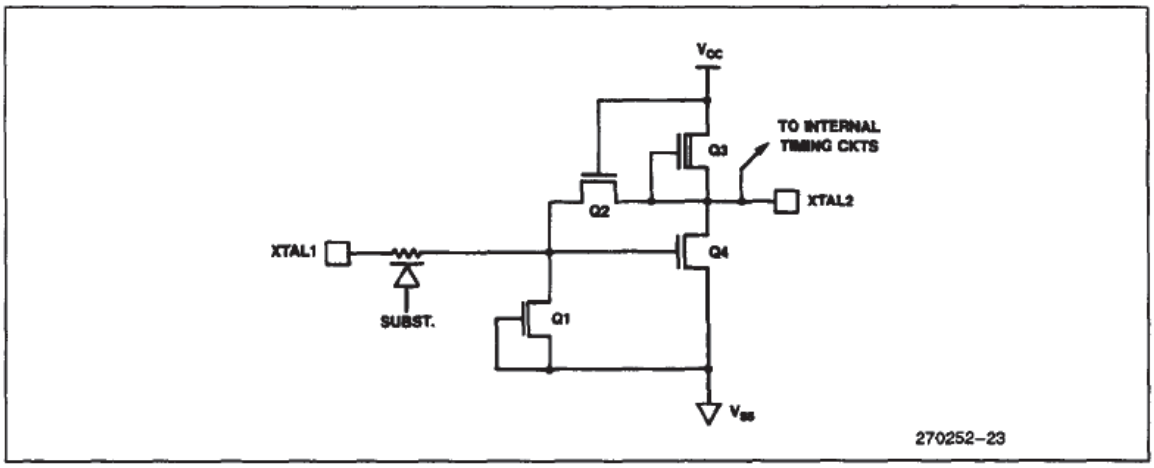



**Figure 29. On-Chip Oscillator Circuitry in the HMOS Versions of the MCS®-51 Family**

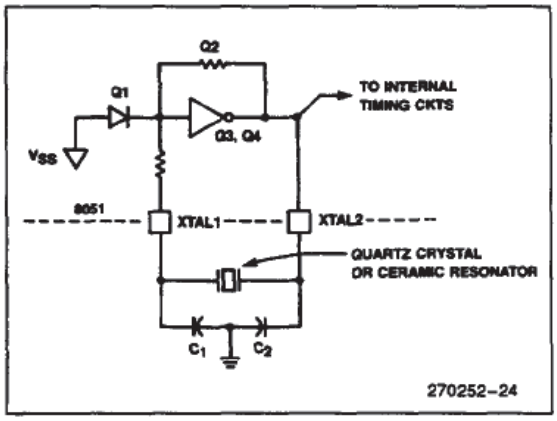



**Figure 30. Using the HMOS On-Chip Oscillator**

The crystal specifications and capacitance values (C1 and C2 in Figure 30) are not critical. 30 pF can be used in these positions at any frequency with good quality crystals. A ceramic resonator can be used in place of the crystal in cost-sensitive applications. When a ceramic resonator is used, C1 and C2 are normally selected to be of somewhat higher values, typically, 47 pF. The manufacturer of the ceramic resonator should be consulted for recommendations on the values of these capacitors.

In general, crystals used with these devices typically have the following specifications:

| | |
|---|---|
| ESR (Equivalent Series Resistance) | see Figure 31 |
| $C_O$ (Shunt Capacitance) | 7.0 pF max. |
| $C_L$ (Load Capacitance) | 30 pF ±3 pF |
| Drive Level | 1 mW |

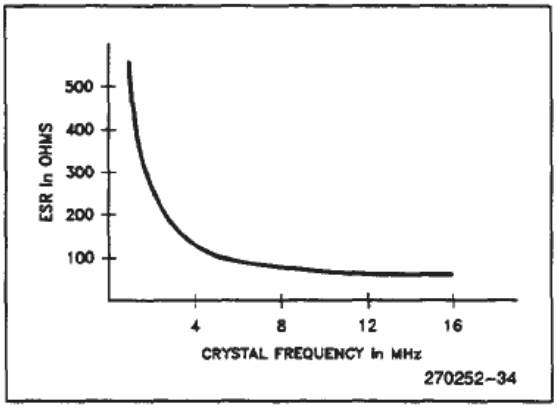



**Figure 31. ESR vs Frequency**

Frequency, tolerance and temperature range are determined by the system requirements.

A more in-depth discussion of crystal specifications, ceramic resonators, and the selection of values for C1 and C2 can be found in Application Note AP-155, "Oscillators for Microcontrollers," which is included in the *Embedded Applications Handbook*.

To drive the HMOS parts with an external clock source, apply the external clock signal to XTAL2, and ground XTAL1, as shown in Figure 32. A pullup resistor may be used (to increase noise margin), but is optional if VOH of the driving gate exceeds the VIH MIN specification of XTAL2.

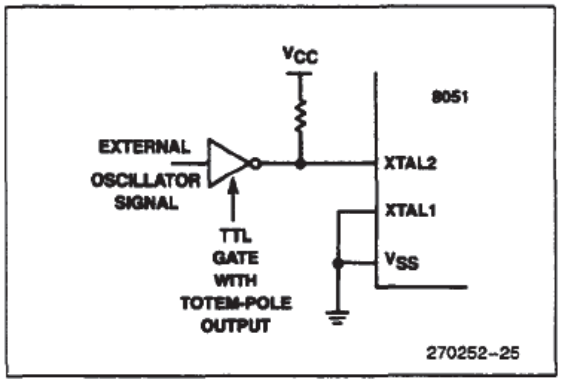



**Figure 32. Driving the HMOS MCS®-51 Parts with an External Clock Source**

## CHMOS Versions

The on-chip oscillator circuitry for the 80C51BH, shown in Figure 33, consists of a single stage linear inverter intended for use as a crystal-controlled, positive reactance oscillator in the same manner as the HMOS parts. However, there are some important differences.

One difference is that the 80C51BH is able to turn off its oscillator under software control (by writing a 1 to the PD bit in PCON). Another difference is that in the 80C51BH the internal clocking circuitry is driven by the signal at XTAL1, whereas in the HMOS versions it is by the signal at XTAL2.

The feedback resistor $R_f$ in Figure 33 consists of paralleled n- and p- channel FETs controlled by the PD bit, such that $R_f$ is opened when PD = 1. The diodes D1 and D2, which act as clamps to VCC and VSS, are parasitic to the $R_f$ FETs.

The oscillator can be used with the same external components as the HMOS versions, as shown in Figure 34. Typically, C1 = C2 = 30 pF when the feedback element is a quartz crystal, and C1 = C2 = 47 pF when a ceramic resonator is used.

To drive the CHMOS parts with an external clock source, apply the external clock signal to XTAL1, and leave XTAL2 float, as shown in Figure 35.

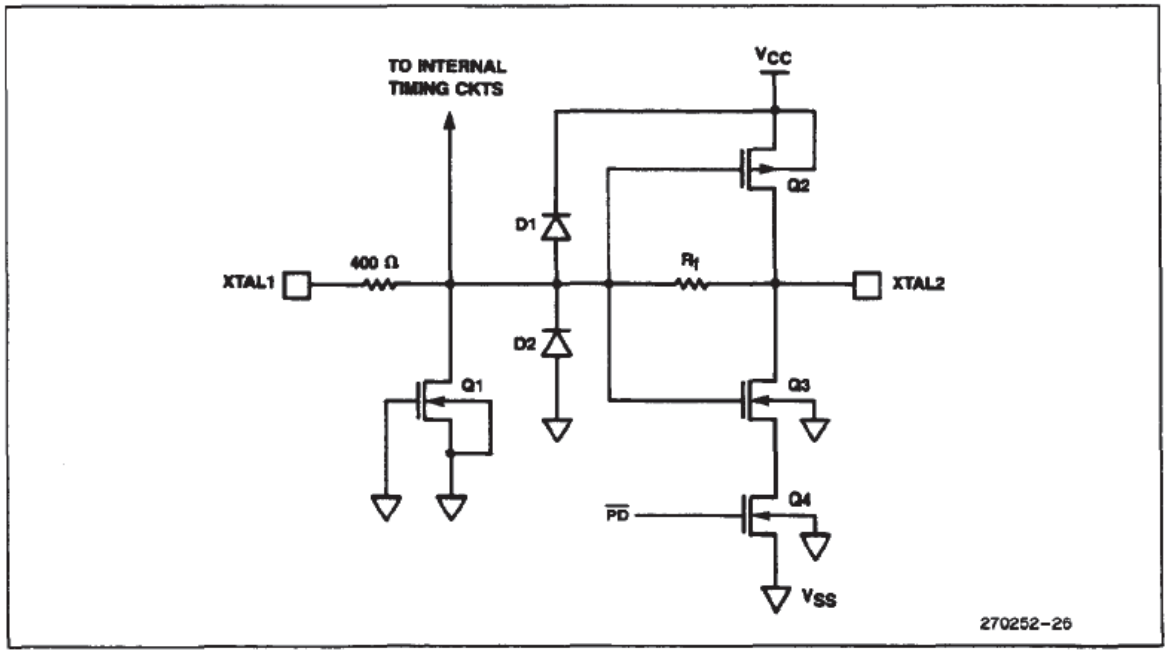



**Figure 33. On-Chip Oscillator Circuitry in the CHMOS Versions of the MCS®-51 Family**

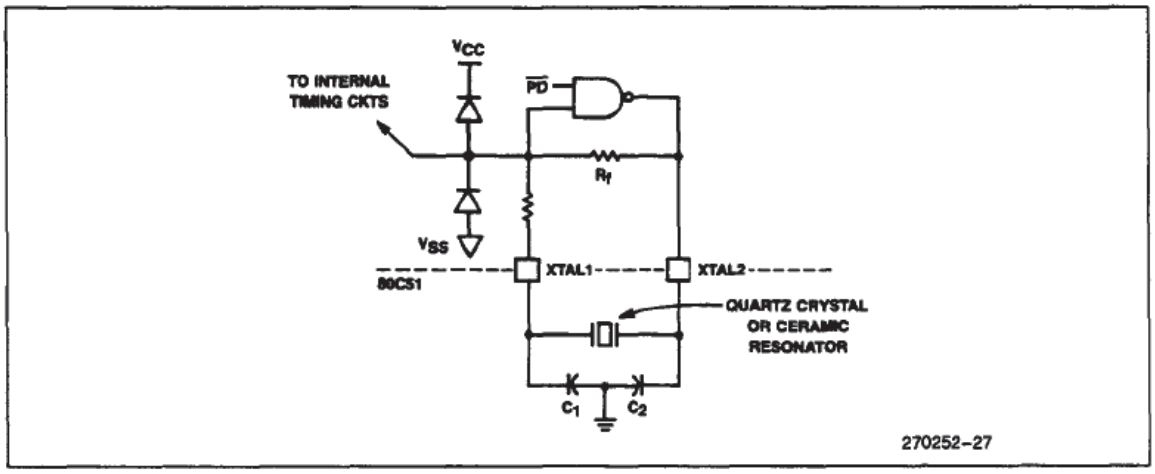



**Figure 34. Using the CHMOS On-Chip Oscillator**

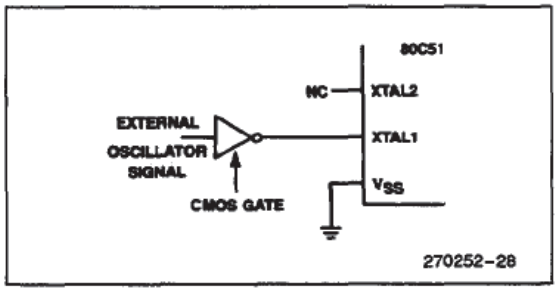



**Figure 35. Driving the CHMOS MCS®-51 Parts with an External Clock Source**

The reason for this change from the way the HMOS part is driven can be seen by comparing Figures 29 and 33. In the HMOS devices the internal timing circuits are driven by the signal at XTAL2. In the CHMOS devices the internal timing circuits are driven by the signal at XTAL1.

## INTERNAL TIMING

Figures 36 through 39 show when the various strobe and port signals are clocked internally. The figures do not show rise and fall times of the signals, nor do they show propagation delays between the XTAL signal and events at other pins.

Rise and fall times are dependent on the external loading that each pin must drive. They are often taken to be something in the neighborhood of 10 nsec, measured between 0.8V and 2.0V.

Propagation delays are different for different pins. For a given pin they vary with pin loading, temperature, VCC, and manufacturing lot. If the XTAL waveform is taken as the timing reference, prop delays may vary from 25 to 125 nsec.

The AC Timings section of the data sheets do not reference any timing to the XTAL waveform. Rather, they relate the critical edges of control and input signals to each other. The timings published in the data sheets include the effects of propagation delays under the specified test conditions.

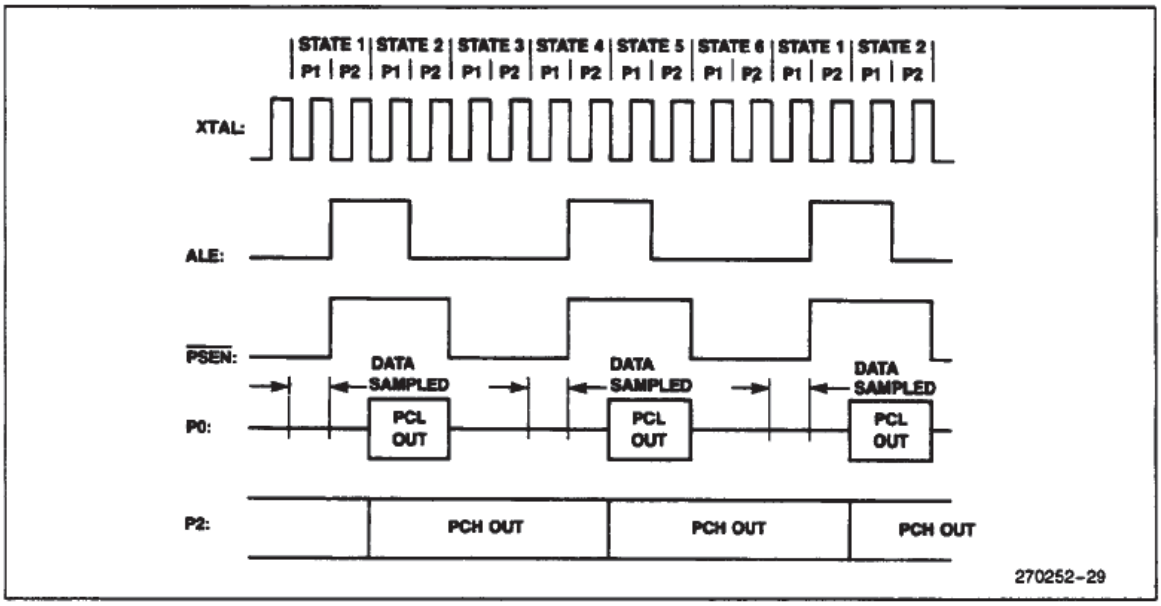



Figure 36. External Program Memory Fetches

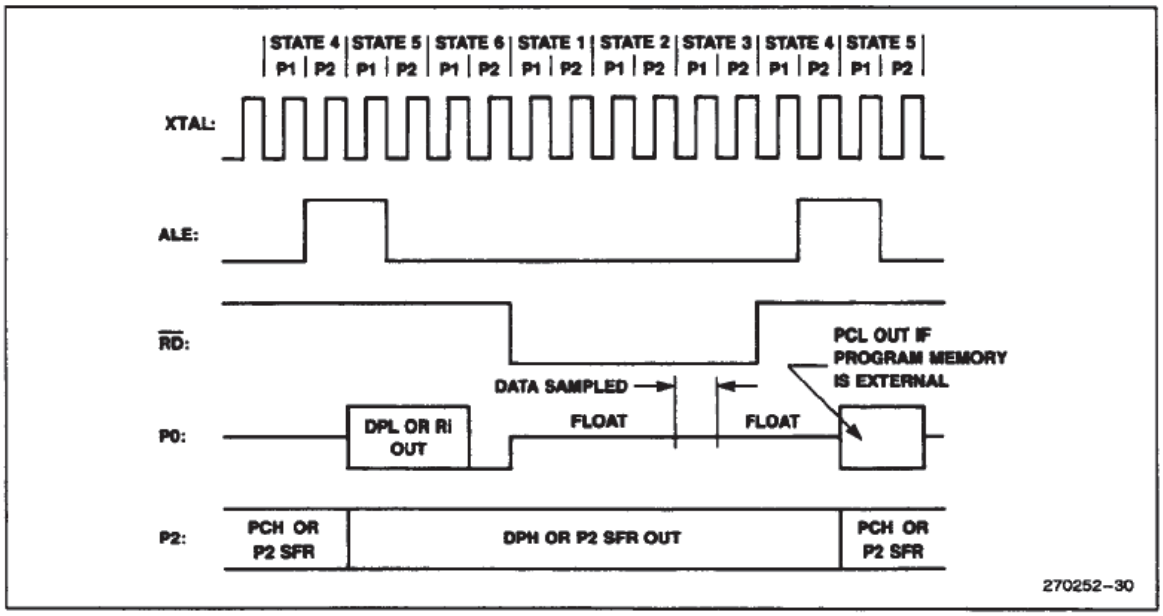



Figure 37. External Data Memory Read Cycle

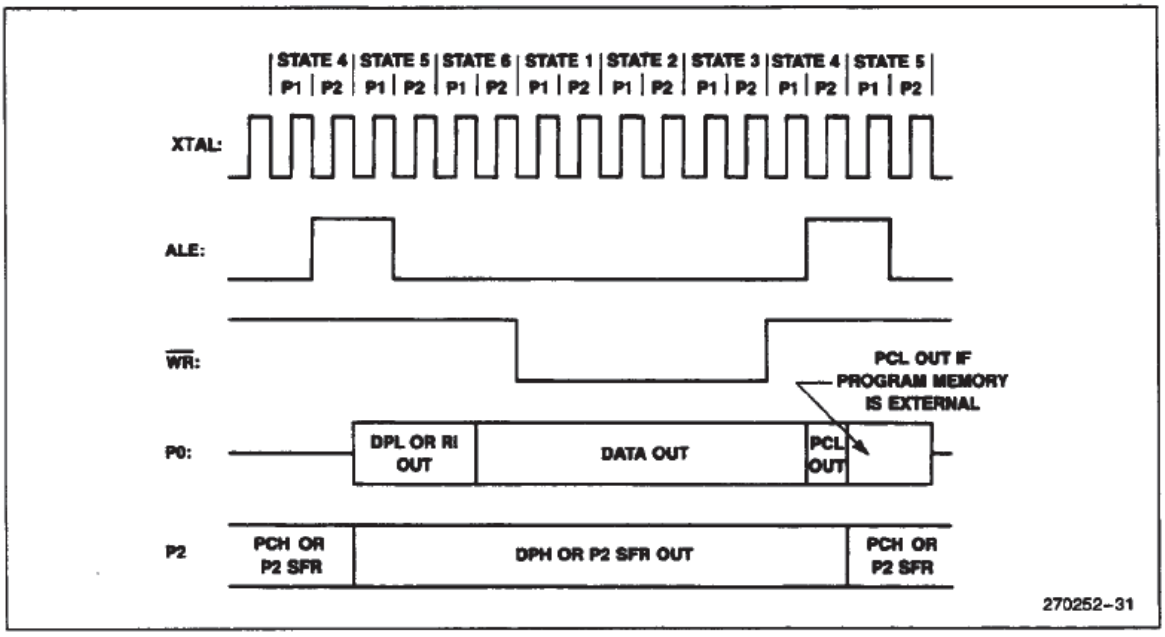



Figure 38. External Data Memory Write Cycle

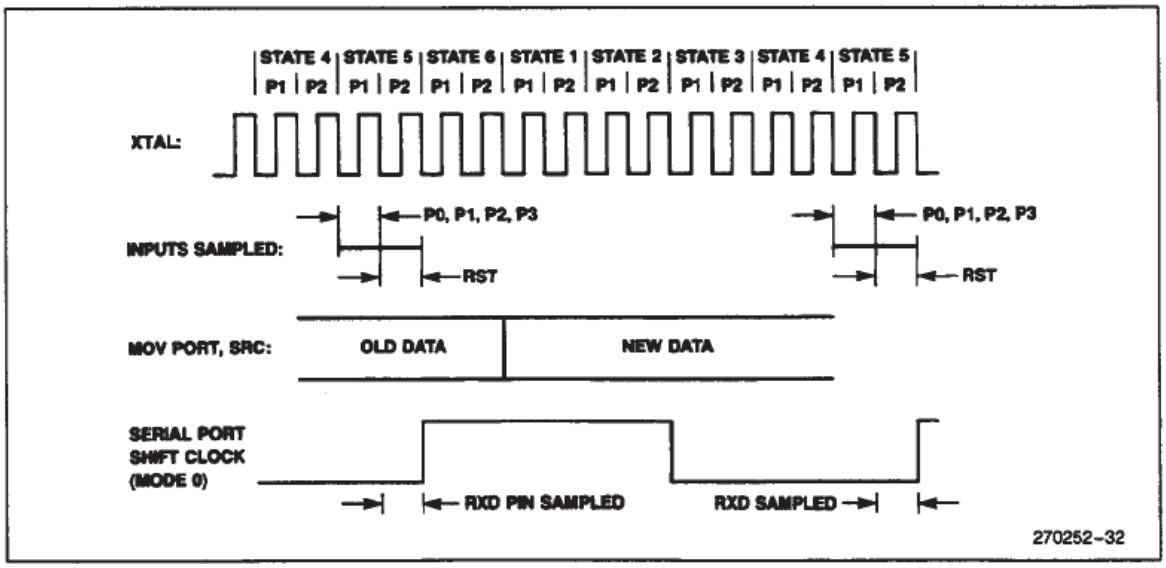



Figure 39. Port Operation

## ADDITIONAL REFERENCES

The following application notes and articles are found in the *Embedded Applications* handbook. (Order Number: 270648)

1. AP-125 "Designing Microcontroller Systems for Electrically Noisy Environments".
2. AP-155 "Oscillators for Microcontrollers".
3. AP-252 "Designing with the 80C51BH".
4. AR-517 "Using the 8051 Microcontroller with Resonant Transducers".

# 8XC52/54/58 Hardware Description

# 4

# 8XC52/54/58 HARDWARE DESCRIPTION

CONTENTS PAGE

# 8XC52/54/58 HARDWARE DESCRIPTION

## INTRODUCTION

The 8XC52/54/58 is a highly integrated 8-bit microcontroller based on the MCS®-51 architecture. The key features are an enhanced serial port for multi-processor communications and an up/down timer/counter. As this product is CHMOS, it has two software selectable reduced power modes: Idle Mode and Power Down Mode. Being a member of the MCS-51 family, the 8XC52/54/58 is optimized for control applications.

This document presents a comprehensive description of the on-chip hardware features of the 8XC52/54/58 as they differ from the 80C51BH. It begins by describing how the I/O functions are different and then discusses each of the peripherals as follows:

- 256 Bytes On-Chip RAM
- Special Function Registers (SFR)
- Timer 2
  - — Capture Timer/Counter
  - — Up/Down Timer/Counter
  - — Baud Rate Generator
- Full-Duplex Programmable Serial Interface with
  - — Framing Error Detection
  - — Automatic Address Recognition
- 6 Interrupt Sources
- Enhanced Power Down Mode
- Power Off Flag
- ONCE Mode

The 8XC52/54/58 uses the standard 8051 instruction set and is pin-for-pin compatible with the existing MCS-51 family of products. Table 1 summarizes the product names and memory differences of the various 8XC52/54/58 products currently available. Throughout this document, the products will generally be referred to as the 8XC5X.

**Table 1. 8XC52/54/58 Microcontrollers**

| ROM Device | EPROM Version | ROMless Version | ROM/EPROM Bytes | RAM Bytes |
|---|---|---|---|---|
| 80C52 | 87C52 | 80C32 | 8K | 256 |
| 80C54 | 87C54 | 80C32 | 16K | 256 |
| 80C58 | 87C58 | 80C32 | 32K | 256 |

For a description of the features that are the same as the 80C51, the reader should refer to the MCS-51 Architectural Overview, MCS-51 Programmers Guide/ Instruction Set, and the Hardware Description of the 80C51 in the Embedded Microcontrollers and Processors Handbook (Order #270645).

## PIN DESCRIPTION

The 8XC5X pin-out is the same as the 80C51. The only difference is the alternate function of pins P1.0 and P1.1. P1.0 is the external clock input for Timer 2. P1.1 is the Reload/Capture/Direction Control for Timer 2.

## DATA MEMORY

The 8XC5X implements 256 bytes of on-chip RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. That means they have the same addresses, but they are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the CPU knows whether the access is to the upper 128 bytes of RAM or the SFR space by the addressing mode used in the instruction. Instructions that use direct addressing access SFR space. For example,

```
MOV 0A0H, #data (Direct Addressing)
```

accesses the SFR at location 0A0H (which is P2). Instructions that use indirect addressing access the upper 128 bytes of RAM. For example,

```
MOV @R0, #data (Indirect Addressing)
```

where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H). Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

## SPECIAL FUNCTION REGISTERS

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 2.

Note that not all of the addresses are occupied. Unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

User software should not write 1s to these unlisted locations, since they may be used in future MCS-51 products to invoke new features. In that case the reset or inactive values of the new bits will always be 0.



Order Number: 270783-004

**Table 2. 8XC5X SFR Map and Reset Values**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0F8H | | | | | | | | | 0FFH |
| 0F0H | B<br>00000000 | | | | | | | | 0F7H |
| 0E8H | | | | | | | | | 0EFH |
| 0E0H | ACC<br>00000000 | | | | | | | | 0E7H |
| 0D8H | | | | | | | | | 0DFH |
| 0D0H | PSW<br>00000000 | | | | | | | | 0D7H |
| 0C8H | T2CON<br>00000000 | T2MOD<br>XXXXXX00 | RCAP2L<br>00000000 | RCAP2H<br>00000000 | TL2<br>00000000 | TH2<br>00000000 | | | 0CFH |
| 0C0H | | | | | | | | | 0C7H |
| 0B8H | IP<br>X0000000 | SADEN<br>00000000 | | | | | | | 0BFH |
| 0B0H | P3<br>11111111 | | | | | | | IPH<br>X0000000 | 0B7H |
| 0A8H | IE<br>00000000 | SADDR<br>00000000 | | | | | | | 0AFH |
| 0A0H | P2<br>11111111 | | | | | | | | 0A7H |
| 98H | SCON<br>00000000 | SBUF<br>XXXXXXXX | | | | | | | 9FH |
| 90H | P1<br>11111111 | | | | | | | | 97H |
| 88H | TCON<br>00000000 | TMOD<br>00000000 | TL0<br>00000000 | TL1<br>00000000 | TH0<br>00000000 | TH1<br>00000000 | | | 8FH |
| 80H | P0<br>11111111 | SP<br>00000111 | DPL<br>00000000 | DPH<br>00000000 | | | | PCON<br>00000000 | 87H |

**Timer Registers**—Control and status bits are contained in registers T2CON and T2MOD for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

**Serial Port Registers**—Registers SADDR and SADEN are used to define the Given and the Broadcast addresses for the Automatic Address Recognition feature.

**Interrupt Registers**—The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the 6 interrupt sources in the IP register. The IPH register allows four priorities.

## TIMER 2

Timer 2 is a 16-bit Timer/Counter which can operate either as a timer or an event counter. This is selectable by bit C/$\overline{T2}$ in the SFR T2CON (Table 3). It has three

operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON as shown in Table 4.

Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Thus one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is $1/12$ of the oscillator frequency.

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is $1/24$ of the oscillator frequency. To ensure that a given level is sampled at least once before it changes, it should be held for at least one full machine cycle.

**Table 3. T2CON—Timer/Counter 2 Control Register**

T2CON Address = 0C8H — Reset Value = 0000 0000B

Bit Addressable

| | TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | CP/$\overline{T2}$ | CP/$\overline{RL2}$ |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| TF2 | Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1. |
| EXF2 | Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1). |
| RCLK | Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock. |
| TCLK | Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock. |
| EXEN2 | Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX. |
| TR2 | Start/Stop control for Timer 2. TR2 = 1 starts the timer. |
| C/$\overline{T2}$ | Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered). |
| CP/$\overline{RL2}$ | Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow. |

**Table 4. Timer 2 Operating Modes**

| RCLK + TCLK | CP/$\overline{RL2}$ | TR2 | MODE |
|---|---|---|---|
| 0 | 0 | 1 | 16-Bit Auto-Reload |
| 0 | 1 | 1 | 16-Bit Capture |
| 1 | X | 1 | Baud Rate Generator |
| X | X | 0 | (Off) |

## CAPTURE MODE

In the capture mode there are two options selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16-bit timer or counter which upon overflow sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in TH2 and TL2 to be captured into RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set. The EXF2 bit, like TF2, can generate an interrupt. The capture mode is illustrated in Figure 1.

## AUTO-RELOAD (Up or Down Counter)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by a bit named DCEN (Down Counter Enable) located in the SFR T2MOD (see Table 5). Upon reset the DCEN bit is set to 0 so that Timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down depending on the value of the T2EX pin.

Figure 2 shows Timer 2 automatically counting up when DCEN = 0. In this mode there are two options selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in RCAP2H and RCAP2L are preset by software. If EXEN2 = 1, a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX. This transition also sets the EXF2 bit. Both the TF2 and EXF2 bits can generate an interrupt if enabled.

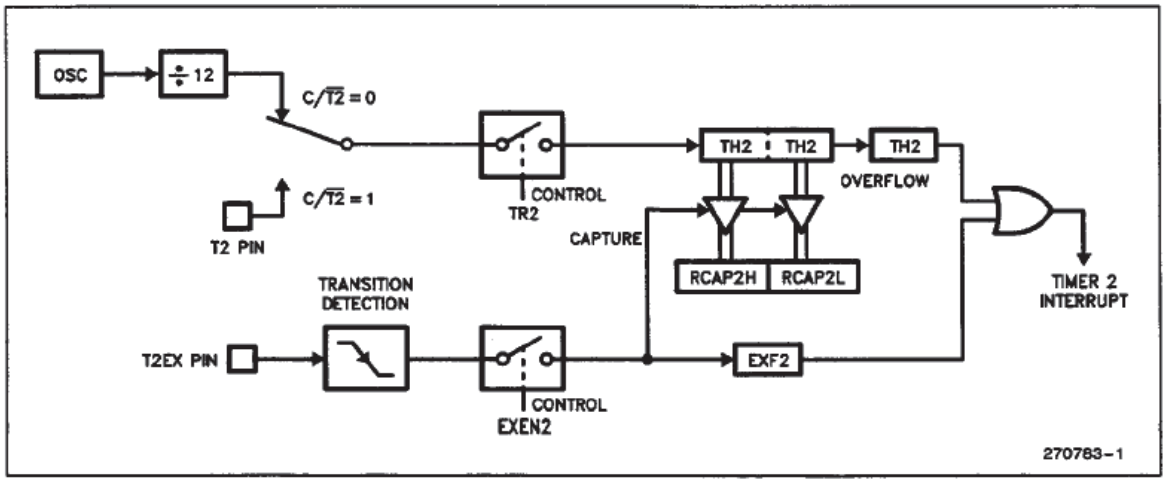



**Figure 1. Timer 2 in Capture Mode**

**Table 5. T2MOD—Timer 2 Mode Control Register**

T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

| — | — | — | — | — | — | T20E | DCEN |
|---|---|---|---|---|---|---|---|
| Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| — | Not implemented, reserved for future use. |
| T20E | Timer 2 Output Enable bit. |
| DCEN | When set, this bit allows Timer 2 to be configured as an up/down counter. |

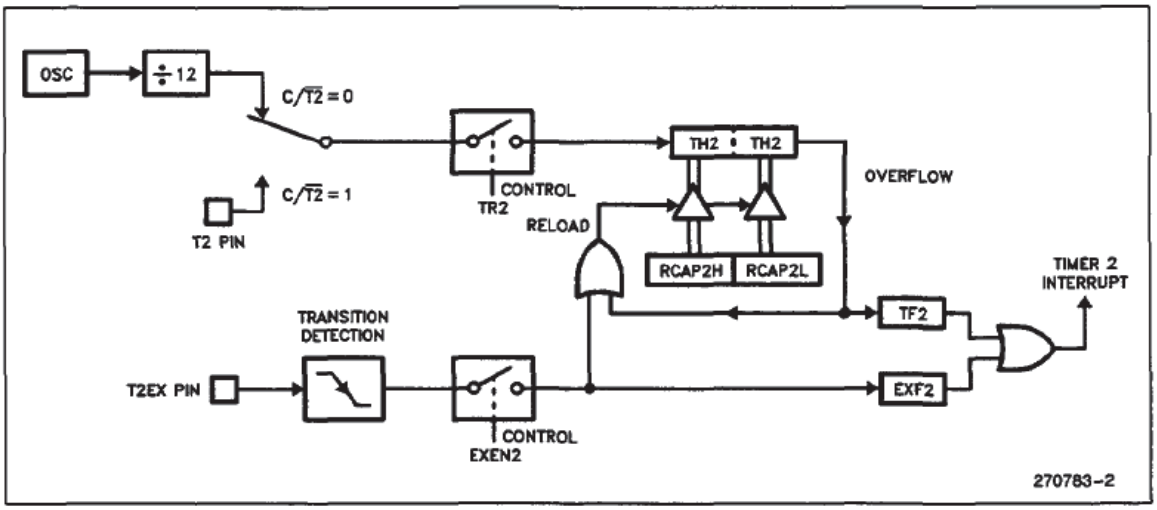



**Figure 2. Timer 2 Auto Reload Mode (DCEN = 0)**

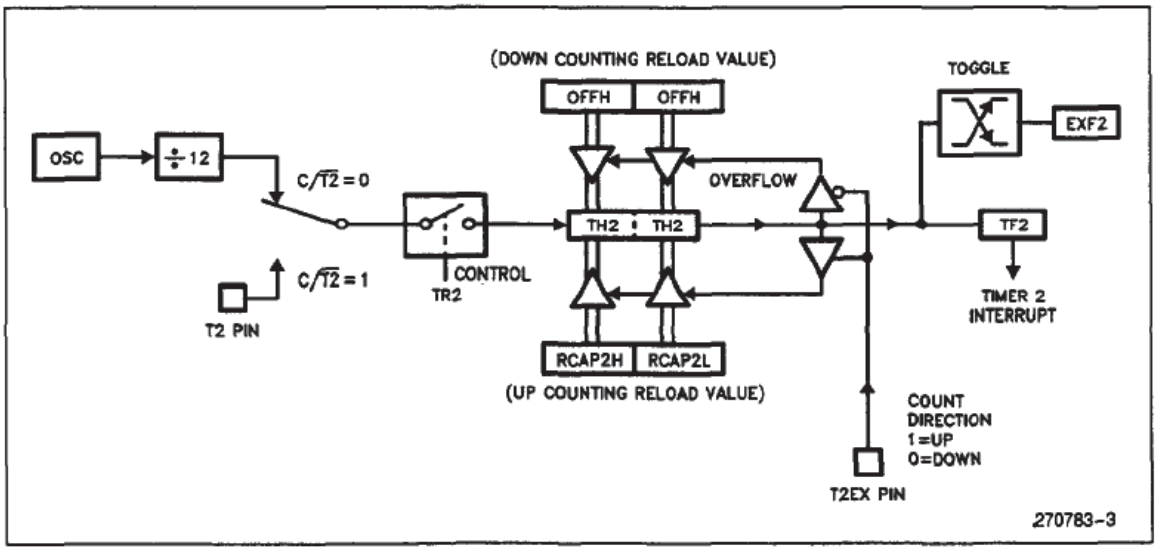



**Figure 3. Timer 2 Auto Reload Mode (DCEN = 1)**

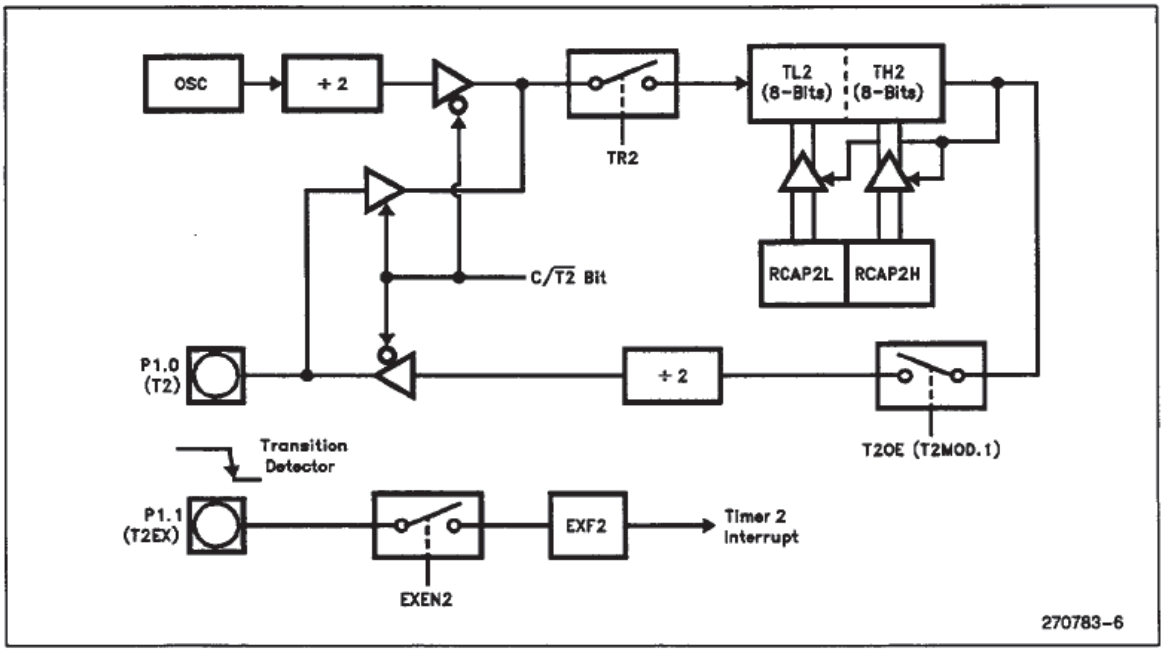



**Figure 4. Timer 2 in Clock-Out Mode**

Setting the DCEN bit enables Timer 2 to count up or down as shown in Figure 3. In this mode the T2EX pin controls the direction of count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit. This overflow also causes the 16-bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

A logic 0 at T2EX makes Timer 2 count down. Now the timer underflows when TH2 and TL2 equal the values stored in RCAP2H and RCAP2L. The underflow sets the TF2 bit and causes 0FFFFH to be reloaded into the timer registers.

The EXF2 bit toggles whenever Timer 2 overflows or underflows. This bit can be used as a 17th bit of resolution if desired. In this operating mode, EXF2 does not flag an interrupt.

## BAUD RATE GENERATOR

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Table 3). Note that the baud rates for transmit and receive can be different. This is accomplished by using Timer 2 for the receiver or transmitter and using Timer 1 for the other function. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 5.

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

The baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate as follows:

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either "timer" or "counter" operation. In most applications, it is configured for "timer" operation ($C/\overline{T2} = 0$). The "timer" operation is different for Timer 2 when it's being used as a baud rate generator. Normally, as a timer, it increments every machine cycle (thus at $1/12$ the oscillator frequency). As a baud rate generator, however, it increments every state time (thus at $1/2$ the oscillator frequency). The baud rate formula is given below:

$$\text{Modes 1 and 3 Baud Rate} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - (\text{RCAP2H, RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 5. This figure is valid only if RCLK or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

It should be noted that when Timer 2 is running (TR2 = 1) in "timer" function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the Timer is being incremented every state time, and the results of a read or write may not be accurate. The RCAP2 registers may be read, but shouldn't be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

## PROGRAMMABLE CLOCK OUT

A 50% duty cycle clock can be programmed to come out on P1.0. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed (1) to input the external clock for Timer/Counter 2 or (2) to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz at a 16 MHz operating frequency.

To configure the Timer/Counter 2 as a clock generator, bit C/T2 (T2CON.1) must be cleared and bit T2OE (T2MOD.1) must be set. Bit TR2 (T2CON.2) starts and stops the timer.

The Clock-Out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, TCAP2L) as shown in this equation:

$$\text{Clock-Out Frequency} = \frac{\text{Oscillator Frequency}}{4 \times (65536 - \text{RCAP2H, RCAP2L})}$$

In the clock-out mode, Timer 2 roll-overs will not generate an interrupt. This is similar to when Timer 2 is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and clock-out frequencies can not be determined independently from one another since they both use RCAP2H and RCAP2L.

## UART

The UART in the 8XC5X operates identically to the UART in the 80C51 except for the following enhancements. For a complete understanding of the 8XC5X UART please refer to the description in the 80C51 Hardware Description chapter in the Embedded Microcontrollers and Processors Handbook.

**Framing Error Detection**—Framing Error Detection allows the serial port to check for valid stop bits in modes 1, 2 or 3. A missing stop bit can be caused, for example, by noise on the serial lines, or transmission by two CPUs simultaneously.

If a stop bit is missing a Framing Error bit (FE) is set. The FE bit can be checked in software after each reception to detect communication errors. Once set, the FE bit must be cleared in software. A valid stop bit will not clear FE.

The FE bit is located in SCON and shares the same bit address as SM0. Control bit SMOD0 in the PCON register (location PCON.6) determines whether the SM0 or FE bit is accessed. If SMOD0 = 0, then accesses to SCON.7 are to SM0. If SMOD0 = 1, then accesses to SCON.7 are to FE.

**Automatic Address Recognition**—Automatic Address Recognition reduces the CPU time required to service the serial port. Since the CPU is only interrupted when it receives its own address, the software overhead to compare addresses is eliminated. With this feature enabled in one of the 9-bit modes, the Receive Interrupt (RI) flag will only get set when the received byte corresponds to either a Given or Broadcast address.

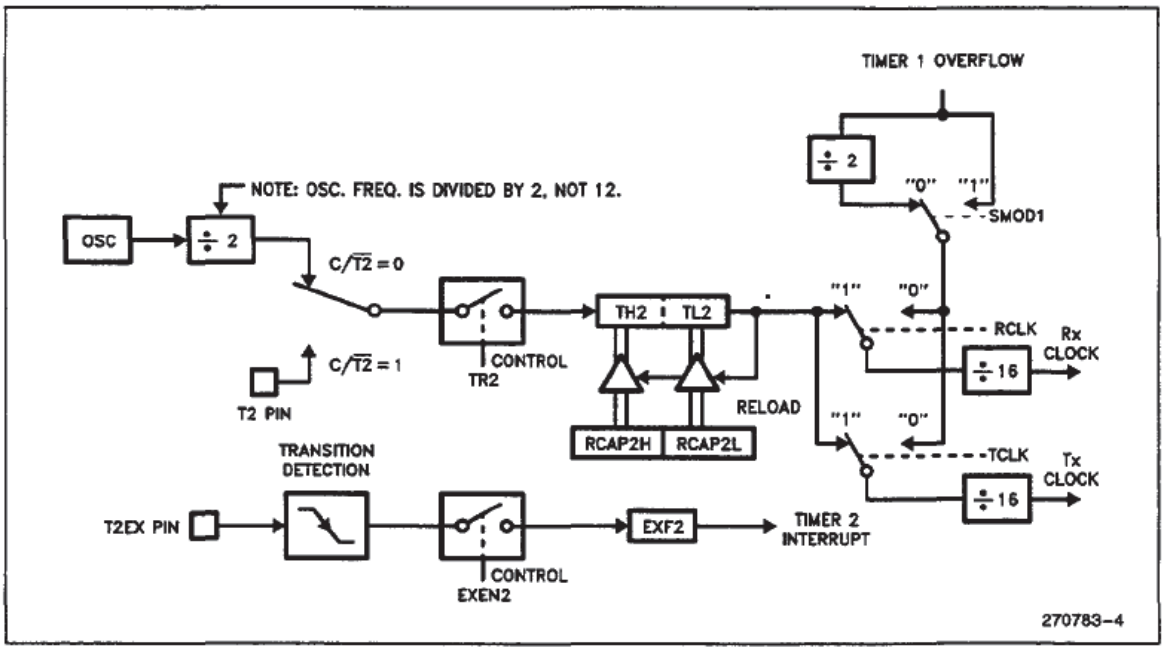



**Figure 5. Timer 2 in Baud Rate Generator Mode**

A way to use this feature in multiprocessor systems is as follows:

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. Remember, an address byte has its 9th bit set to 1, whereas a data byte has its 9th bit set to 0. All the slave processors should have their SM2 bits set to 1 so they will only be interrupted by an address byte. The Automatic Address Recognition feature allows only the addressed slave to be interrupted. In this mode, the address comparison occurs in hardware, not software. (On the 80C51 serial port, an address byte interrupts all slaves for an address comparison).

The addressed slave then clears its SM2 bit and prepares to receive the data bytes that will be coming. The other slaves are unaffected by these data bytes as they are still waiting to receive an address byte.

The feature works the same way in the 8-bit mode (Mode 1) as in the 9-bit modes, except that the stop bit takes the place of the 9th data bit. If SM2 is set, the RI flag is set only if the received byte matches the Given or Broadcast Address and is terminated by a valid stop bit. Setting the SM2 bit has no effect on Mode 0.

The master can selectively communicate with groups of slaves by using the Given Address. Addressing all slaves at once is possible with the Broadcast Address. These addresses are defined for each slave by two Special Function Registers: SADDR and SADEN.

A slave's individual address is specified in SADDR. SADEN is a mask byte that defines don't-care bits to form the Given Address. These don't-cares allow flexibility in the user-defined protocol to address one or more slaves at a time. The following is an example of how the user could define Given Addresses to selectively address different slaves.

**Slave 1:**

| | | |
|---|---|---|
| SADDR | = | 1111 0001 |
| SADEN | = | 1111 1010 |
| GIVEN | = | 1111 0X0X |

**Slave 2:**

| | | |
|---|---|---|
| SADDR | = | 1111 0011 |
| SADEN | = | 1111 1001 |
| GIVEN | = | 1111 0XX1 |

The SADEN bits are selected such that each slave can be addressed separately. Notice that bit 0 (LSB) is a don't-care for Slave 1's Given Address, but bit 0 = 1 for Slave 2. Thus, to selectively communicate with just Slave 1 the master must send an address with bit 0 = 0 (e.g., 1111 0000).

Similarly, bit 1 = 0 for Slave 1, but is a don't-care for Slave 2. Now to communicate with just Slave 2 an address with bit 1 = 1 must be used (e.g., 1111 0111).

Finally, for a master to communicate with both slaves at once the address must have bit 0 = 1 and bit 1 = 0. Notice, however, that bit 2 is a don't-care for both slaves. This allows two different addresses to select both slaves (1111 0001 or 1111 0101). If a third slave was added that required its bit 2 = 0, then the latter address could be used to communicate with Slave 1 and 2 but not Slave 3.

The master can also communicate with all slaves at once with the Broadcast Address. It is formed from the logical OR of the SADDR and SADEN registers with zeroes defined as don't-cares. The don't-cares also allow flexibility in defining the Broadcast Address, but in most applications a Broadcast Address will be 0FFH.

SADDR and SADEN are located at address 0A9H and 0B9H, respectively. On reset, the SADDR and SADEN registers are initialized to 00H which defines the Given and Broadcast Addresses as XXXX XXXX (all don't-cares). This assures the 8XC5X serial port to be backwards compatible with other MCS®-51 products which do not implement automatic address recognition.

## INTERRUPTS

The 8XC5X has a total of 6 interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1 and 2) and the serial port interrupt. These interrupts are all shown in Figure 6.

Timer 2 Interrupt is generated by the logical OR of bits TF2 and EXF2 in register T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt and that bit will have to be cleared in software.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag, TF2 is set at S2P2 and is polled in the same cycle in which the timer overflows.

## Interrupt Priority Structure

A second Interrupt Priority register (IPH) has been added, increasing the number of priority levels to four. Table 6 shows this second register. The added register becomes the MSB of the priority select bits and the existing IP register acts as the LSB. This scheme maintains compatibility with the rest of the MCS-51 family. Table 7 shows the bit values and priority levels associated with each combination.

**Table 6. IPH: Interrupt Priority High Register**

IPH Address = 0B7H — Reset Value = X000 0000

| — | PPCH | PT2H | PSH | PT1H | PX1H | PT0H | PX0H |
|---|---|---|---|---|---|---|---|
| Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| — | Not Implemented, reserved for future use. |
| PPCH | PCA interrupt priority high bit. |
| PT2H | Timer 2 interrupt priority high bit. |
| PSH | Serial Port interrupt priority high bit. |
| PT1H | Timer 1 interrupt priority high bit. |
| PX1H | External interrupt 1 priority high bit. |
| PT0H | Timer 0 interrupt priority high bit. |
| PX0H | External interrupt priority high bit. |

**Table 7. Priority Level Bit Values**

| Priority Bits | | Interrupt Priority Level |
|---|---|---|
| IPH.x | IP.x | |
| 0 | 0 | Level 0 (Lowest) |
| 0 | 1 | Level 1 |
| 1 | 0 | Level 2 |
| 1 | 1 | Level 3 (Highest) |

## POWER DOWN MODE

The 8XC5X can exit Power Down with either a hardware reset or external interrupt. Reset redefines all the SFRs but does not change the on-chip RAM. An external interrupt allows both the SFRs (except PD in PCON) and the on-chip RAM to retain their values.

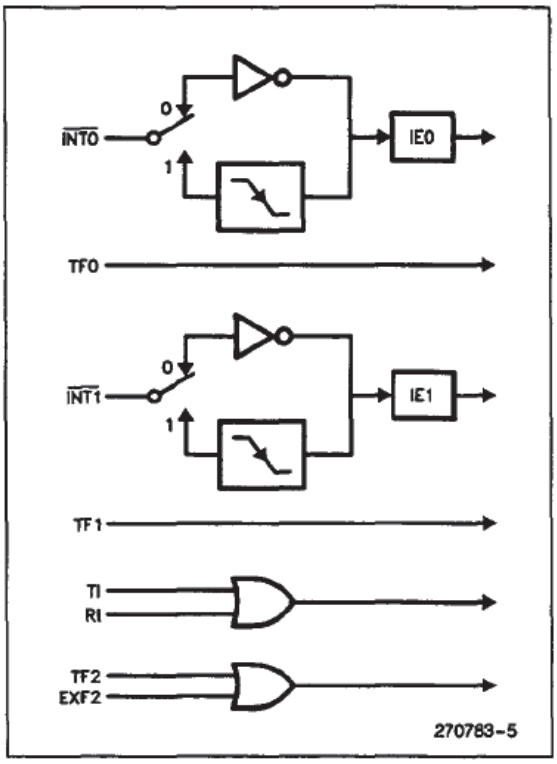



**Figure 6. Interrupt Sources**

To properly terminate Power Down the reset or external interrupt should not be applied before $V_{CC}$ is restored to its normal operating level and must be held active long enough for the oscillator to restart and stabilize (normally less than 10 msec).

With an external interrupt, $\overline{INT0}$ or $\overline{INT1}$ must be enabled and configured as level-sensitive before entering Power Down. Holding the pin low restarts the oscillator and bringing the pin back high completes the exit. After the RETI instruction is executed in the interrupt service routine, the next instruction will be the one following the instruction that put the device in Power Down.

## POWER OFF FLAG

The Power Off Flag (POF) located at PCON.4 is set by hardware when $V_{CC}$ rises from 0 to approximately 5V. POF can also be set or cleared by software. This allows the user to distinguish between a "cold start" reset and a "warm start" reset.

A cold start reset is one that is coincident with $V_{CC}$ being turned on to the device after it was turned off. A warm start reset occurs while $V_{CC}$ is still applied to the device and could be generated, for example, by an exit from Power Down.

Immediately after reset, the user's software can check the status of the POF bit. POF = 1 would indicate a cold start. The software then clears POF and commences its tasks. POF = 0 immediately after reset would indicate a warm start.

$V_{CC}$ must remain above 3V for POF to retain a 0.

## Program Memory Lock

In some microcontroller applications it is desirable that the Program Memory be secure from software piracy. The 8XC5X has varying degrees of program protection depending on the device. Table 8 outlines the lock schemes available for each device.

**Encryption Array:** Within the EPROM/ROM is an array of encryption bytes that are initially unprogrammed (all 1's). For EPROM devices, the user can program the encryption array to encrypt the program code bytes during EPROM verification. For ROM devices, the user submits the encryption array to be programmed by the factory. If an encryption array is submitted, LB1 will also be programmed by the factory. The encryption array is not available without the Lock Bit. Program code verification is performed as usual except that each code byte comes out exclusive-NOR'ed (XNOR) with one of the key bytes. Therefore, to read the ROM/EPROM code, the user has to know the encryption key bytes in their proper sequence.

Unprogrammed bytes have the value 0FFH. If the Encryption Array is left unprogrammed, all the key bytes have the value 0FFH. Since any code byte XNOR'ed

with 0FFH leaves the byte unchanged, leaving the Encryption Array unprogrammed in effect bypasses the encryption feature.

**Program Lock Bits:** Also included in the Program Lock scheme are Lock Bits which can be enabled to provide varying degrees of protection. Table 9 lists the Lock Bits and their corresponding influence on the microcontroller. Refer to Table 8 for the Lock Bits available on the various products. The user is responsible for programming the Lock Bits on EPROM devices. On ROM devices, LB1 is automatically set by the factory when the encryption array is submitted. The Lock Bit is not available without the encryption array on ROM devices.

Erasing the EPROM also erases the Encryption Array and the Lock Bits, returning the part to full functionality.

**Table 8. Program Protection**

| Device | Lock Bits | Encrypt Array |
|---|---|---|
| 80C52 | LB1 | 64 Bytes |
| 80C54 | LB1 | 64 Bytes |
| 80C58 | LB1 | 64 Bytes |
| 87C52 | LB1, LB2, LB3 | 64 Bytes |
| 87C54 | LB1, LB2, LB3 | 64 Bytes |
| 87C58 | LB1, LB2, LB3 | 64 Bytes |

## ONCE MODE

The ON-Circuit Emulation (ONCE) mode facilitates testing and debugging of systems using the 8XC5X without having to remove the device from the circuit. The ONCE mode is invoked by either:

1. Pulling ALE low while the device is in reset and $\overline{PSEN}$ is high;
2. Holding ALE low as RESET is deactivated.

While the device is in ONCE mode, the Port 0 pins go into a float state, and the other port pins, ALE, and $\overline{PSEN}$ are weakly pulled high. The oscillator circuit remains active. While the device is in this mode, an emulator or test CPU can be used to drive the circuit.

Normal operation is restored after a valid reset is applied.

## ADDITIONAL REFERENCES

The following application notes provide supplemental information to this document and can be found in the *Embedded Applications* handbook (Order No. 270648).

1. AP-125 "Designing Microcontroller Systems for Electrically Noisy Environments"
2. AP-155 "Oscillators for Microcontrollers"
3. AP-252 "Designing with the 80C51BH"
4. AP-410 "Enhanced Serial Port on the 83C51FA"

**Table 9. Lock Bits**

| | Program Lock Bits | | | Protection Type |
|---|---|---|---|---|
| | LB1 | LB2 | LB3 | |
| 1 | U | U | U | No program lock features enabled. (Code verify will still be encrypted by the encryption array if programmed.) |
| 2 | P | U | U | MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the EPROM is disabled. |
| 3 | P | P | U | Same as 2, also verify is disabled. |
| 4 | P | P | P | Same as 3, also external execution is disabled. |

P = Programmed
U = Unprogrammed
Any other combination of Lock Bits is not defined.

# 8XC51FX Hardware Description

# 5

# HARDWARE DESCRIPTION OF THE 8XC51FX

## CONTENTS



## CONTENTS

## 1.0 INTRODUCTION

The 8XC51FX is a highly integrated 8-bit microcontroller based on the MCS-51 architecture. As a member of the MCS-51 family, the 8XC51FX is optimized for control applications. Its key feature is the programmable counter array (PCA) which is capable of measuring and generating pulse information on five I/O pins. Also included are an enhanced serial port for multi-processor communications, an up/down timer/counter, and a program lock scheme for the on-chip program memory. Since the 8XC51FX products are CHMOS, they have two software selectable reduced power modes: Idle Mode and Power Down Mode.

The 8XC51FX uses the standard 8051 instruction set and is pin-for-pin compatible with the existing MCS-51 family of products.

This document presents a comprehensive description of the on-chip hardware features of the 8XC51FX. It begins with a discussion of the on-chip memory and then discusses each of the peripherals listed below.

Please note that 8XC51FX does not include the 80C51FA and 83C51FA. Therefore, these devices do not have some of the features found on the 8XC51FX. These features are: programmable clock out, four level interrupt priority structure, enhanced program lock scheme and asynchronous port reset.

- Four 8-Bit Bidirectional Parallel Ports
- Three 16-Bit Timer/Counters with
  - One Up/Down Timer/Counter
  - Clock Out
- Programmable Counter Array with
  - Compare/Capture
  - Software Timer
  - High Speed Output
  - Pulse Width Modulator
  - Watchdog Timer
- Full-Duplex Programmable Serial Port with
  - Framing Error Detection
  - Automatic Address Recognition
- Interrupt Structure with
  - Seven Interrupt Sources
  - Four Priority Levels
- Power-Saving Modes
  - Idle Mode
  - Power Down Mode

Table 1 summarizes the product names and memory differences of the various 8XC51FX products currently available. Throughout this document, the products will generally be referred to as the C51FX.

**Table 1. C51FX Family of Microcontrollers**

| ROM Device | EPROM Version | ROMless Version | ROM/ EPROM Bytes | RAM Bytes |
|---|---|---|---|---|
| 83C51FA | 87C51FA | 80C51FA | 8K | 256 |
| 83C51FB | 87C51FB | 80C51FA | 16K | 256 |
| 83C51FC | 87C51FC | 80C51FA | 32K | 256 |

## 2.0 MEMORY ORGANIZATION

All MCS-51 devices have a separate address space for Program and Data Memory. Up to 64 Kbytes each of external Program and Data Memory can be addressed.

### 2.1 Program Memory

If the $\overline{EA}$ pin is connected to $V_{SS}$, all program fetches are directed to external memory. On the 83C51FA (or 87C51FA), if the $\overline{EA}$ pin is connected to $V_{CC}$, then program fetches to addresses 0000H through 1FFFH are directed to internal ROM and fetches to addresses 2000H through FFFFH are to external memory.

On the 83C51FB (or 87C51FB) if $\overline{EA}$ is connected to VCC, program fetches to addresses 0000H through 3FFFH are directed to internal ROM, and fetches to addresses 4000H through FFFFH are to external memory.

On the 83C51FC (or 87C51FC) if $\overline{EA}$ is connected to $V_{CC}$, program fetches to addresses 0000H through 7FFFH are directed to internal ROM or EPROM and fetches to addresses 8000H through FFFFH are to external memory.

### 2.2 Data Memory

The C51FX implements 256 bytes of on-chip data RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. That means they have the same addresses, but are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the CPU knows whether the access is to the upper 128 bytes of data RAM or to SFR space by the addressing mode used in the instruction. Instructions that use direct addressing access SFR space. For example:

```
MOV 0A0H, #data
```

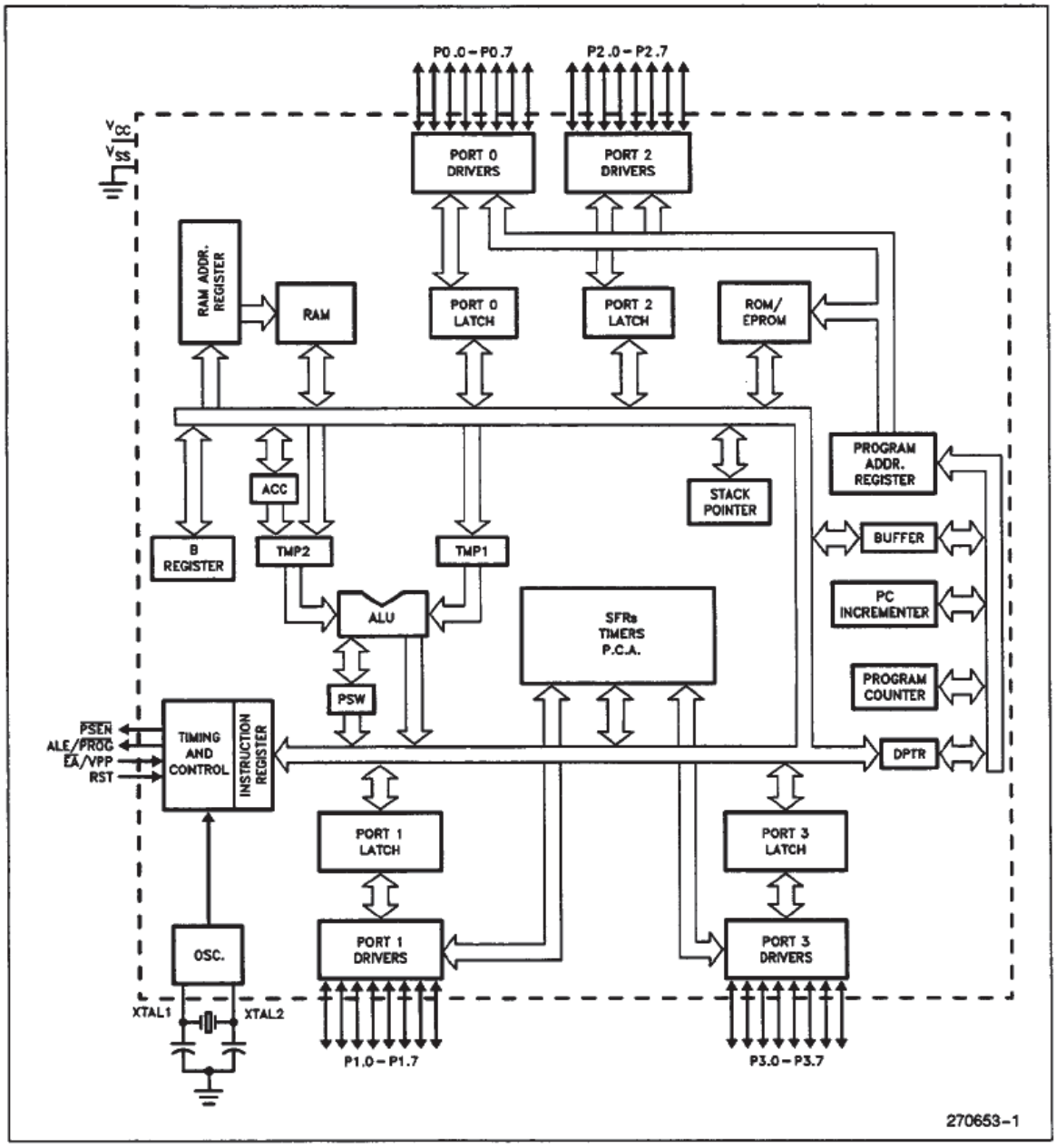



**Figure 1. 8XC51FX Functional Block Diagram**

accesses the SFR at location 0A0H (which is P2). Instructions that use indirect addressing access the upper 128 bytes of data RAM. For example:

MOV @R0, #data

where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H). Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

## 3.0 SPECIAL FUNCTION REGISTERS

A map of the on-chip memory area called the SFR (Special Function Register) space is shown in Table 2.

Note that not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

User software should not write 1s to these unimplemented locations, since they may be used in future MCS-51 products to invoke new features. In that case the reset or inactive values of the new bits will always be 0, and their active values will be 1.

The functions of the SFRs are outlined below. More information on the use of specific SFRs for each peripheral is included in the description of that peripheral.

**Accumulator:** ACC is the Accumulator register. The mnemonics for Accumulator-Specific instructions, however, refer to the Accumulator simply as A.

**Table 2. SFR Mapping and Reset Values**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| F8 | | CH<br>00000000 | CCAP0H<br>XXXXXXXX | CCAP1H<br>XXXXXXXX | CCAP2H<br>XXXXXXXX | CCAP3H<br>XXXXXXXX | CCAP4H<br>XXXXXXXX | | FF |
| F0 | * B<br>00000000 | | | | | | | | F7 |
| E8 | | CL<br>00000000 | CCAP0L<br>XXXXXXXX | CCAP1L<br>XXXXXXXX | CCAP2L<br>XXXXXXXX | CCAP3L<br>XXXXXXXX | CCAP4L<br>XXXXXXXX | | EF |
| E0 | * ACC<br>00000000 | | | | | | | | E7 |
| D8 | CCON<br>00X00000 | CMOD<br>00XXX000 | CCAPM0<br>X0000000 | CCAPM1<br>X0000000 | CCAPM2<br>X0000000 | CCAPM3<br>X0000000 | CCAPM4<br>X0000000 | | DF |
| D0 | * PSW<br>00000000 | | | | | | | | D7 |
| C8 | T2CON<br>00000000 | T2MOD<br>XXXXXX00 | RCAP2L<br>00000000 | RCAP2H<br>00000000 | TL2<br>00000000 | TH2<br>00000000 | | | CF |
| C0 | | | | | | | | | C7 |
| B8 | * IP<br>X0000000 | SADEN<br>00000000 | | | | | | | BF |
| B0 | * P3<br>11111111 | | | | | | | IPH<br>X0000000 | B7 |
| A8 | * IE<br>00000000 | SADDR<br>00000000 | | | | | | | AF |
| A0 | * P2<br>11111111 | | | | | | | | A7 |
| 98 | * SCON<br>00000000 | * SBUF<br>XXXXXXXX | | | | | | | 9F |
| 90 | * P1<br>11111111 | | | | | | | | 97 |
| 88 | * TCON<br>00000000 | * TMOD<br>00000000 | * TL0<br>00000000 | * TL1<br>00000000 | * TH0<br>00000000 | * TH1<br>00000000 | | | 8F |
| 80 | * P0<br>11111111 | * SP<br>00000111 | * DPL<br>00000000 | * DPH<br>00000000 | | | | * PCON **<br>00XX0000 | 87 |

* = Found in the 8051 core (See 8051 Hardware Description for explanations of these SFRs).
** = See description of PCON SFR. Bit PCON.4 is not affected by reset.
X = Undefined.

**Table 3. PSW: Program Status Word Register**

**PSW** Address = 0D0H Reset Value = 0000 0000B

Bit Addressable

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

| Symbol | Function |
|---|---|
| CY | Carry flag. |
| AC | Auxiliary Carry flag. (For BCD Operations) |
| F0 | Flag 0. (Available to the user for general purposes). |
| RS1 | Register bank select bit 1. |
| RS0 | Register bank select bit 0. |

| RS1 | RS0 | Working Register Bank and Address | |
|---|---|---|---|
| 0 | 0 | Bank 0 | (00H–07H) |
| 0 | 1 | Bank 1 | (08H–0FH) |
| 1 | 0 | Bank 2 | (10H–17H) |
| 1 | 1 | Bank 3 | (18H–1FH) |

| | |
|---|---|
| OV | Overflow flag. |
| — | User definable flag. |
| P | Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the Accumulator, i.e., even parity. |

**B Register:** The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

**Stack Pointer:** The Stack Pointer Register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. The stack may reside anywhere in on-chip RAM. On reset, the Stack Pointer is initialized to 07H causing the stack to begin at location 08H.

**Data Pointer:** The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address, but it may be manipulated as a 16-bit register or as two independent 8-bit registers.

**Program Status Word:** The PSW register contains program status information as detailed in Table 3.

**Ports 0 to 3 Registers:** P0, P1, P2, and P3 are the SFR latches of Port 0, Port 1, Port 2, and Port 3 respectively.

**Timer Registers:** Register pairs (TH0, TL0), (TH1, TL1), and (TH2, TL2) are the 16-bit count registers for Timer/Counters 0, 1, and 2 respectively. Control and status bits are contained in registers TCON and TMOD for Timers 0 and 1 and in registers T2CON and T2MOD for Timer 2. The register pair (RCAP2H, RCAP2L) are the capture/reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

**Programmable Counter Array (PCA) Registers:** The 16-bit PCA timer/counter consists of registers CH and CL. Registers CCON and CMOD contain the control and status bits for the PCA. The CCAPMn (n = 0, 1, 2, 3, or 4) registers control the mode for each of the five PCA modules. The register pairs (CCAPnH, CCAPnL) are the 16-bit compare/capture registers for each PCA module.

**Serial Port Registers:** The Serial Data Buffer, SBUF, is actually two separate registers: a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF initiates the transmission). When data is moved from SBUF, it comes from the receive buffer. Register SCON contains the control and status bits for the Serial Port. Registers SADDR and SADEN are used to define the Given and the Broadcast addresses for the Automatic Address Recognition feature.

**Interrupt Registers:** The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the 7 interrupts in the IP register.

**Power Control Register:** PCON controls the Power Reduction Modes. Idle and Power Down Modes.

## 4.0 PORT STRUCTURES AND OPERATION

All four ports in the C51FX are bidirectional. Each consists of a latch (Special Function Registers P0 through P3), an output driver, and an input buffer.

The output drivers of Ports 0 and 2, and the input buffers of Port 0, are used in accesses to external memory. In this application, Port 0 outputs the low byte of the external memory address, time-multiplexed with the byte being written or read. Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise the Port 2 pins continue to emit the P2 SFR content.

All the Port 1 and Port 3 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed in Table 4.

The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise the port pin is stuck at 0.

### 4.1 I/O Configurations

Figure 2 shows a functional diagram of a typical bit latch and I/O buffer in each of the four ports. The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which clocks in a value from the internal bus in response to a "write to latch" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in response to a "read latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal. See the Read-Modify-Write Feature section.

As shown in Figure 2, the output drivers of Ports 0 and 2 are switchable to an internal ADDRESS and ADDRESS/DATA bus by an internal CONTROL signal for use in external memory accesses. During external memory accesses, the P2 SFR remains unchanged, but the P0 SFR gets 1s written to it.

**Table 4. Alternate Port Functions**

| Port Pin | Alternate Function |
|---|---|
| P0.0/AD0–P0.7/AD7 | Multiplexed Byte of Address/Data for External Memory |
| P1.0/T2 | Timer 2 External Clock Input/Clock-Out |
| P1.1/T2EX | Timer 2 Reload/Capture/Direction Control |
| P1.2/ECI | PCA External Clock Input |
| P1.3/CEX0 | PCA Module 0 Capture Input, Compare/PWM Output |
| P1.4/CEX1 | PCA Module 1 Capture Input, Compare/PWM Output |
| P1.5/CEX2 | PCA Module 2 Capture Input, Compare/PWM Output |
| P1.6/CEX3 | PCA Module 3 Capture Input, Compare/PWM Output |
| P1.7/CEX4 | PCA Module 4 Capture Input, Compare/PWM Output |
| P2.0/A8–P2.7/A15 | High Byte of Address for External Memory |
| P3.0/RXD | Serial Port Input |
| P3.1/TXD | Serial Port Output |
| P3.2/$\overline{\text{INT0}}$ | External Interrupt 0 |
| P3.3/$\overline{\text{INT}}$ | External Interrupt 1 |
| P3.4/T0 | Timer 0 External Clock Input |
| P3.5/T1 | Timer 1 External Clock Input |
| P3.6/$\overline{\text{WR}}$ | Write Strobe for External Memory |
| P3.7/$\overline{\text{RD}}$ | Read Strobe for External Memory |

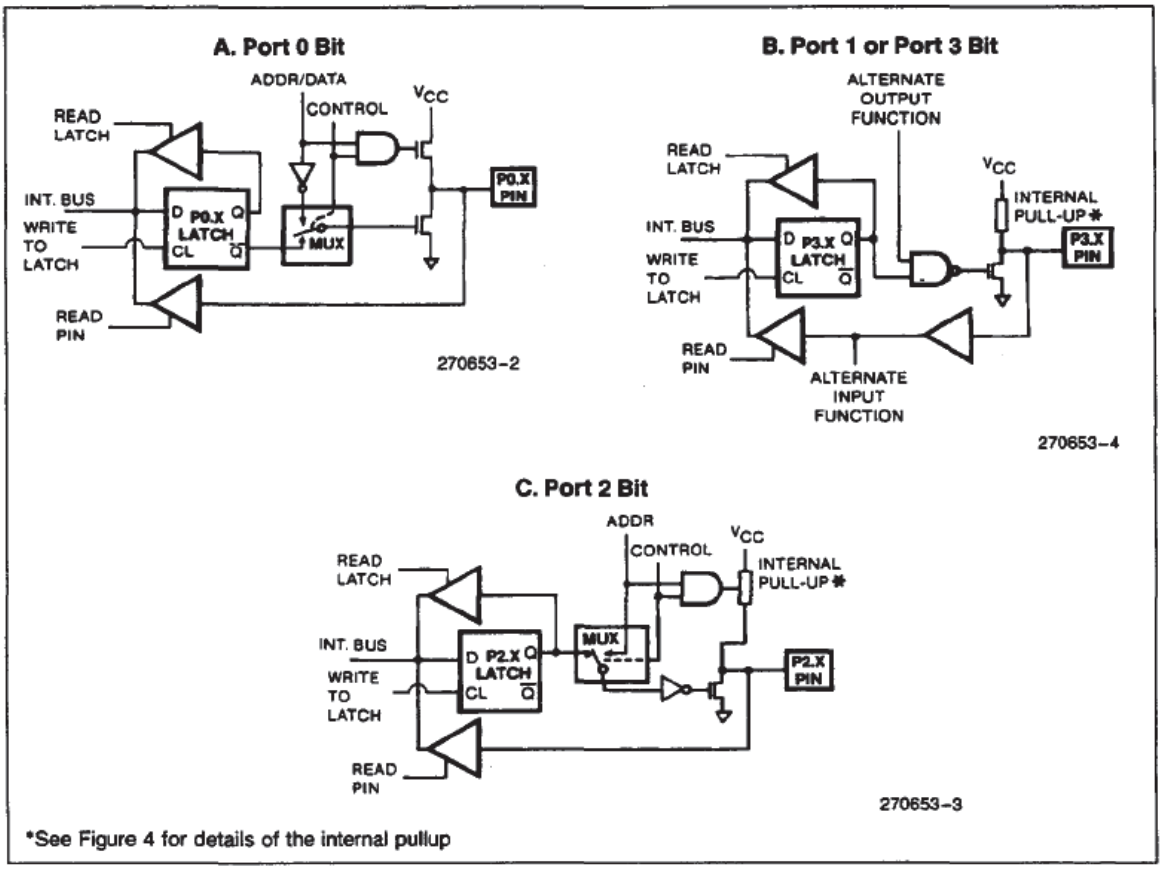



**Figure 2. C51FX Port Bit Latches and I/O Buffers**

Also shown in Figure 2 is that if a P1 or P3 latch contains a 1, then the output level is controlled by the signal labeled "alternate output function." The actual pin level is always available to the pin's alternate input function, if any.

Ports 1, 2, and 3 have internal pullups. Port 0 has open drain outputs. Each I/O line can be independently used as an input or an output (Ports 0 and 2 may not be used as general purpose I/O when being used as the ADDRESS/DATA BUS). To be used as an input, the port bit latch must contain a 1, which turns off the output driver FET. On Ports 1, 2, and 3, the pin is pulled high by the internal pullup, but can be pulled low by an external source.

Port 0 differs from the other ports in not having internal pullups. The pullup FET in the P0 output driver (see Figure 2) is used only when the Port is emitting 1s during external memory accesses. Otherwise the pullup FET is off. Consequently P0 lines that are being used as output port lines are open drain. Writing a 1 to the bit latch leaves both output FETs off, which floats the pin and allows it to be used as a high-impedance input. Because Ports 1 through 3 have fixed internal pullups they are sometimes call "quasi-bidirectional" ports. When configured as inputs they pull high and will source current (IIL in the data sheets) when externally pulled low. Port 0, on the other hand, is considered "true" bidirectional, because it floats when configured as an input.

All the port latches have 1s written to them by the reset function. If a 0 is subsequently written to a port latch, it can be reconfigured as an input by writing a 1 to it.

## 4.2 Writing to a Port

In the execution of an instruction that changes the value in a port latch, the new value arrives at the latch during State 6 Phase 2 of the final cycle of the instruction. However, port latches are in fact sampled by their output buffers only during Phase 1 of any clock period. (During Phase 2 the output buffer holds the value it saw during the previous Phase 1). Consequently, the new value in the port latch won't actually appear at the output pin until the next Phase 1, which will be at S1P1 of the next machine cycle. Refer to Figure 3. For more information on internal timings refer to the CPU Timing section.

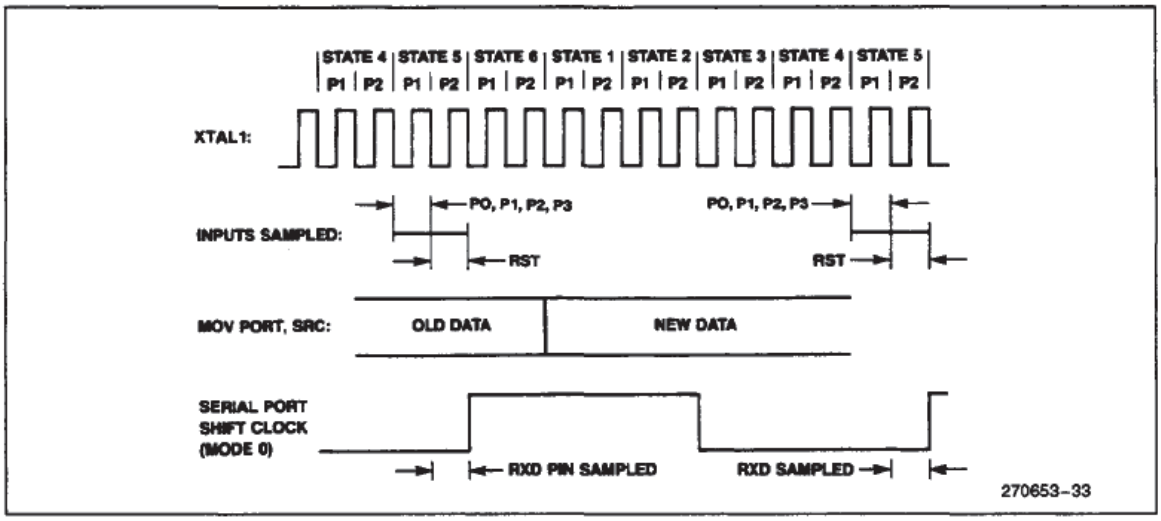


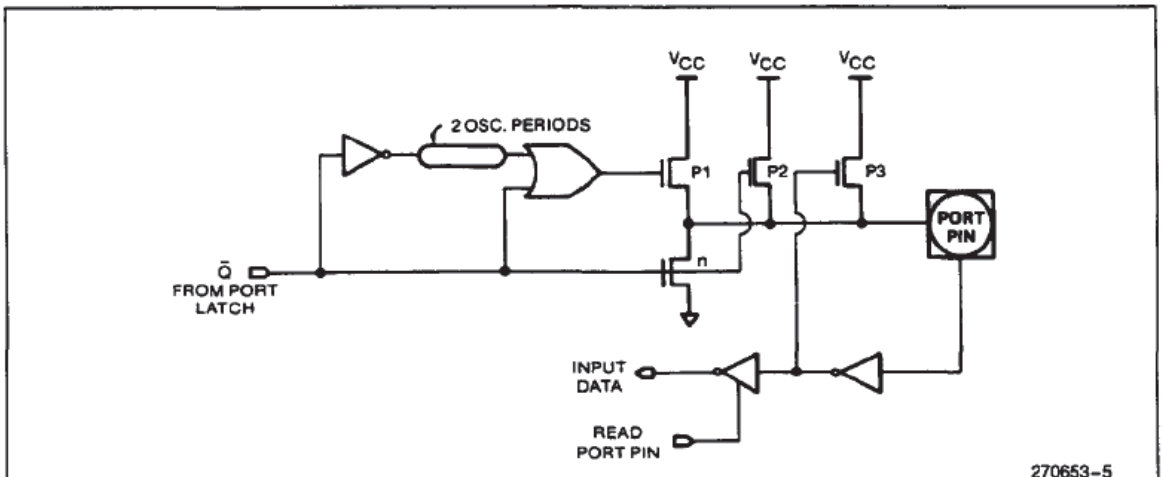

**Figure 3. Port Operation**

If the change requires a 0-to-1 transition in Ports 1, 2, and 3, an additional pullup is turned on during S1P1 and S1P2 of the cycle in which the transition occurs. This is done to increase the transition speed. The extra pullup can source about 100 times the current that the normal pullup can. The internal pullups are field-effect transistors, not linear resistors. The pull-up arrangements are shown in Figure 4.

The pullup consists of three pFETs. Note that an n-channel FET (nFET) is turned on when a logical 1 is applied to its gate, and is turned off when a logical 0 is applied to its gate. A p-channel FET (pFET) is the opposite: it is on when its gate sees a 0, and off when its gate sees a 1.

pFET 1 in is the transistor that is turned on for 2 oscillator periods after a 0-to-1 transition in the port latch. A 1 at the port pin turns on pFET3 (a weak pull-up), through the invertor. This invertor and pFET form a latch which hold the 1.

If the pin is emitting a 1, a negative glitch on the pin from some external source can turn off pFET3, causing the pin to go into a float state. pFET2 is a very weak pullup which is on whenever the nFET is off, in traditional CMOS style. It's only about $1/10$ the strength of pFET3. Its function is to restore a 1 to the pin in the event the pin had a 1 and lost it to a glitch.




CHMOS Configuration. pFET 1 is turned on for 2 osc. periods after $\overline{Q}$ makes a 0-to-1 transition. During this time, pFET 1 also turns on pFET 3 through the inverter to form a latch which holds the 1. pFET 2 is also on. Port 2 is similar except that it holds the strong pullup on while emitting 1s that are address bits. (See text, "Accessing External Memory".)

**Figure 4. Ports 1 and 3 Internal Pullup Configurations**

## 4.3 Port Loading and Interfacing

The output buffers of Ports 1, 2, and 3 can each sink 1.6 mA at 0.45 V. These port pins can be driven by open-collector and open-drain outputs although 0-to-1 transitions will not be fast since there is little current pulling the pin up. An input 0 turns off pullup pFET3, leaving only the very weak pullup pFET2 to drive the transition.

In external bus mode, Port 0 output buffers can each sink 3.2 mA at 0.45 V. However, as port pins they require external pullups to be able to drive any inputs.

See the latest revision of the data sheet for design-in information.

## 4.4 Read-Modify-Write Feature

Some instructions that read a port read the latch and others read the pin. Which ones do which? The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write" instructions. Listed below are the read-modify-write instructions. When the destination operand is a port, or a port bit, these instructions read the latch rather than the pin:

| | |
|---|---|
| ANL | (logical AND, e.g., ANL P1, A) |
| ORL | (logical OR, e.g., ORL P2, A) |
| XRL | (logical EX-OR, e.g., XRL P3, A) |
| JBC | (jump if bit = 1 and clear bit, e.g., JBC P1.1, LABEL) |
| CPL | (complement bit, e.g., CPL P3.0) |
| INC | (increment, e.g., INC P2) |
| DEC | (decrement, e.g., DEC P2) |
| DJNZ | (decrement and jump if not zero, e.g., DJNZ P3, LABEL) |
| MOV, PX.Y, C | (move carry bit to bit Y of Port X) |
| CLR PX.Y | (clear bit Y of Port X) |
| SETB PX.Y | (set bit Y of Port X) |

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of a transistor. When a 1 is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as a 0. Reading the latch rather than the pin will return the correct value of 1.

## 4.5 Accessing External Memory

Accesses to external memory are of two types: accesses to external Program Memory and accesses to external Data Memory. Accesses to external Program Memory use signal $\overline{PSEN}$ (program store enable) as the read strobe. Accesses to external Data Memory use $\overline{RD}$ or $\overline{WR}$ (alternate functions of P3.7 and P3.6) to strobe the memory. Refer to Figures 5 through 7.

Fetches from external Program Memory always use a 16-bit address. Accesses to external Data Memory can use either a 16-bit address (MOVX @ DPTR) or an 8-bit address (MOVX @ Ri).

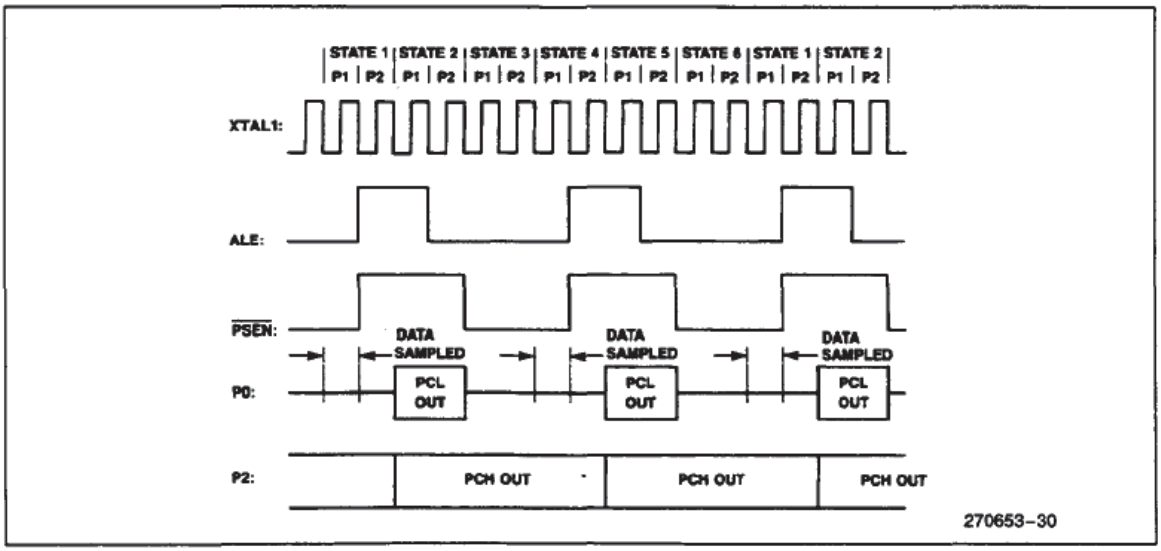



**Figure 5. External Program Memory Fetches**

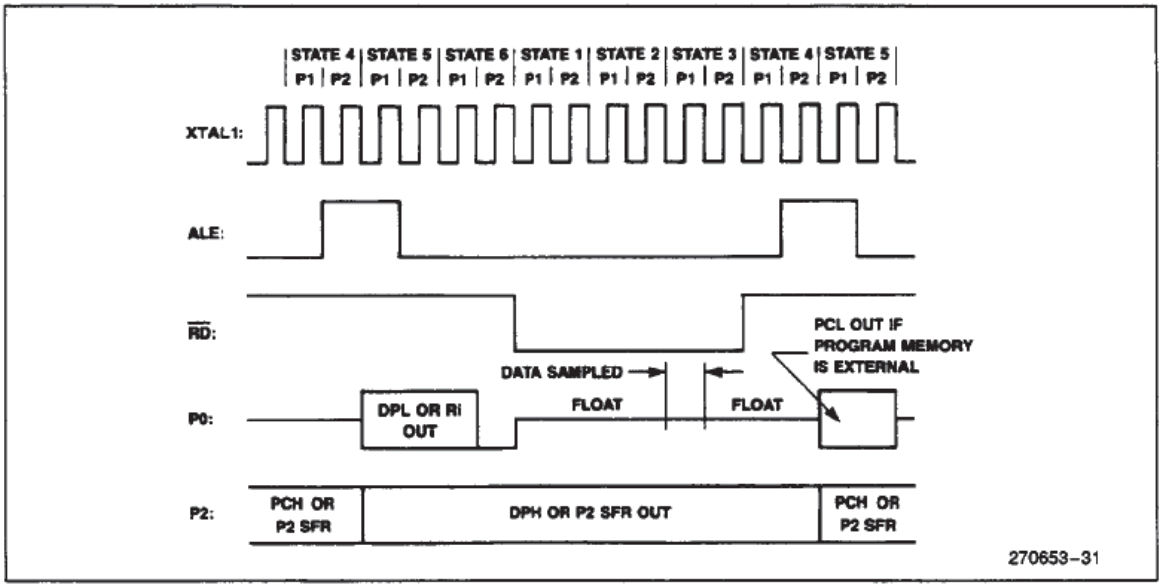



**Figure 6. External Data Memory Read Cycle**

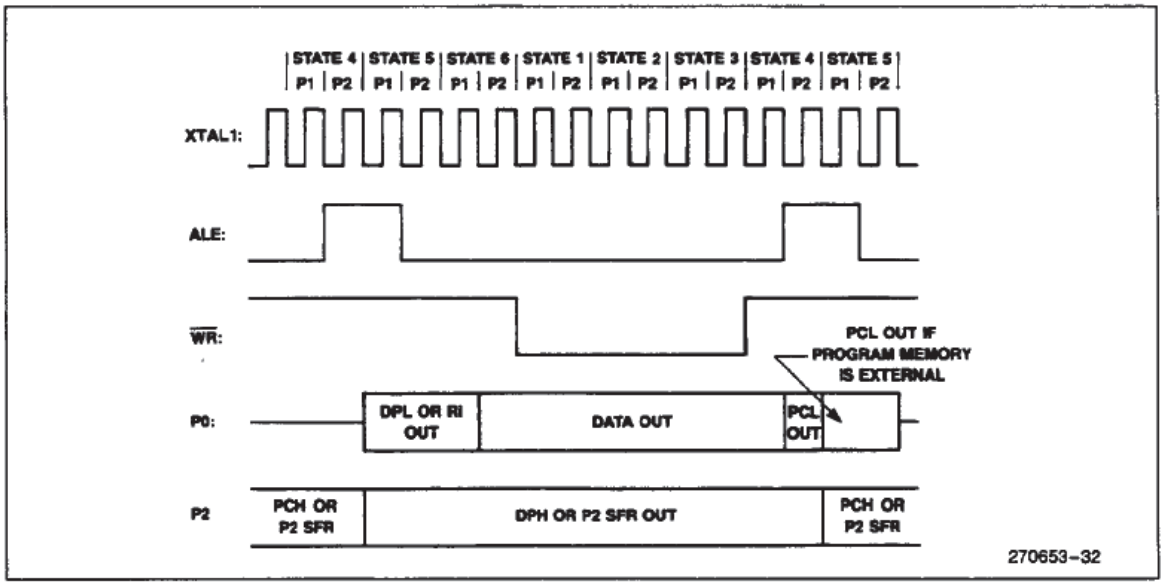



**Figure 7. External Data Memory Write Cycle**

Whenever a 16-bit address is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle. The Port 2 drivers use the strong pullups during the entire time that they are emitting address bits that are 1s. This occurs when the MOVX @ DPTR instruction is executed. During this time the Port 2 latch (the Special Function Register) does not have to contain 1s, and the contents of the Port 2 SFR are not modified. If the external memory cycle is not immediately followed by another external memory cycle, the undisturbed contents of the Port 2 SFR will reappear in the next cycle.

If an 8-bit address is being used (MOVX @ Ri), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. In this case, Port 2 pins can be used to page the external data memory.

In either case, the low byte of the address is time-multiplexed with the data byte on Port 0. The ADDRESS/DATA signal drives both FETs in the Port 0 output buffers. Thus, in external bus mode the Port 0 pins are not open-drain outputs and do not require external pullups. The ALE (Address Latch Enable) signal should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before $\overline{WR}$ is activated, and remains there until after $\overline{WR}$ is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe ($\overline{RD}$) is deactivated.

During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating the information in the Port 0 SFR. Also, a MOV P0 instruction must not take place during external memory accesses. If the user writes to Port 0 during an external memory fetch, the incoming code byte is corrupted. Therefore, do not write to Port 0 if external program memory is used.

External Program Memory is accessed under two conditions:

1. Whenever signal $\overline{EA}$ is active, or
2. Whenever the program counter (PC) contains an address greater than 1FFFH (8K) for the 8XC51FA or 3FFFH (16K) for the 8XC51FB, or 7FFFH (32K) for the 87C51FC.

This requires that the ROMless versions have $\overline{EA}$ wired to $V_{SS}$ enable the lower 8K, 16K, or 32K program bytes to be fetched from external memory.

When the CPU is executing out of external Program Memory, all 8 bits of Port 2 are dedicated to an output function and may not be used for general purpose I/O. During external program fetches they output the high byte of the PC with the Port 2 drivers using the strong pullups to emit bits that are 1s.

## 5.0 TIMERS/COUNTERS

The C51FX has three 16-bit Timer/Counters: Timer 0, Timer 1, and Timer 2. Each consists of two 8-bit registers, THx and TLx, (x = 0, 1, and 2). All three can be configured to operate either as timers or event counters.

In the Timer function, the TLx register is incremented every machine cycle. Thus one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin—T0, T1, or T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is $1/_{24}$ of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full machine cycle.

In addition to the Timer or Counter selection, Timer 0 and Timer 1 have four operating modes from which to select: Modes 0 – 3. Timer 2 has three modes of operation: Capture, Auto-Reload, and Baud Rate Generator.

### 5.1 Timer 0 and Timer 1

The Timer or Counter function is selected by control bits C/$\overline{T}$ in the Special Function Register TMOD (Table 5). These two Timer/Counters have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for both Timer/Counters. Mode 3 operation is different for the two timers.

#### MODE 0

Either Timer 0 or Timer 1 in Mode 0 is an 8-bit Counter with a divide-by-32 prescaler. Figure 8 shows the Mode 0 operation for either timer.

In this mode, the Timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TFx. The counted input is enabled to the Timer when TRx = 1 and either GATE = 0 or $\overline{INTx}$ = 1. (Setting GATE = 1 allows the Timer to be controlled by external input $\overline{INTx}$, to facilitate pulse width measurements). TRx and TFx are

control bits in SFR TCON (Table 6). The GATE bit is in TMOD. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

The 13-bit register consists of all 8 bits of THx and the lower 5 bits of TLx. The upper 3 bits of TLx are indeterminate and should be ignored. Setting the run flag (TRx) does not clear these registers.

### MODE 1

Mode 1 is the same as Mode 0, except that the Timer register uses all 16 bits. Refer to Figure 9. In this mode, THx and TLx are cascaded; there is no prescaler.

### MODE 2

Mode 2 configures the Timer register as an 8-bit Counter (TLx) with automatic reload, as shown in Figure 10. Overflow from TLx not only sets TFx, but also reloads TLx with the contents of THx, which is preset by software. The reload leaves THx unchanged.

**Table 5. TMOD: Timer/Counter Mode Control Register**

**TMOD** Address = 89H Reset Value = 0000 0000B

Not Bit Addressable

| | TIMER 1 | | | | TIMER 0 | | | |
|---|---|---|---|---|---|---|---|---|
| | GATE | C/$\overline{T}$ | M1 | M0 | GATE | C/$\overline{T}$ | M1 | M0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| GATE | Gating control when set. Timer/Counter 0 or 1 is enabled only while $\overline{INT0}$ or $\overline{INT1}$ pin is high and TR0 or TR1 control pin is set. When cleared, Timer 0 or 1 is enabled whenever TR0 or TR1 control bit is set. |
| C/$\overline{T}$ | Timer or Counter Selector. Clear for Timer operation (input from internal system clock). Set for Counter operation (input from T0 or T1 input pin). |

| M1 | M0 | Operating Mode |
|---|---|---|
| 0 | 0 | 8-bit Timer/Counter. THx with TLx as 5-bit prescaler. |
| 0 | 1 | 16-bit Timer/Counter. THx and TLx are cascaded; there is no prescaler. |
| 1 | 0 | 8-bit auto-reload Timer/Counter. THx holds a value which is to be reloaded into TLx each time it overflows. |
| 1 | 1 | (Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits. |
| 1 | 1 | (Timer 1) Timer/Counter stopped. |

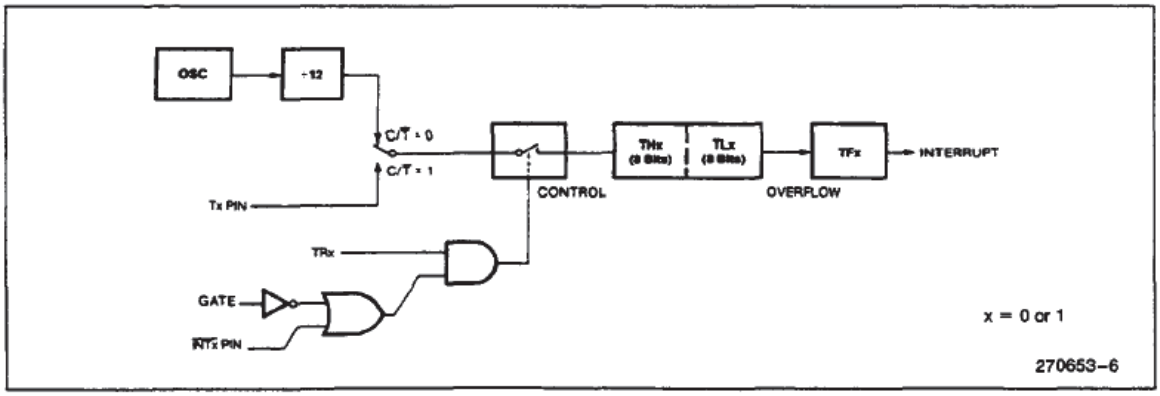



**Figure 8. Timer/Counter 0 or 1 in Mode 0: 13-Bit Counter**

**Table 6. TCON: Timer/Counter Control Register**

**TCON** Address = 88H Reset Value = 0000 0000B

Bit Addressable

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

| Symbol | Function |
|---|---|
| TF1 | Timer 1 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine. |
| TR1 | Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter 1 on/off. |
| TF0 | Timer 0 overflow Flag. Set by hardware on Timer/Counter 0 overflow. Cleared by hardware when processor vectors to interrupt routine. |
| TR0 | Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter 0 on/off. |
| IE1 | Interrupt 1 flag. Set by hardware when external interrupt 1 edge is detected (transmitted or level-activated). Cleared when interrupt processed only if transition-activated. |
| IT1 | Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupt 1. |
| IE0 | Interrupt 0 flag. Set by hardware when external interrupt 0 edge is detected (transmitted or level-activated). Cleared when interrupt processed only if transition-activated. |
| IT0 | Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupt 0. |

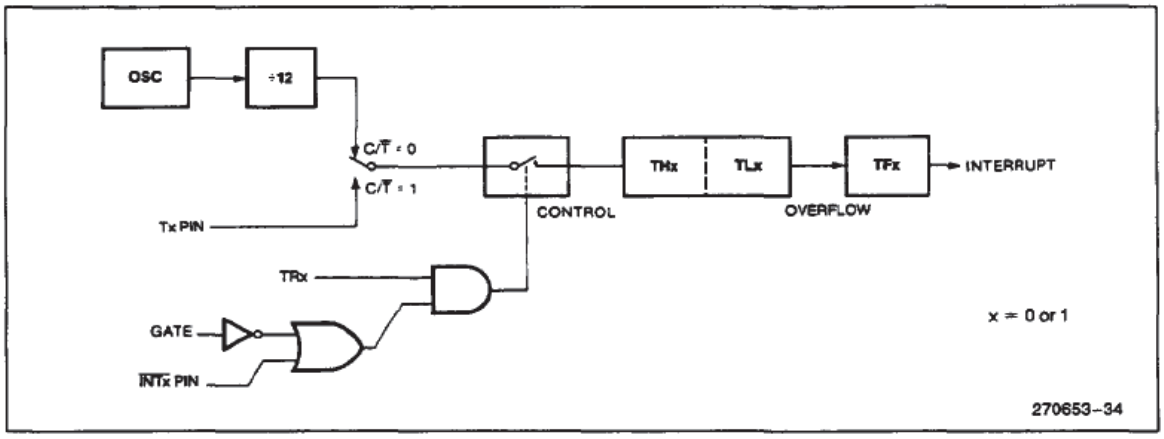



**Figure 9. Timer/Counter 0 or 1 in Mode 1: 16-Bit Counter**

## MODE 3

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0.

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 11. TL0 uses the Timer 0 control bits: C/$\overline{T}$, GATE, TR0, $\overline{INT0}$, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus TH0 now controls the Timer 1 interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer or counter. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3, or can still be used by the serial port as a baud rate generator, or in any application not requiring an interrupt.

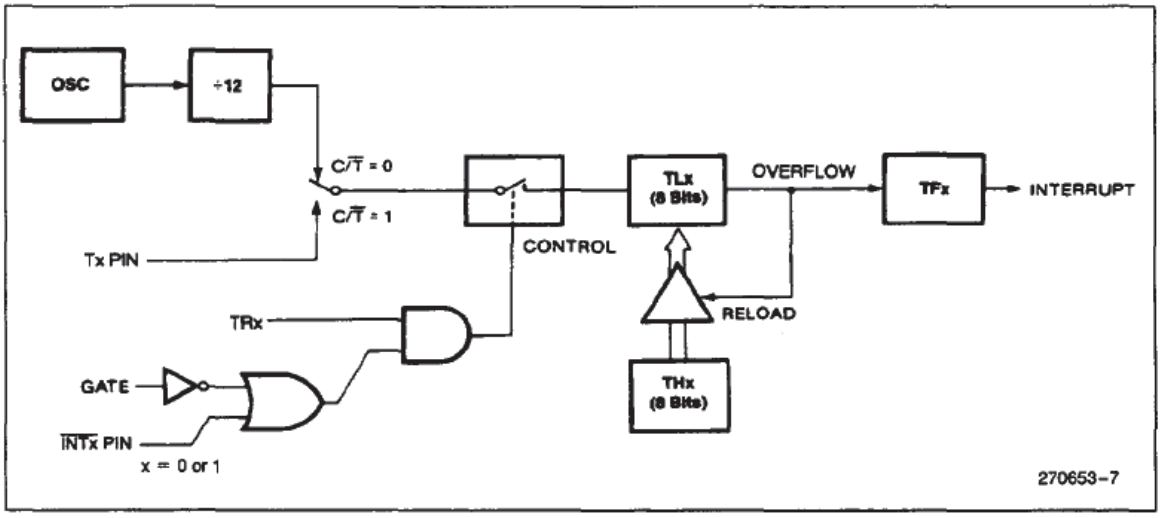



**Figure 10. Timer/Counter 1 Mode 2: 8-Bit Auto-Reload**

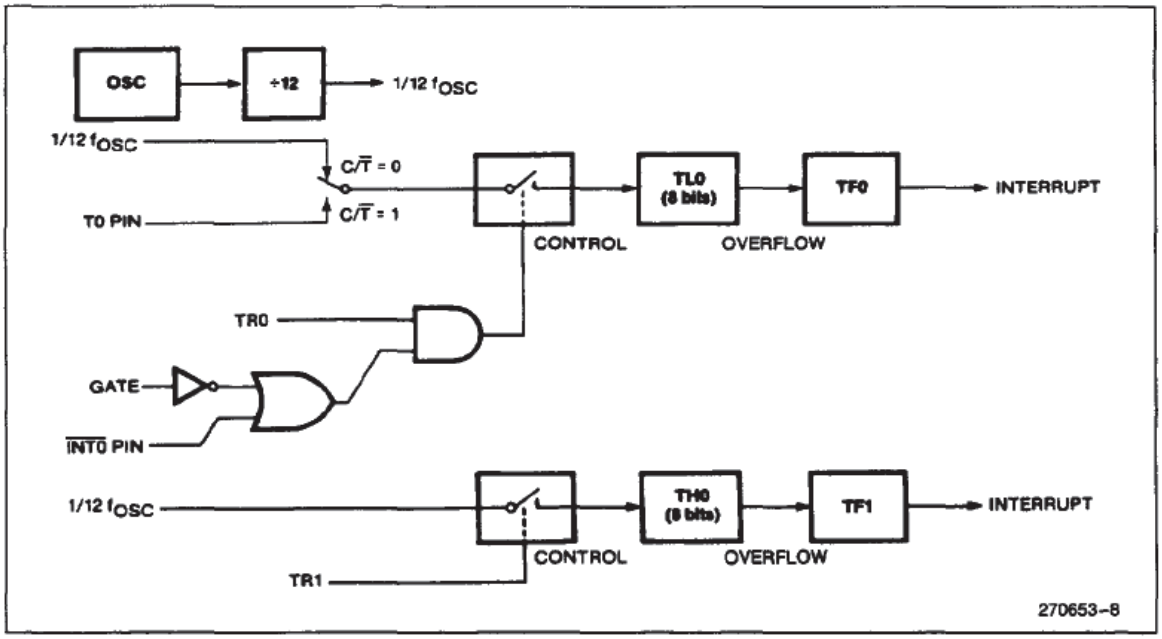



**Figure 11. Timer/Counter 0 Mode 3: Two 8-Bit Counters**

## 5.2 Timer 2

Timer 2 is a 16-bit Timer/Counter which can operate either as a timer or as an event counter. This is selected by bit C/$\overline{T2}$ in the Special Function Register T2CON (Table 8). It has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON as shown in Table 7.

**Table 7. Timer 2 Operating Modes**

| RCLK + TCLK | CP/$\overline{RL2}$ | T2*OE | TR2 | Mode |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 16-Bit Auto-Reload |
| 0 | 1 | 0 | 1 | 16-Bit Capture |
| 1 | X | X | 1 | Baud Rate Generator |
| X | 0 | 1 | 1 | Clock-Out on P1.0 |
| X | X | X | 0 | Timer Off |

**Table 8. T2CON: Timer/Counter 2 Control Register**

**T2CON** Address = 0C8H Reset Value = 0000 0000B

Bit Addressable

| | TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/$\overline{T2}$ | CP/$\overline{RL2}$ |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| TF2 | Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1. |
| EXF2 | Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1). |
| RCLK | Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock. |
| TCLK | Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock. |
| EXEN2 | Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX. |
| TR2 | Start/stop control for Timer 2. A logic 1 starts the timer. |
| C/$\overline{T2}$ | Timer or counter select. (Timer 2)<br>0 = Internal timer (OSC/12 or OSC/2 in baud rate generator mode).<br>1 = External event counter (falling edge triggered). |
| CP/$\overline{RL2}$ | Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow. |

## CAPTURE MODE

In the capture mode there are two options selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16-bit timer or counter which upon overflow sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 still does the above, but with the added feature that a 1-to-0 tran-

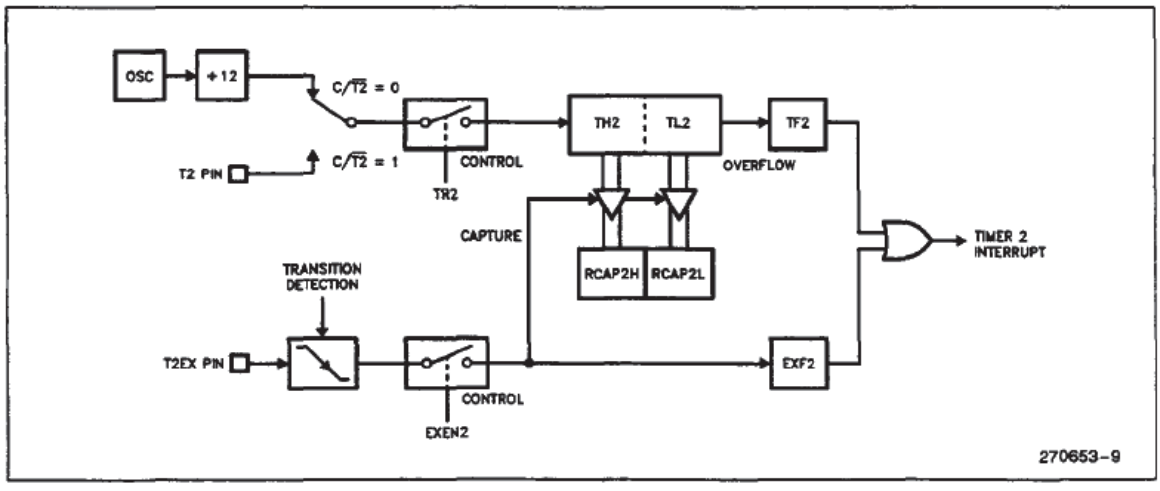



**Figure 12. Timer 2 in Capture Mode**

sition at external input T2EX causes the current value in the Timer 2 registers, TH2 and TL2, to be captured into registers RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set. The EXF2 bit, like TF2, can generate an interrupt. The capture mode is illustrated in Figure 12.

## AUTO-RELOAD MODE (UP OR DOWN COUNTER)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by a bit named DCEN (Down Counter Enable) located in the SFR T2MOD (see Table 9). Upon reset the DCEN bit is set to 0 so that Timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down depending on the value of the T2EX pin.

Figure 13 shows Timer 2 automatically counting up when DCEN = 0. In this mode there are two options selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in RCAP2H and RCAP2L are preset by software. If EXEN2 = 1, a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX. This transition also sets the EXF2 bit. Either the TF2 or EXF2 bit can generate the Timer 2 interrupt if it is enabled.

**Table 9. T2MOD: Timer 2 Mode Control Register**

**T2MOD** Address = 0C9H Reset Value = XXXX XX00B

Not Bit Addressable

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

| Symbol | Function |
|---|---|
| — | Not implemented, reserved for future use.* |
| T2OE | Timer 2 Output Enable bit. |
| DCEN | Down Count Enable bit. When set, this allows Timer 2 to be configured as an up/down *counter.* |

*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

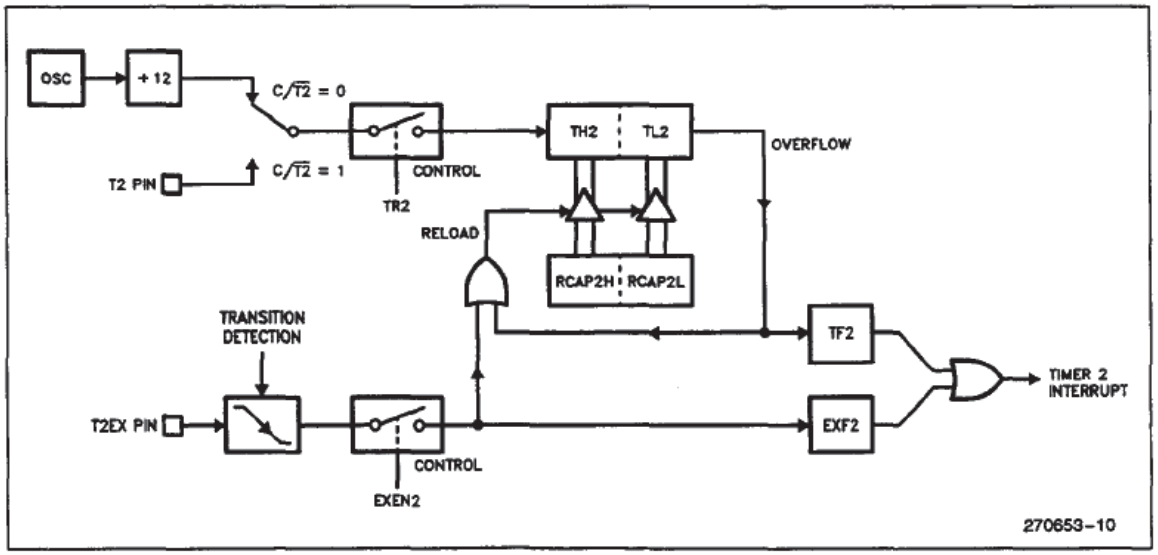



**Figure 13. Timer 2 Auto Reload Mode (DCEN = 0)**

Setting the DCEN bit enables Timer 2 to count up or down as shown in Figure 14. In this mode the T2EX pin controls the direction of count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit which can then generate an interrupt if it is enabled. This overflow also causes a the 16-bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

A logic 0 at T2EX makes Timer 2 count down. Now the timer underflows when TH2 and TL2 equal the values stored in RCAP2H and RCAP2L. The underflow sets the TF2 bit and causes 0FFFFH to be reloaded into the timer registers.

The EXF2 bit toggles whenever Timer 2 overflows or underflows. This bit can be used as a 17th bit of resolution if desired. In this operating mode, EXF2 does not generate an interrupt.

### BAUD RATE GENERATOR MODE

The baud rate generator mode is selected by setting the RCLK and/or TCLK bits in T2CON. Timer 2 in this mode will be described in conjunction with the serial port.

### PROGRAMMABLE CLOCK OUT

A 50% duty cycle clock can be programmed to come out on P1.0. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed (1) to input the external clock for Timer/Counter 2 or (2) to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz at a 16 MHz operating frequency.

To configure the Timer/Counter 2 as a clock generator, bit C/T2 (in T2CON) must be cleared and bit T2OE in T2MOD must be set. Bit TR2 (T2CON.2) also must be set to start the timer (see Table 6 for operating modes).

The Clock-out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L) as shown in this equation:

$$\text{Clock-out Frequency} = \frac{\text{Oscillator Frequency}}{4 \times (65536 - \text{RCAP2H, RCAP2L})}$$

In the Clock-Out mode Timer 2 roll-overs will not generate an interrupt. This is similar to when Timer 2 is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and Clock-out frequencies cannot be determined independently of one another since they both use the values in RCAP2H and RCAP2L.

## 6.0 PROGRAMMABLE COUNTER ARRAY

The Programmable Counter Array (PCA) consists of a 16-bit timer/counter and five 16-bit compare/capture modules as shown in Figure 15a. The PCA timer/counter serves as a common time base for the five modules and is the only timer which can service the PCA. Its clock input can be programmed to count any one of the following signals:

- oscillator frequency ÷ 12
- oscillator frequency ÷ 4
- Timer 0 overflow
- external input on ECI (P1.2).

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

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

Module 4 can also be programmed as a watchdog timer.

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

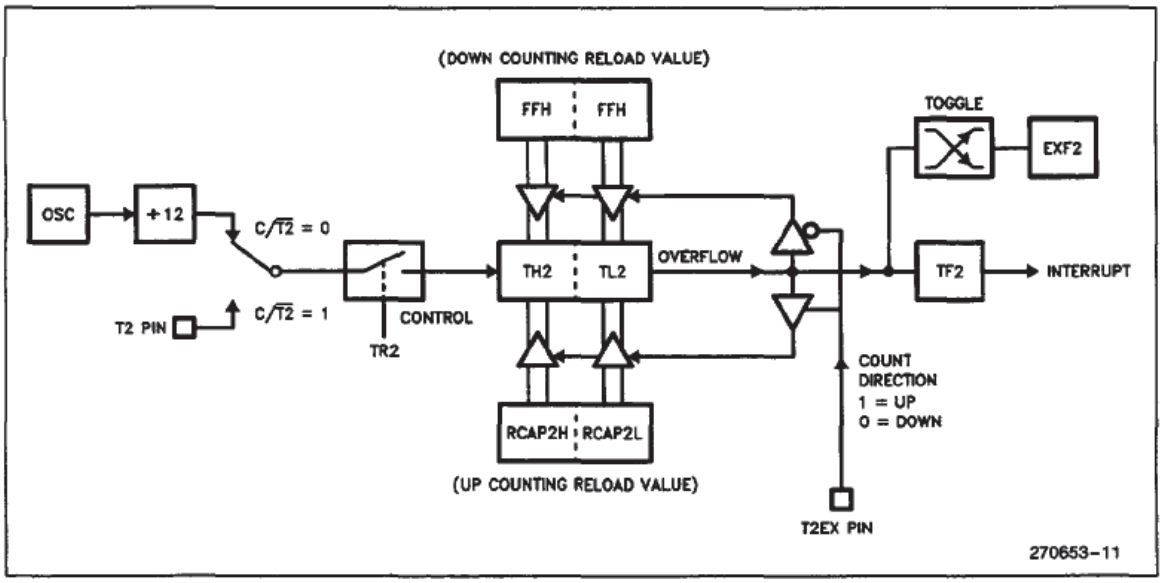



**Figure 14. Timer 2 Auto Reload Mode (DCEN = 1)**

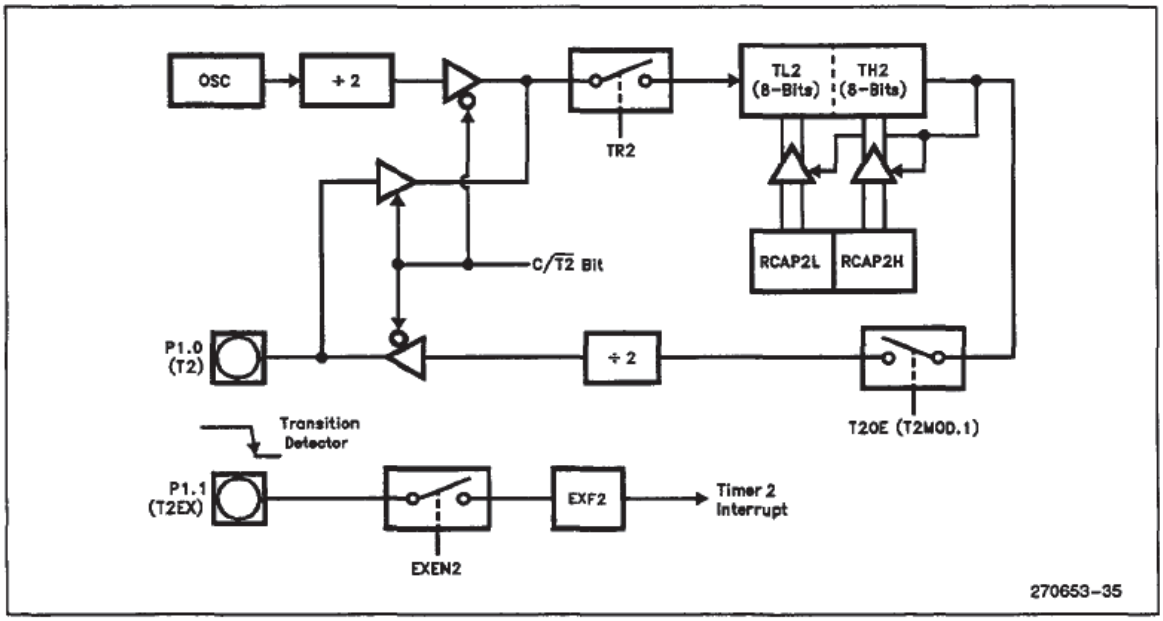



**Figure 15. Timer 2 in Clock-Out Mode**

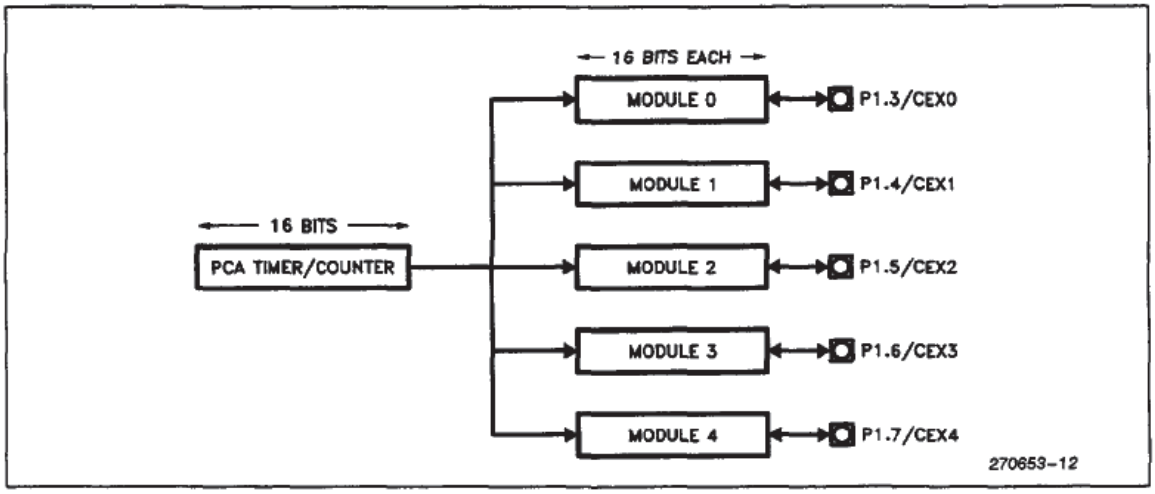



**Figure 15a. Programmable Counter Array**

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

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

## 6.1 PCA 16-Bit Timer/Counter

The PCA has a free-running 16-bit timer/counter consisting of registers CH and CL (the high and low bytes of the count value). These two registers can be read or written to at any time. Figure 16 shows a block diagram of this timer. The clock input can be selected from the following four modes:

- Oscillator frequency ÷ 12
  The CL register is incremented at S5P2 of every machine cycle. With a 16 MHz crystal, the timer increments every 750 nanoseconds.
- Oscillator frequency ÷ 4
  The CL register is incremented at S1P2, S3P2 and S5P2 of every machine cycle. With a 16 MHz crystal, the timer increments every 250 nanoseconds.
- Timer 0 overflows
  The CL register is incremented at S5P2 of the machine cycle when Timer 0 overflows. This mode allows a programmable input frequency to the PCA.
- External input
  The CL register is incremented at the first one of S1P2, S3P2 and S5P2 after a 1-to-0 transition is detected on the ECI pin (P1.2). P1.2 is sampled at S1P2, S3P2 and S5P2 of every machine cycle. The maximum input frequency in this mode is oscillator frequency ÷ 8.

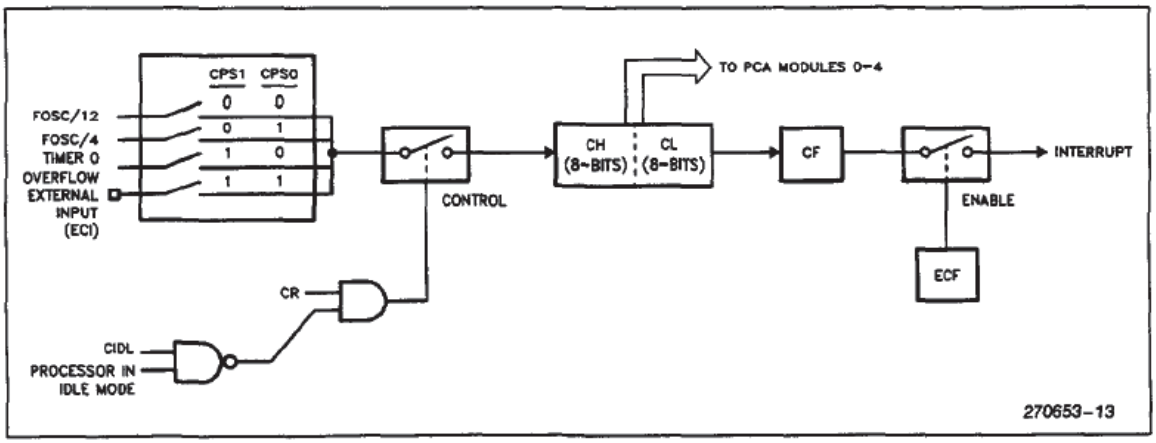



**Figure 16. PCA Timer/Counter**

CH is incremented after two oscillator periods when CL overflows.

The mode register CMOD contains the Count Pulse Select bits (CPS1 and CPS0) to specify the clock input. CMOD is shown in Table 10. This register also contains the ECF bit which enables the PCA counter overflow to generate the PCA interrupt. In addition, the user has the option of turning off the PCA timer during Idle Mode by setting the Counter Idle bit (CIDL). The Watchdog Timer Enable bit (WDTE) will be discussed in a later section.

The CCON register, shown in Table 11, contains two more bits which are associated with the PCA timer/counter. The CF bit gets set by hardware when the counter overflows, and the CR bit is set or cleared to turn the counter on or off. The other five bits in this register are the event flags for the compare/capture modules and will be discussed in the next section.

**Table 10. CMOD: PCA Counter Mode Register**

**CMOD** Address = 0D9H Reset Value = 00XX X000B

Not Bit Addressable

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

| Symbol | Function |
|---|---|
| CIDL | Counter Idle control: CIDL = 0 programs the PCA Counter to continue functioning during idle Mode. CIDL = 1 programs it to be gated off during idle. |
| WDTE | Watchdog Timer Enable: WDTE = 0 disables Watchdog Timer function on PCA Module 4. WDTE = 1 enables it. |
| — | Not implemented, reserved for future use.* |
| CPS1 | PCA Count Pulse Select bit 1. |
| CPS0 | PCA Count Pulse Select bit 0. |

| CPS1 | CPS0 | Selected PCA Input** |
|---|---|---|
| 0 | 0 | Internal clock, Fosc ÷ 12 |
| 0 | 1 | Internal clock, Fosc ÷ 4 |
| 1 | 0 | Timer 0 overflow |
| 1 | 1 | External clock at ECI/P1.2 pin (max. rate = Fosc ÷ 8) |

| | |
|---|---|
| ECF | PCA Enable Counter Overflow interrupt: ECF = 1 enables CF bit in CCON to generate an interrupt. ECF = 0 disables that function of CF. |

**NOTE:**
*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.
**Fosc = oscillator frequency

**Table 11. CCON: PCA Counter Control Register**

**CCON** Address = 0D8H — Reset Value = 00X0 0000B

Bit Addressable

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

| Symbol | Function |
|---|---|
| CF | PCA Counter Overflow flag. Set by hardware when the counter rolls over. CF flags an interrupt if bit ECF in CMOD is set. CF may be set by either hardware or software but can only be cleared by software. |
| CR | PCA Counter Run control bit. Set by software to turn the PCA counter on. Must be cleared by software to turn the PCA counter off. |
| — | Not implemented, reserved for future use*. |
| CCF4 | PCA Module 4 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software. |
| CCF3 | PCA Module 3 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software. |
| CCF2 | PCA Module 2 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software. |
| CCF1 | PCA Module 1 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software. |
| CCF0 | PCA Module 0 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software. |

***NOTE:**
User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

## 6.2 Capture/Compare Modules

Each of the five compare/capture modules has six possible functions it can perform:

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

In addition, module 4 can be used as a Watchdog Timer. The modules can be programmed in any combination of the different modes.

Each module has a mode register called CCAPMn (n = 0, 1, 2, 3, or 4) to select which function it will perform. The CCAPMn register is shown in Table 12. Note the ECCFn bit which enables the PCA interrupt when a module's event flag is set. The event flags (CCFn) are located in the CCON register and get set when a capture event, software timer, or high speed output event occurs for a given module.

Table 13 shows the combinations of bits in the CCAPMn register that are valid and have a defined function. Invalid combinations will produce undefined results.

Each module also has a pair of 8-bit compare/capture registers (CCAPnH and CCAPnL) associated with it. These registers store the time when a capture event occurred or when a compare event should occur. For the PWM mode, the high byte regiser CCAPnH controls the duty cycle of the waveform.

The next five sections describe each of the compare/capture modes in detail.

**Table 12. CCAPMn: PCA Modules Compare/Capture Registers**

**CCAPMn** Address (n = 0–4): CCAPM0 0DAH, CCAPM1 0DBH, CCAPM2 0DCH, CCAPM3 0DDH, CCAPM4 0DEH — Reset Value = X000 0000B

Not Bit Addressable

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

**Symbol Function**

| | |
|---|---|
| — | Not implemented, reserved for future use*. |
| ECOMn | Enable Comparator. ECOMn = 1 enables the comparator function. |
| CAPPn | Capture Positive, CAPPn = 1 enables positive edge capture. |
| CAPNn | Capture Negative, CAPNn = 1 enables negative edge capture. |
| MATn | Match. When MATn = 1, a match of the PCA counter with this module's compare/capture register causes the CCFn bit in CCON to be set, flagging an interrupt. |
| TOGn | Toggle. When TOGn = 1, a match of the PCA counter with this module's compare/capture register causes the CEXn pin to toggle. |
| PWMn | Pulse Width Modulation Mode. PWMn = 1 enables the CEXn pin to be used as a pulse width modulated output. |
| ECCFn | Enable CCF interrupt. Enables compare/capture flag CCFn in the CCON register to generate an interrupt. |

**NOTE:**
*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

***Table 13. PCA Module Modes (CCAPMn Register)***

| — | ECOMn | CAPPn | CAPNn | MATn | TOGn | PWMn | ECCFn | Module Function |
|---|---|---|---|---|---|---|---|---|
| X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | No operation |
| X | X | 1 | 0 | 0 | 0 | 0 | X | 16-bit capture by a postive-edge trigger on CEXn |
| X | X | 0 | 1 | 0 | 0 | 0 | X | 16-bit capture by a negative-edge trigger on CEXn |
| X | X | 1 | 1 | 0 | 0 | 0 | X | 16-bit capture by a transition on CEXn |
| X | 1 | 0 | 0 | 1 | 0 | 0 | X | 16-bit Software Timer |
| X | 1 | 0 | 0 | 1 | 1 | 0 | X | 16-bit High Speed Output |
| X | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 8-bit PWM |
| X | 1 | 0 | 0 | 1 | x | 0 | x | Watchdog Timer |

X = Don't Care

## 6.3 16-Bit Capture Mode

Both positive and negative transitions can trigger a capture with the PCA. This gives the PCA the flexibility to measure periods, pulse widths, duty cycles, and phase differences on up to five separate inputs. Setting the CAPPn and/or CAPNn in the CCAPMn mode register select the input trigger—positive and/or negative transition—for module n. Refer to Figure 17.

The external input pins CEX0 through CEX4 are sampled for a transition. When a valid transition is detected (positive and/or negative edge), hardware loads the 16-bit value of the PCA timer (CH, CL) into the module's capture registers (CCAPnH, CCAPnL). The resulting value in the capture registers reflects the PCA timer value at the time a transition was detected on the CEXn pin.

Upon a capture, the module's event flag (CCFn) in CCON is set, and an interrupt is flagged if the ECCFn bit in the mode register CCAPMn is set. The PCA interrupt will then be generated if it is enabled. Since the hardware does not clear an event flag when the interrupt is vectored to, the flag must be cleared in software. In the interrupt service routine, the 16-bit capture value must be saved in RAM before the next capture event occurs. A subsequent capture on the same CEXn pin will write over the first capture value in CCAPnH and CCAPnL.

## 6.4 16-Bit Software Timer Mode

In the compare mode, the 16-bit value of the PCA timer is compared with a 16-bit value pre-loaded in the module's compare registers (CCAPnH, CCAPnL). The comparison occurs three times per machine cycle in order to recognize the fastest possible clock input (i.e. ¼ x oscillator frequency). Setting the ECOMn bit in the mode register CCAPMn enables the comparator function as shown in Figure 18.

For the Software Timer mode, the MATn bit also needs to be set. When a match occurs between the PCA timer and the compare registers, a match signal is generated and the module's event flag (CCFn) is set. An interrupt is then flagged if the ECCFn bit is set. The PCA interrupt is generated only if it has been properly enabled. Software must clear the event flag before the next interrupt will be flagged.

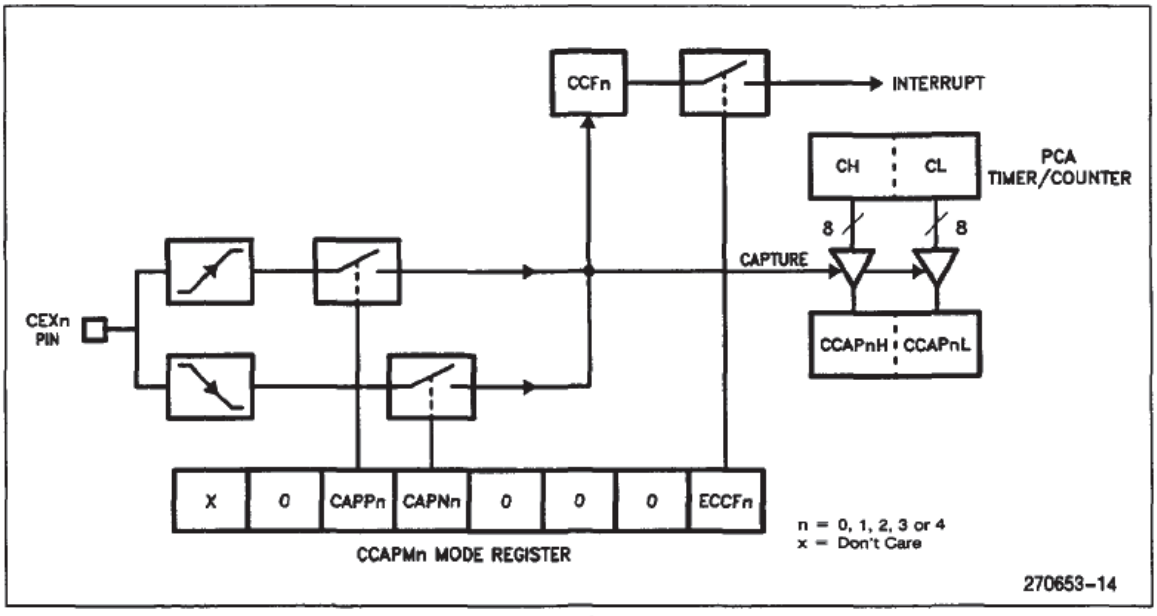



**Figure 17. PCA 16-Bit Capture Mode**

During the interrupt routine, a new 16-bit compare value can be written to the compare registers (CCAPnH and CCAPnL). *Notice, however, that a write to CCAPnL clears the ECOMn bit which temporarily disables the comparator function while these registers are being updated so an invalid match does not occur.* A write to CCAPnH sets the ECOMn bit and re-enables the comparator. For this reason, user software should write to CCAPnL first, then CCAPnH.

## 6.5 High Speed Output Mode

The High Speed Output (HSO) mode toggles a CEXn pin when a match occurs between the PCA timer and a pre-loaded value in a module's compare registers. For this mode, the TOGn bit needs to be set in addition to the ECOMn and MATn bits as seen in Figure 18. By setting or clearing the pin in software, the user can select whether the CEXn pin will change from a logical 0 to a logical 1 or vice versa. The user also has the option of flagging an interrupt when a match event occurs by setting the ECCFn bit.

The HSO mode is more accurate than toggling port pins in software because the toggle occurs *before* branching to an interrupt. That is, interrupt latency will not effect the accuracy of the output. If the user does not change the compare registers in an interrupt routine, the next toggle will occur when the PCA timer rolls over and matches the last compare value.

## 6.6 Watchdog Timer Mode

A Watchdog Timer is a circuit that automatically invokes a reset unless the system being watched sends regular hold-off signals to the Watchdog. These circuits are used in applications that are subject to electrical noise, power glitches, electrostatic discharges, etc., or where high reliability is required.

The Watchdog Timer function is only available on PCA module 4. In this mode, every time the count in the PCA timer matches the value stored in module 4's compare registers, an internal reset is generated. (See Figure 19.) The bit that selects this mode is WDTE in the CMOD register. Module 4 must be set up in either compare mode as a Software Timer or High Speed Output.

When the PCA Watchdog Timer times out, it resets the chip just like a hardware reset, except that it does not drive the reset pin high.

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

(1) periodically change the compare value so it will never match the PCA timer,

(2) periodically change the PCA timer value so it will never match the compare value,

(3) disable the Watchdog by clearing the WDTE bit before a match occurs and then later re-enable it.

The first two options are more reliable because the Watchdog Timer is never disabled as in option #3. The second option is not recommended if other PCA modules are being used since this timer is the time base for all five modules. Thus, in most applications the first solution is the best option.

If a Watchdog Timer is not needed, module 4 can still be used in other modes.

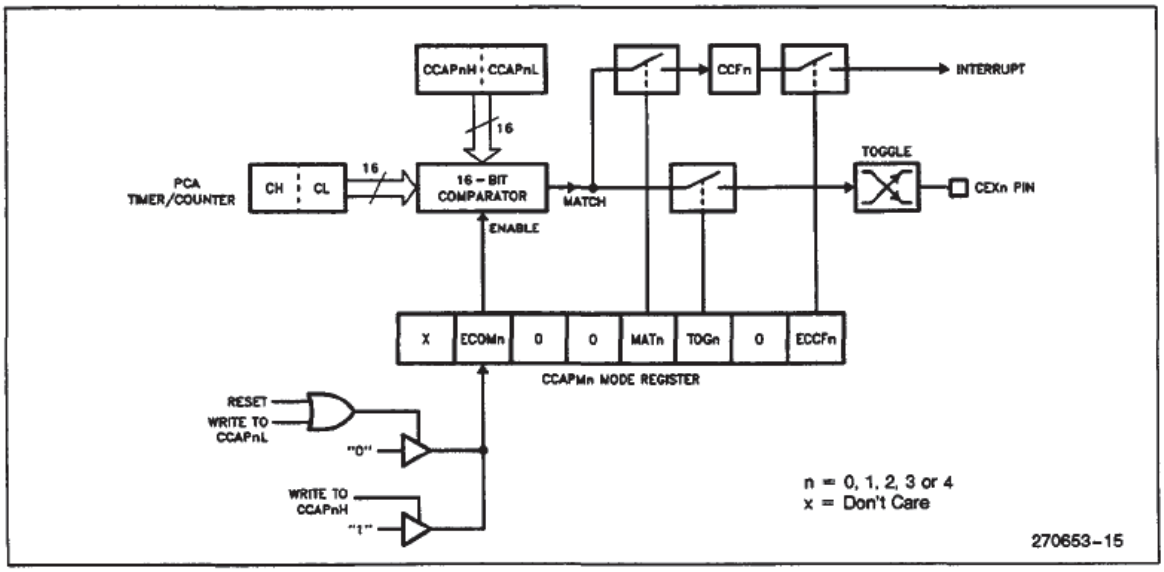



**Figure 18. PCA 16-Bit Comparator Mode: Software Timer and High Speed Output**

## 6.7 Pulse Width Modulator Mode

Any or all of the five PCA modules can be programmed to be a Pulse Width Modulator. The PWM output can be used to convert digital data to an analog signal by simple external circuitry. The frequency of the PWM depends on the clock sources for the PCA timer. With a 16 MHz crystal the maximum frequency of the PWM waveform is 15.6 KHz.

The PCA generates 8-bit PWMs by comparing the low byte of the PCA timer (CL) with the low byte of the module's compare registers (CCAPnL). Refer to Figure 20. When CL < CCAPnL the output is low. When CL ≥ CCAPnL the output is high. The value in CCAPnL controls the duty cycle of the waveform. To change the value in CCAPnL without output glitches, the user must write to the high byte register (CCAPnH). This value is then shifted by hardware into CCAPnL when CL rolls over from 0FFH to 00H which corresponds to the next period of the output.

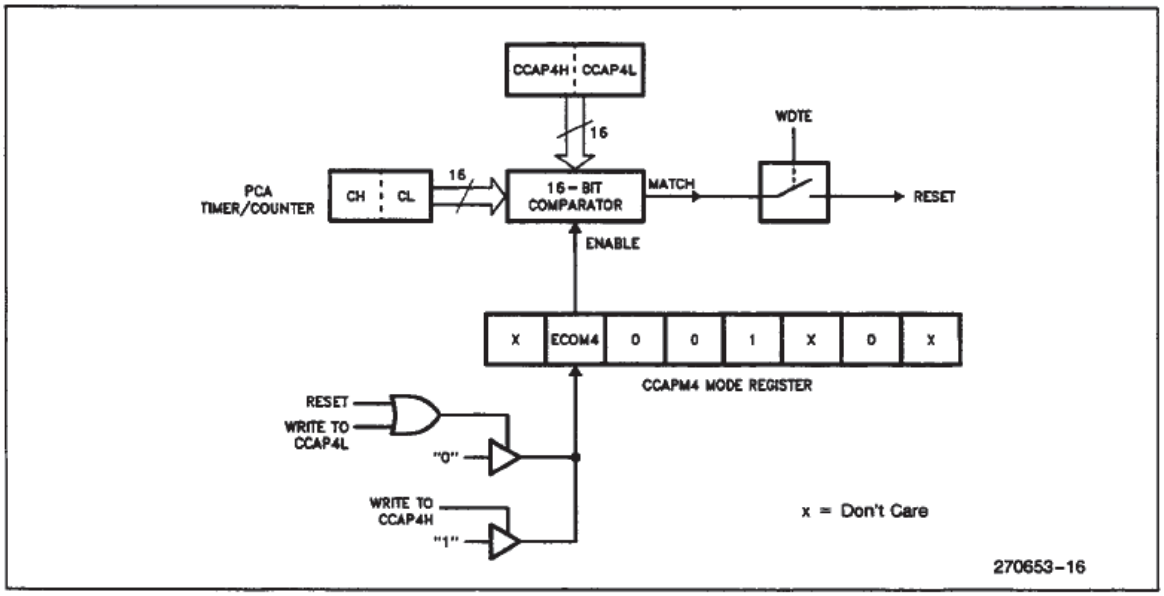



**Figure 19. Watchdog Timer Mode**

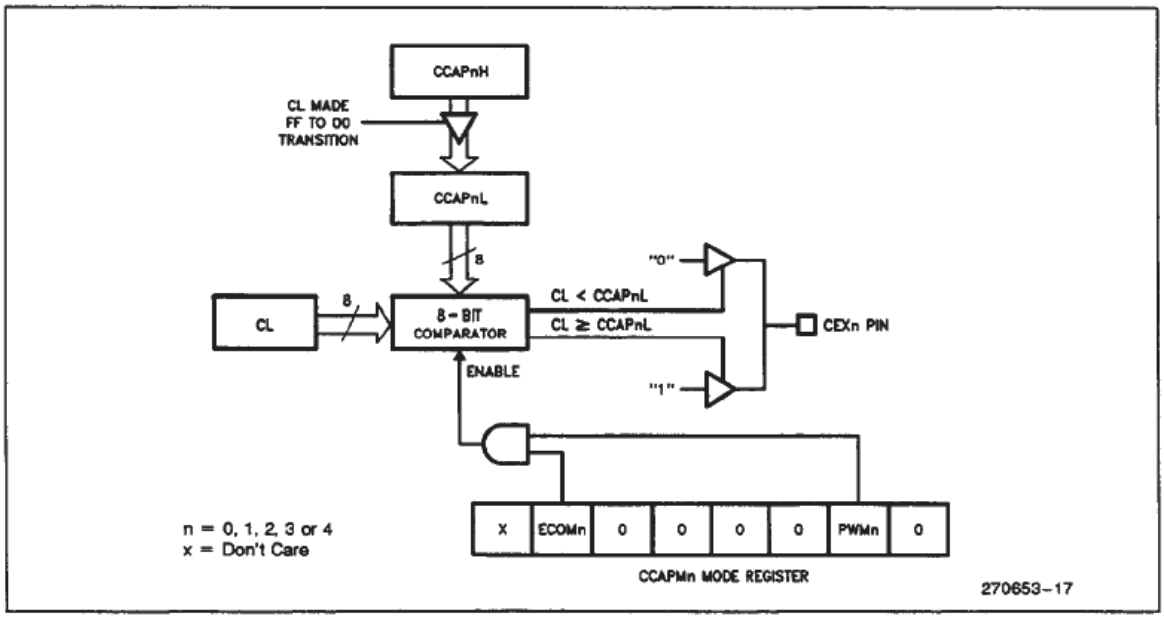



**Figure 20. PCA 8-Bit PWM Mode**

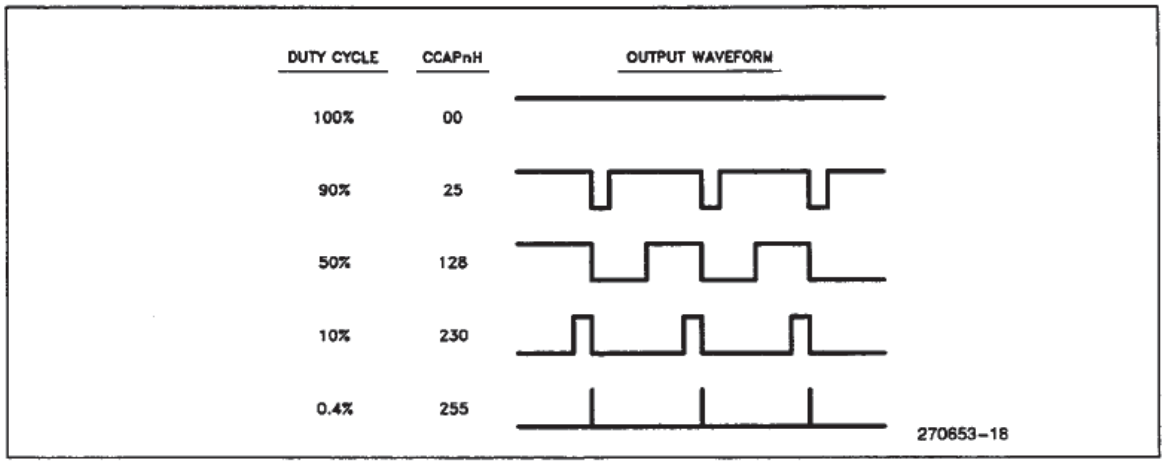



**Figure 21. CCAPnH Varies Duty Cycle**

CCAPnH can contain any integer from 0 to 255 to vary the duty cycle from a 100% to 0.4% (see Figure 21).

## 7.0 SERIAL INTERFACE

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register. (However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost). The serial port receive and transmit registers are both accessed through Special Function Register SBUF. Actually, SBUF is two separate registers, a transmit buffer and a receive buffer. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

The serial port control and status register is the Special Function Register SCON, shown in Table 14. This register contains the mode selection bits (SM0 and SM1); the SM2 bit for the multiprocessor modes (see Multiprocessor Communications section); the Receive Enable bit (REN); the 9th data bit for transmit and receive (TB8 and RB8); and the serial port interrupt bits (TI and RI).

The serial port can operate in 4 modes:

**Mode 0:** Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at 1/12 the oscillator frequency.

**Mode 1:** 10 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in Special Function Register SCON. The baud rate is variable.

**Mode 2:** 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). Refer to Figure 22. On Transmit, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P in the PSW) could be moved into TB8. On receive, the 9th data bit goes into RB8 in SCON, while the stop bit is ignored. (The validity of the stop bit can be checked with Framing Error Detection.) The baud rate is programmable to either $1/32$ or $1/64$ the oscillator frequency.

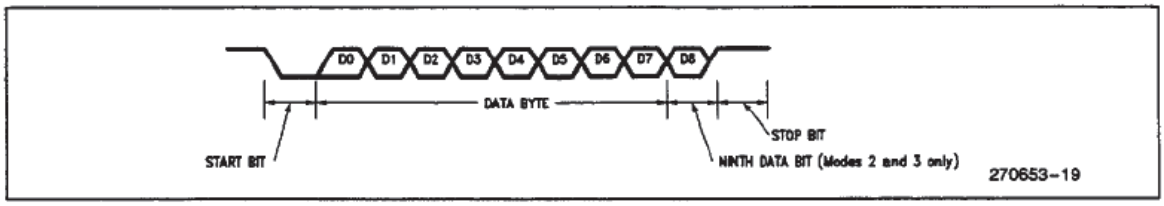



**Figure 22. Data Frame: Modes 1, 2 and 3**

**Mode 3:** 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except the baud rate. The baud rate in Mode 3 is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1. For more detailed information on each serial port mode, refer to the "Hardware Description of the 8051, 8052, and 80C51."

## 7.1 Framing Error Detection

Framing Error Detection allows the serial port to check for valid stop bits in modes 1, 2, or 3. A missing stop bit can be caused, for example, by noise on the serial lines, or transmission by two CPUs simultaneously.

If a stop bit is missing, a Framing Error bit FE is set. The FE bit can be checked in software after each reception to detect communication errors. Once set, the FE bit must be cleared in software. A valid stop bit will not clear FE.

The FE bit is located in SCON and shares the same bit address as SM0. Control bit SMOD0 in the PCON register (location PCON.6) determines whether the SM0 or FE bit is accessed. If SMOD0 = 0, then accesses to SCON.7 are to SM0. If SMOD0 = 1, then accesses to SCON.7 are to FE.

## 7.2 Multiprocessor Communications

Modes 2 and 3 provide a 9-bit mode to facilitate multiprocessor comunication. The 9th bit allows the controller to distinguish between address and data bytes. The 9th bit is set to 1 for address bytes and set to 0 for data bytes. When receiving, the 9th bit goes into RB8 in SCON. When transmitting, TB8 is set or cleared in software.

The serial port can be programmed such that when the stop bit is received the serial port interrupt will be activated only if the received byte is an address byte (RB8 = 1). This feature is enabled by setting the SM2 bit in SCON. A way to use this feature in multiprocessor systems is as follows.

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. Remember, an address byte has its 9th bit set to 1, whereas a data byte has its 9th bit set to 0. All the slave processors should have their SM2 bits set to 1 so they will only be interrupted by an address byte. In fact, the C51FX has an Automatic Address Recognition feature which allows only the addressed slave to be interrupted. That is, the address comparison occurs in hardware, not software. (On the 8051 serial port, an address byte interrupts all slaves for an address comparison.)

The addressed slave's software then clears its SM2 bit and prepares to receive the data bytes that will be coming. The other slaves are unaffected by these data bytes. They are still waiting to be addressed since their SM2 bits are all set.

## 7.3 Automatic Address Recognition

Automatic Address Recognition reduces the CPU time required to service the serial port. Since the CPU is only interrupted when it receives its own address, the software overhead to compare addresses is eliminated. With this feature enabled in one of the 9-bit modes, the Receive Interrupt (RI) flag will only get set when the received byte corresponds to either a Given or Broadcast address.

The feature works the same way in the 8-bit mode (Mode 1) as in the 9-bit modes, except that the stop bit takes the place of the 9th data bit. If SM2 is set, the RI flag is set only if the received byte matches the Given or Broadcast Address and is terminated by a valid stop bit. Setting the SM2 bit has no effect in Mode 0.

The master can selectively communicate with groups of slaves by using the Given Address. Addressing all slaves at once is possible with the Broadcast Address. These addresses are defined for each slave by two Special Function Registers: SADDR and SADEN.

A slave's individual address is specified in SADDR. SADEN is a mask byte that defines don't-cares to form the Given Address. These don't-cares allow flexibility in the user-defined protocol to address one or more slaves at a time. The following is an example of how the user could define Given Addresses to selectively address different slaves.

**Slave 1:**

| | | | |
|---|---|---|---|
| SADDR | = | 1111 | 0001 |
| SADEN | = | 1111 | 1010 |
| GIVEN | = | 1111 | 0X0X |

**Slave 2:**

| | | | |
|---|---|---|---|
| SADDR | = | 1111 | 0011 |
| SADEN | = | 1111 | 1001 |
| GIVEN | = | 1111 | 0XX1 |

**Table 14. SCON: Serial Port Control Register**

**SCON** Address = 98H Reset Value = 0000 0000B

Bit Addressable

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

Bit: (SMOD0 = 0/1)*

| Symbol | Function |
|---|---|
| FE | Framing Error bit. This bit is set by the receiver when an invalid stop bit is detected. The FE bit is not cleared by valid frames but should be cleared by software. The SMOD0* bit must be set to enable access to the FE bit. |
| SM0 | Serial Port Mode Bit 0, (SMOD0 must = 0 to access bit SM0) |
| SM1 | Serial Port Mode Bit 1 |

| SM0 | SM1 | Mode | Description | Baud Rate** |
|---|---|---|---|---|
| 0 | 0 | 0 | shift register | $F_{OSC}/12$ |
| 0 | 1 | 1 | 8-bit UART | variable |
| 1 | 0 | 2 | 9-bit UART | $F_{OSC}/64$ or $F_{OSC}/32$ |
| 1 | 1 | 3 | 9-bit UART | variable |

| | |
|---|---|
| SM2 | Enables the Automatic Address Recognition feature in Modes 2 or 3. If SM2 = 1 then RI will not be set unless the received 9th data bit (RB8) is 1, indicating an address, and the received byte is a Given or Broadcast Address. In Mode 1, if SM2 = 1 then RI will not be activated unless a valid stop bit was received, and the received byte is a Given or Broadcast Address. In Mode 0, SM2 should be 0. |
| REN | Enables serial reception. Set by software to enable reception. Clear by software to disable reception. |
| TB8 | The 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired. |
| RB8 | In modes 2 and 3, the 9th data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used. |
| TI | Transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software. |
| RI | Receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software. |

**NOTE:**
*SMOD0 is located at PCON6.
**$F_{OSC}$ = oscillator frequency

The SADEN byte are selected such that each slave can be addressed separately. Notice that bit 1 (LSB) is a don't-care for Slave 1's Given Address, but bit 1 = 1 for Slave 2. Thus, to selectively communicate with just Slave 1 the master must send an address with bit 1 = 0 (e.g. 1111 0000).

Similarly, bit 2 = 0 for Slave 1, but is a don't-care for Slave 2. Now to communicate with just Slave 2 an address with bit 2 = 1 must be used (e.g. 1111 0111).

Finally, for a master to communicate with both slaves at once the address must have bit 1 = 1 and bit 2 = 0. Notice, however, that bit 3 is a don't-care for both slaves. This allows two different addresses to select both slaves (1111 0001 or 1111 0101). If a third slave was added that required its bit 3 = 0, then the latter address could be used to communicate with Slave 1 and 2 but not Slave 3.

The master can also communicate with all slaves at once with the Broadcast Address. It is formed from the logical OR of the SADDR and SADEN registers with zeros defined as don't-cares. The don't-cares also allow

flexibility in defining the Broadcast Address, but in most applications a Broadcast Address will be 0FFH.

SADDR and SADEN are located at address A9H and B9H, respectively. On reset, the SADDR and SADEN registers are initialized to 00H which defines the Given and Broadcast Addresses as XXXX XXXX (all don't-cares). This assures the C51FX serial port to be backwards compatibility with other MCS®-51 products which do not implement Automatic Addressing.

## 7.4 Baud Rates

The baud rate in Mode 0 is fixed:

$$\text{Mode 0 Baud Rate} = \frac{\text{Oscillator Frequency}}{12}$$

The baud rate in Mode 2 depends on the value of bit SMOD1 in Special Function Register PCON. If SMOD1 = 0 (which is the value on reset), the baud rate is $1/64$ the oscillator frequency. If SMOD1 = 1, the baud rate is $1/32$ the oscillator frequency.

$$\text{Mode 2 Baud Rate} = 2^{\text{SMOD1}} \times \frac{\text{Oscillator Frequency}}{64}$$

The baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate, or by Timer 2 overflow rate, or by both (one for transmit and the other for receive).

## 7.5 Using Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD1 as follows:

$$\text{Modes 1 and 3 Baud Rate} = 2^{\text{SMOD1}} \times \frac{\text{Timer 1 Overflow Rate}}{32}$$

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either "timer" or "counter" operation, and in any of its 3 running modes. In most applications, it is configured for "timer" operation in the auto-reload mode (high nibble of TMOD = 0010B). In this case, the baud rate is given by the formula:

$$\text{Modes 1 and 3 Baud Rate} = \frac{2^{\text{SMOD1}} \times \text{Oscillator Frequency}}{32 \times 12 \times [256 - (\text{TH1})]}$$

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the Timer 1 interrupt to do a 16-bit software reload.

Table 15 lists various commonly used baud rates and how they can be obtained from Timer 1.

## 7.6 Using Timer 2 to Generate Baud Rates

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Table 7). Note that the baud rates for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 23.

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

**Table 15. Timer 1 Generated Commonly Used Baud Rates**

| Baud Rate | $f_{OSC}$ | SMOD | Timer 1 | | |
|---|---|---|---|---|---|
| | | | C/$\overline{T}$ | Mode | Reload Value |
| Mode 0 Max: 1 MHz | 12 MHz | X | X | X | X |
| Mode 2 Max: 375K | 12 MHz | 1 | X | X | X |
| Modes 1, 3: 62.5K | 12 MHz | 1 | 0 | 2 | FFH |
| 19.2K | 11.059 MHz | 1 | 0 | 2 | FDH |
| 9.6K | 11.059 MHz | 0 | 0 | 2 | FDH |
| 4.8K | 11.059 MHz | 0 | 0 | 2 | FAH |
| 2.4K | 11.059 MHz | 0 | 0 | 2 | F4H |
| 1.2K | 11.059 MHz | 0 | 0 | 2 | E8H |
| 137.5 | 11.986 MHz | 0 | 0 | 2 | 1DH |
| 110 | 6 MHz | 0 | 0 | 2 | 72H |
| 110 | 12 MHz | 0 | 0 | 1 | FEEBH |

The baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate as follows:

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either "timer" or "counter" operation. In most applications, it is configured for "timer" operation ($C/\overline{T2} = 0$). The "Timer" operation is different for Timer 2 when it's being used as a baud rate generator. Normally, as a timer, it increments every machine cycle (1/12 the oscillator frequency). As a baud rate generator, however, it increments every state time (½ the oscillator frequency). The baud rate formula is given below:

$$\text{Modes 1 and 3 Baud Rate} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - (\text{RCAP2H, RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 23. This figure is valid only if RCLK and/or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Therefore, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

It should be noted that when Timer 2 is running (TR2 = 1) in "timer" function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the Timer is being incremented every state time, and the results of a read or write may not be accurate. The RCAP2 registers may be read, but shouldn't be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

Table 16 lists commonly used baud rates and how they can be obtained from Timer 2.

**Table 16. Timer 2 Generated Commonly Used Baud Rates**

| Baud Rate | Osc Freq | Timer 2 | |
|---|---|---|---|
| | | RCAP2H | RCAP2L |
| 375K | 12 MHz | FF | FF |
| 9.6K | 12 MHz | FF | D9 |
| 4.8K | 12 MHz | FF | B2 |
| 2.4K | 12 MHz | FF | 64 |
| 1.2K | 12 MHz | FE | C8 |
| 300 | 12 MHz | FB | 1E |
| 110 | 12 MHz | F2 | AF |
| 300 | 6 MHz | FD | 8F |
| 110 | 6 MHz | F9 | 57 |

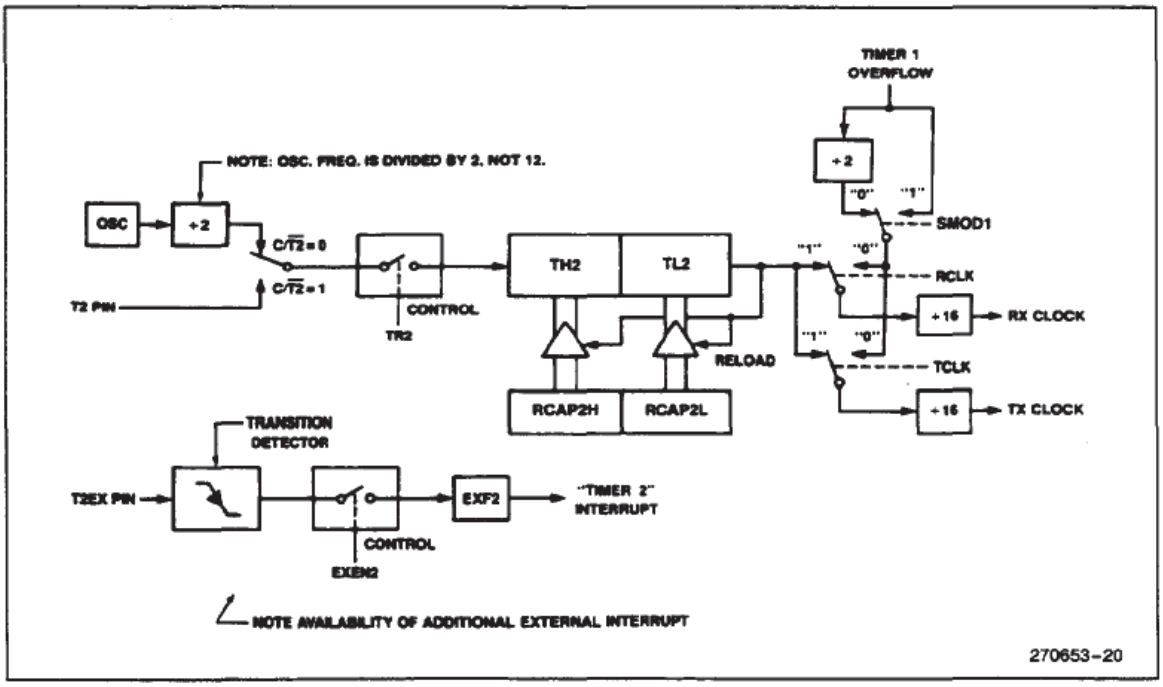


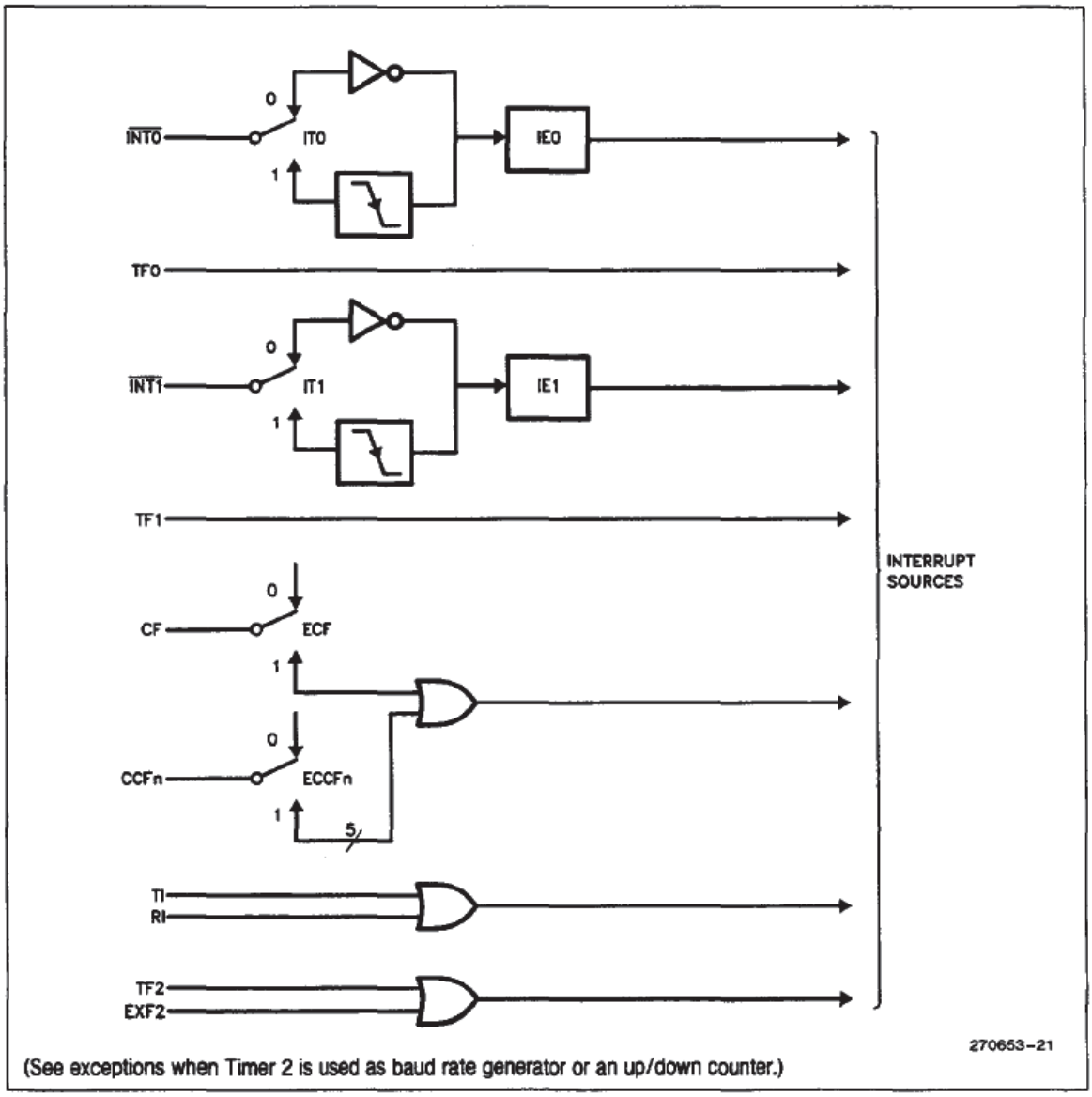

**Figure 23. Timer 2 in Baud Rate Generator Mode**

## 8.0 INTERRUPTS

The C51FX has a total of 7 interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1, and 2), the PCA interrupt, and the serial port interrupt. These interrupts are all shown in Figure 24.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled in software.

Each of these interrupts will be briefly described followed by a discussion of the interrupt enable bits and the interrupt priority levels.




(See exceptions when Timer 2 is used as baud rate generator or an up/down counter.)

**Figure 24. Interrupt Sources**

## 8.1 External Interrupts

External Interrupts $\overline{INT0}$ and $\overline{INT1}$ can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in register TCON. If ITx = 0, external interrupt x is triggered by a detected low at the $\overline{INTx}$ pin. If ITx = 1, external interrupt x is negative edge-triggered. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON. These flags are cleared by hardware when the service routine is vectored to only if the interrupt was transition-activated. If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

Since the external interrupt pins are sampled once each machine cycle, an input high or low should hold for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one cycle, and then hold it low for at least one cycle to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.

If external interrupt $\overline{INT0}$ or $\overline{INT1}$ is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

## 8.2 Timer Interrupts

Timer 0 and Timer 1 Interrupts are generated by TF0 and TF1 in register TCON, which are set by a rollover in their respective Timer/Counter registers (except see Timer 0 in Mode 3). When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

Timer 2 Interrupt is generated by the logical OR of bits TF2 and EXF2 in register T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared in software.

## 8.3 PCA Interrupt

The PCA interrupt is generated by the logical OR of CF, CCF0, CCF1, CCF2, CCF3, and CCF4 in register CCON. None of these flags is cleared by hardware when the service routine is vectored to. Normally the service routine will have to determine which bit flagged the interrupt and clear that bit in software. The PCA interrupt is enabled by bit EC in the Interrupt Enable register (see Table 16). In addition, the CF flag and each of the CCFn flags must also be enabled by bits ECF and ECCFn in registers CMOD and CCAPMn respectively, in order for that flag to be able to cause an interrupt.

## 8.4 Serial Port Interrupt

The serial port interrupt is generated by the logical OR of bits RI and TI in register SCON. Neither of these flags is cleared by hardware when the service routine is vectored to. The service routine will normally have to determine whether it was RI or TI that generated the interrupt, and the bit will have to be cleared in software.

## 8.5 Interrupt Enable

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in the Interrupt Enable (IE) register. (See Table 17.) Note that IE also contains a global disable bit, EA. If EA is set (1), the interrupts are individually enabled or disabled by their corresponding bits in IE. If EA is clear (0), all interrupts are disabled.

## 8.6 Priority Level Structure

Each interrupt source can also be individually programmed to one of two priority levels, by setting or clearing a bit in the Interrupt Priority (IP) register shown in Table 18. A low-priority interrupt can itself be interrupted by a higher priority interrupt, but not by another low-priority interrupt. A high priority interrupt cannot be interrupted by any other interrupt source.

**Table 17. IE: Interrupt Enable Register**

**IE** Address = 0A8H Reset Value = 0000 0000B

Bit Addressable

| | EA | EC | ET2 | ES | ET1 | EX1 | ET0 | EX0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Enable Bit = 1 enables the interrupt.
Enable Bit = 0 disables it.

| Symbol | Function |
|---|---|
| EA | Global disable bit. If EA = 0, all Interrupts are disabled. If EA = 1, each Interrupt can be individually enabled or disabled by setting or clearing its enable bit. |
| EC | PCA interrupt enable bit. |
| ET2 | Timer 2 interrupt enable bit. |
| ES | Serial Port interrupt enable bit. |
| ET1 | Timer 1 interrupt enable bit. |
| EX1 | External interrupt 1 enable bit. |
| ET0 | Timer 0 interrupt enable bit. |
| EX0 | External interrupt 0 enable bit. |

**Table 18. IP: Interrupt Priority Registers**

**IP** **Address = 0B8H** **Reset Value = X000 0000B**

Bit Addressable

| | — | PPC | PT2 | PS | PT1 | PX1 | PT0 | PX0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Priority Bit = 1 assigns high priority
Priority Bit = 0 assigns low priority

| Symbol | Function |
|---|---|
| — | Not implemented, reserved for future use.* |
| PPC | PCA interrupt priority bit. |
| PT2 | Timer 2 interrupt priority bit. |
| PS | Serial Port interrupt priority bit. |
| PT1 | Timer 1 interrupt priority bit. |
| PX1 | External interrupt 1 priority bit. |
| PT0 | Timer 0 interrupt priority bit. |
| PX0 | External interrupt 0 priority bit. |

**NOTE:**
*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence shown in Table 19.

Note that the "priority within level" structure is only used to resolve *simultaneous requests of the same priority level.*

**Table 19. Interrupt Priority within Level Polling Sequence**

| | |
|---|---|
| 1 (Highest) | $\overline{\text{INT0}}$ |
| 2 | Timer 0 |
| 3 | $\overline{\text{INT1}}$ |
| 4 | Timer 1 |
| 5 | PCA |
| 6 | Serial Port |
| 7 (Lowest) | Timer 2 |

### 8XC51FX Interrupt Priority Structure

In the 8XC51FX, a second Interrupt Priority register (IPH) has been added, increasing the number of priority levels to four. Table 20 shows this second register. The added register becomes the MSB of the priority select bits and the existing IP register acts as the LSB. This scheme maintains compatibility with the rest of the MCS-51 family. Table 21 shows the bit values and priority levels associated with each combination.

**Table 21. Priority Level Bit Values**

| Priority Bits | | Interrupt Priority Level |
|---|---|---|
| IPH.x | IP.x | |
| 0 | 0 | Level 0 (Lowest) |
| 0 | 1 | Level 1 |
| 1 | 0 | Level 2 |
| 1 | 1 | Level 3 (Highest) |

### How Interrupts are Handled

The interrupt flags are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. The Timer 2 interrupt cycle is slightly different, as described in the Response Time section. If one of the flags was in a set condition at S5P2 of the preceding 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 equal or higher priority level is already in progress.
2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE or IP registers.

**Table 20. IPH: Interrupt Priority High Register**

**IPH** **Address = 0B7H** **Reset Value = X000 0000**

Not Bit Addressable

| | — | PPCH | PT2H | PSH | PT1H | PX1H | PT0H | PX0H |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| — | Not implemented, reserved for future use. |
| PPCH | PCA interrupt priority high bit. |
| PT2H | Timer 2 interrupt priority high bit. |
| PSH | Serial Port interrupt priority high bit. |
| PT1H | Timer 1 interrupt priority high bit. |
| PX1H | External interrupt 1 priority high bit. |
| PT0H | Timer 0 interrupt priority high bit. |
| PX0H | External interrupt priority high bit. |

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any write to IE or IP, then at least one more instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at S5P2 of the previous machine cycle. If the interrupt flag for a *level-sensitive* external interrupt is active but not being responded to for one of the above conditions and is not *still* active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The polling cycle/LCALL sequence is illustrated in Figure 25.

Note that if an interrupt of a higher priority level goes active prior to S5P2 of the machine cycle labeled C3 in Figure 25, then in accordance with the above rules it will be vectored to during C5 and C6, without any instruction of the lower priority routine having been executed.

Thus the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine. The hardware-generated LCALL pushes the contents of the Program Counter onto 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 22.

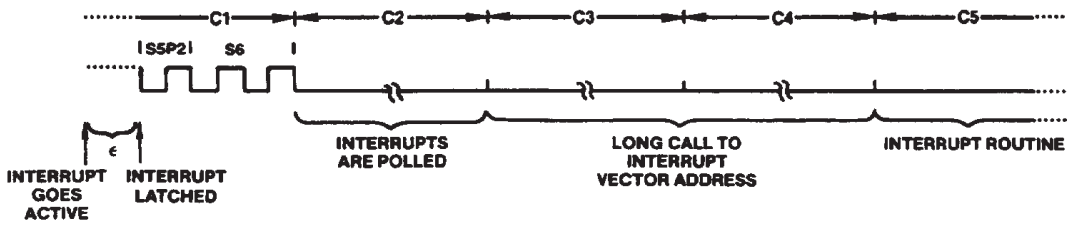
**Table 22. Interrupt Vector Address**

| Interrupt Source | Interrupt Request Bits | Cleared by Hardware | Vector Address |
|---|---|---|---|
| $\overline{\text{INT0}}$ | IE0 | No (level)<br>Yes (trans.) | 0003H |
| TIMER 0 | TF0 | Yes | 000BH |
| $\overline{\text{INT1}}$ | IE1 | No (level)<br>Yes (trans.) | 0013H |
| TIMER 1 | TF1 | Yes | 001BH |
| SERIAL PORT | RI, TI | No | 0023H |
| TIMER 2 | TF2, EXF2 | No | 002BH |
| PCA | CF, CCFn<br>(n = 0-4) | No | 0033H |

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stack and reloads the Program Counter. Execution of the interrupted program continues from where it left off.

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking interrupt was still in progress.

Note that the starting addresses of consecutive interrupt service routines are only 8 bytes apart. That means if consecutive interrupts are being used (IE0 and TF0, for example, or TF0 and IE1), and if the first interrupt routine is more than 7 bytes long, then that routine will have to execute a jump to some other memory location where the service routine can be completed without overlapping the starting address of the next interrupt routine.




This is the fastest possible response when C2 is the final cycle of an instruction other than RETI or write IE or IP.

**Figure 25. Interrupt Response Timing Diagram**

## 8.7 Response Time

The $\overline{INT0}$ and $\overline{INT1}$ levels are inverted and latched into the Interrupt Flags IE0 and IE1 at S5P2 of every machine cycle. Similarly, the Timer 2 flag EXF2 and the Serial Port flags RI and TI are set at S5P2. The values are not actually polled by the circuitry until the next machine cycle.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag TF2 is set at S2P2 and is polled in the same cycle in which the timer overflows.

If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapses between activation of an external interrupt request and the beginning of execution of the service routine's first instruction. Figure 25 shows interrupt response timing.

A longer response time would result if the request is blocked by one of the 3 previously listed conditions. If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is not in its final cycle, the additional wait time cannot be more than 3 cycles, since the longest instructions (MUL and DIV) are only 4 cycles long, and if the instruction in progress is RETI or write to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one or more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV).

Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 9 cycles.

## 9.0 RESET

The reset input is the RST pin, which has a Schmitt Trigger input. 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 generating an internal reset, with the timing shown in Figure 26.

The external reset signal is asynchronous to the internal clock. The RST pin is sampled during State 5 Phase 2 of every machine cycle. ALE and PSEN will maintain their current activities for 19 oscillator periods after a logic 1 has been sampled at the RST pin; that is, for 19 to 31 oscillator periods after the external reset signal has been applied to the RST pin. The port pins are driven to their reset state as soon as a valid high is detected on the RST pin, regardless of whether the clock is running.

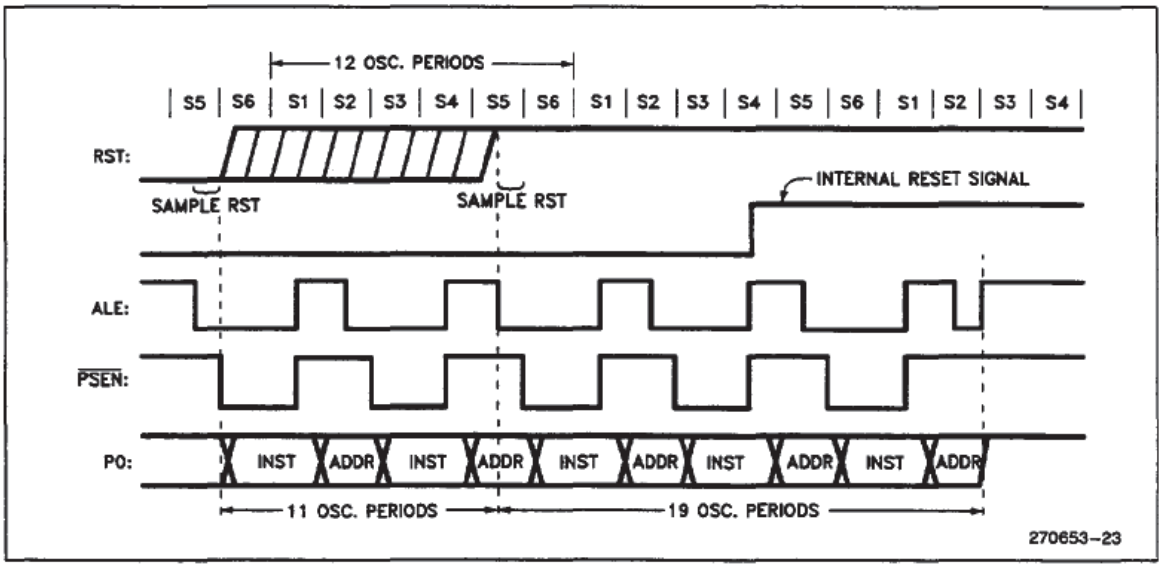



**Figure 26. Reset Timing**

While the RST pin is high, the port pins, ALE and $\overline{PSEN}$ are weakly pulled high. After RST is pulled low, it will take 1 to 2 machine cycles for ALE and $\overline{PSEN}$ to start clocking. For this reason, other devices can not be synchronized to the internal timings of the 8XC51FX.

Driving the ALE and $\overline{PSEN}$ pins to 0 while reset is active could cause the device to go into an indeterminate state.

The internal reset algorithm redefines all the SFRs. Table 1 lists the SFRs and their reset values. The internal RAM is not affected by reset. On power up the RAM content is indeterminate.

## 9.1 Power-On Reset

For CHMOS devices, when VCC is turned on, an automatic reset can be obtained by connecting the RST pin to VCC through a 1 $\mu$F capacitor (Figure 27). The CHMOS devices do not require an external resistor like the HMOS devices because they have an internal pull-down on the RST pin.

When power is turned on, the circuit holds the RST pin high for an amount of time that depends on the capacitor value and the rate at which it charges. To ensure a valid reset the RST pin must be held high long enough to allow the oscillator to start up plus two machine cycles.

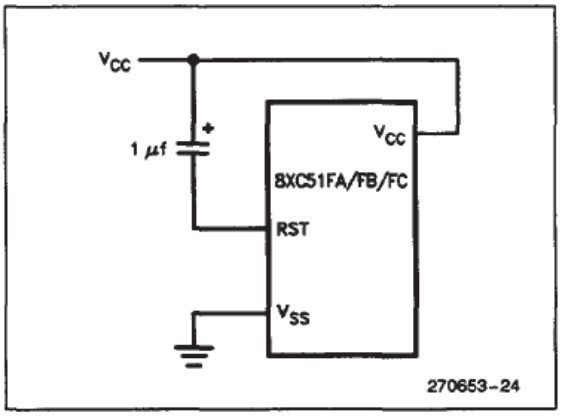



**Figure 27. Power on Reset Circuitry**

On power up, $V_{CC}$ should rise within approximately ten milliseconds. The oscillator start-up time will depend on the oscillator frequency. For a 10 MHz crystal, the start-up time is typically 1 msec. For a 1 MHz crystal, the start-up time is typically 10 msec.

With the given circuit, reducing $V_{CC}$ quickly to 0 causes the RST pin voltage to momentarily fall below 0V. However, this voltage is internally limited and will not harm the device.

*Note that the port pins will be in a random state until the oscillator has started and the internal reset algorithm has written 1s to them.*

Powering up the device without a valid reset could cause the CPU to start executing instructions from an indeterminate location. This is because the SFRs, specifically the Program Counter, may not get properly initialized.

# 10.0 POWER-SAVING MODES OF OPERATION

For applications where power consumption is critical, the C51FX provides two power reducing modes of operation: Idle and Power Down. The input through which backup power is supplied during these operations is $V_{CC}$. Figure 28 shows the internal circuitry which implements these features. In the Idle mode (IDL = 1), the oscillator continues to run and the Interrupt, Serial Port, PCA, and Timer blocks continue to be clocked, but the clock signal is gated off to the CPU. In Power Down (PD = 1), the oscillator is frozen. The Idle and Power Down modes are activated by setting bits in Special Function Register PCON (Table 23).

## 10.1 Idle Mode

An instruction that sets PCON.0 causes that to be the last instruction executed before going into the Idle mode. In the Idle mode, the internal clock signal is gated off to the CPU, but not to the Interrupt, Timer, and Serial Port functions. The PCA can be programmed either to pause or continue operating during Idle (refer to the PCA section for more details). The CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical states they had at the time Idle was activated. ALE and $\overline{PSEN}$ hold at logic high levels.

There are two ways to terminate the Idle Mode. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that put the device into Idle.

The flag bits (GF0 and GF1) can be used to give an indication if an interrupt occurred during normal operation or during Idle. For example, an instruction that activates Idle can also set one or both flag bits. When Idle is terminated by an interrupt, the interrupt service routine can examine the flag bits.

The other way of terminating the Idle mode is with a hardware reset. Since the clock oscillator is still running, the hardware reset needs to be held active for only two machine cycles (24 oscillator periods) to complete the reset.

The signal at the RST pin clears the IDL bit directly and asynchronously. At this time the CPU resumes program execution from where it left off; that is, at the instruction following the one that invoked the Idle Mode. As shown in Figure 26, two or three machine cycles of program execution may take place before the

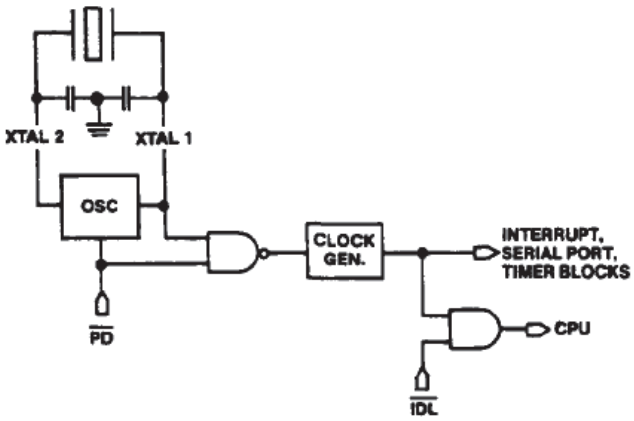



**Figure 28. Idle and Power Down Hardware**

**Table 23. PCON: Power Control Register**

**PCON** Address = 87H Reset Value = 00XX 0000B

Not Bit Addressable

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

| Symbol | Function |
|---|---|
| SMOD1 | Double Baud rate bit. When set to a 1 and Timer 1 is used to generate baud rates, and the Serial Port is used in modes 1, 2, or 3. |
| SMOD0 | When set, Read/Write accesses to SCON.7 are to the FE bit. When clear, Read/Write accesses to SCON.7 are to the SM0 bit. |
| — | Not implemented, reserved for future use.* |
| POF | Power Off Flag. Set by hardware on the rising edge of $V_{CC}$. Set or cleared by software. This flag allows detection of a power failure caused reset. $V_{CC}$ must remain above 3V to retain this bit. |
| GF1 | General-purpose flag bit. |
| GF0 | General-purpose flag bit. |
| PD | Power Down bit. Setting this bit activates Power Down operation. |
| IDL | Idle mode bit. Setting this bit activates idle modes operation. If 1s are written to PD and IDL at the same time, PD takes precedence. |

**NOTE:**
*User software should not write 1s to unimplemented bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate

internal reset algorithm takes control. On-chip hardware inhibits access to the internal RAM during this time, but access to the port pins is not inhibited. To eliminate the possibility of unexpected outputs at the port pins, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external Data RAM.

## 10.2 Power Down Mode

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the Power Down mode. In this mode the on-chip oscillator is stopped. With the clock frozen, all functions are stopped, but the on-chip RAM and Special Function Registers are held. The port pins output the values held by their respective SFRs, and ALE and $\overline{PSEN}$ output lows. In Power Down $V_{CC}$ can be reduced to as low as 2V. Care must be taken, however, to ensure that $V_{CC}$ is not reduced before Power Down is invoked.

The C51FX can exit Power Down with either a hardware reset or external interrupt. Reset redefines all the SFRs but does not change the on-chip RAM. An external interrupt allows both the SFRs and the on-chip RAM to retain their values.

To properly terminate Power Down the reset or external interrupt should not be executed before $V_{CC}$ is restored to its normal operating level and must be held active long enough for the oscillator to restart and stabilize (normally less than 10 msec).

With an external interrupt, $\overline{INT0}$ or $\overline{INT1}$ must be enabled and configured as level-sensitive. Holding the pin low restarts the oscillator and bringing the pin back high completes the exit. After the RETI instruction is executed in the interrupt service routine, the next instruction will be the one following the instruction that put the device in Power Down.

## 10.3 Power Off Flag

The Power Off Flag (POF) located at PCON.4, is set by hardware when $V_{CC}$ rises from 0 to 5 Volts. POF can also be set or cleared by software. This allows the user to distinguish between a "cold start" reset and a "warm start" reset.

A cold start reset is one that is coincident with $V_{CC}$ being turned on to the device after it was turned off. A warm start reset occurs while $V_{CC}$ is still applied to the device and could be generated, for example, by a Watchdog Timer or an exit from Power Down.

Immediately after reset, the user's software can check the status of the POF bit. POF = 1 would indicate a cold start. The software then clears POF and commences its tasks. POF = 0 immediately after reset would indicate a warm start.

$V_{CC}$ must remain above 3 volts for POF to retain a 0.

# 11.0 EPROM VERSIONS

The 8XC51FX uses the Improved "Quick-Pulse" programming™ algorithm. These devices program at $V_{PP}$ = 12.75V (and $V_{CC}$ = 5.0V) using a series of five 100 $\mu$s $\overline{PROG}$ pulses per byte programmed. This results in a total programming time of approximately 5 seconds for the 87C51FA's 8 Kbytes, 10 seconds for the 87C51FB's 16 Kbytes, and 20 seconds for the 87C51FC's 32 Kbytes.

**Exposure to Light:** The EPROM window must be covered with an opaque label when the device is in operation. This is not so much to protect the EPROM array from inadvertent erasure, but to protect the RAM and other on-chip logic. Allowing light to impinge on the silicon die while the device is operating can cause logical malfunction.

# 12.0 PROGRAM MEMORY LOCK

In some microcontroller applications, it is desirable that the Program Memory be secure from software piracy. The C51FX has varying degrees of program protection depending on the device. Table 24 outlines the lock schemes available for each device.

**Encryption Array:** Within the EPROM/ROM is an array of encryption bytes that are initially unprogrammed (all 1's). For EPROM devices, the user can program the encryption array to encrypt the program code bytes during EPROM verification. For ROM devices, the user submits the encryption array to be programmed by the factory. If an encryption array is submitted, LB1 will also be programmed by the factory. The encryption array is not available without the Lock Bit. Program code verification is performed as usual, except that each code byte comes out exclusive-NOR'ed (XNOR) with

one of the key bytes. Therefore, to read the ROM/EPROM code, the user has to know the encryption key bytes in their proper sequence.

Unprogrammed bytes have the value 0FFH. If the Encryption Array is left unprogrammed, all the key bytes have the value 0FFH. Since any code byte XNOR'ed with 0FFH leaves the byte unchanged, leaving the Encryption Array unprogrammed in effect bypasses the encryption feature.

When using the encryption array feature, one important factor should be considered. If a code byte has the value 0FFH, verifying the byte will produce the encryption byte value. If a large block (>64 bytes) of code is left unprogrammed, a verification routine will display the encryption array contents. For this reason all unused code bytes should be programmed with some value other than 0FFH, and not all of them the same value. This will ensure maximum program protection.

**Program Lock Bits:** Also included in the Program Lock scheme are Lock Bits which can be enabled to provide varying degrees of protection. Table 25 lists the Lock Bits and their corresponding influence on the microcontroller. Refer to Table 24 for the Lock Bits available on the various products. The user is responsible for programming the Lock Bits on EPROM devices. On ROM devices, LB1 is automatically set by the factory when the encryption array is submitted. The Lock Bit is not available without the encryption array on ROM devices.

Erasing the EPROM also erases the Encryption Array and the Lock Bits, returning the part to full functionality.

**Table 24. C51FX Program Protection**

| Device | Lock Bits | Encrypt Array |
|---|---|---|
| 83C51FA | None | None |
| 83C51FB | LB1 | 64 Bytes |
| 83C51FC | LB1 | 64 Bytes |
| 87C51FA | LB1, LB2, LB3 | 64 Bytes |
| 87C51FB | LB1, LB2, LB3 | 64 Bytes |
| 87C51FC | LB1, LB2, LB3 | 64 Bytes |

## 13.0 ONCE™ MODE

The ONCE (ON-Circuit Emulation) mode facilitates testing and debugging of systems using the C51FX without having to remove the device from the circuit. The ONCE mode is invoked by:

1. Pulling ALE low while the device is in reset and $\overline{PSEN}$ is high;
2. Holding ALE low as RST is deactivated.

While the device is in ONCE mode, the Port 0 pins go into a float state, and the other port pins, ALE, and $\overline{PSEN}$ are weakly pulled high. The oscillator circuit remains active. While the device is in this mode, an emulator or test CPU can be used to drive the circuit.

Normal operation is restored after a valid reset is applied.

**Table 25. Lock Bits**

| Program Lock Bits | | | | Protection Type |
|---|---|---|---|---|
| | LB1 | LB2 | LB3 | |
| 1 | U | U | U | No program lock features enabled. (Code verify will still be encrypted by the encryption array if programmed.) |
| 2 | P | U | U | MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the EPROM is disabled. |
| 3 | P | P | U | Same as 2, also verify is disabled. |
| 4 | P | P | P | Same as 3, also external execution is disabled. |

P = Programmed
U = Unprogrammed
Any other combination of the Lock Bits is not defined.

## 14.0 ON-CHIP OSCILLATOR

The on-chip oscillator for the CHMOS devices, shown in Figure 29, consists of a single stage linear inverter intended for use as a crystal-controlled, positive reactance oscillator. In this application the crystal is operating in its fundamental response mode as an inductive reactance in parallel resonance with capacitance external to the crystal (Figure 30).

The oscillator on the CHMOS devices can be turned off under software control by setting the PD bit in the PCON register. The feedback resistor $R_f$ in Figure 29 consists of paralleled n- and p-channel FETs controlled by the PD bit, such that $R_f$ is opened when PD = 1. The diodes D1 and D2, which act as clamps to $V_{CC}$ and $V_{SS}$, are parasitic to the $R_f$ FETs.

The crystal specifications and capacitance values (C1 and C2 in Figure 30) are not critical. 30 pF can be used in these positions at any frequency with good quality crystals. In general, crystals used with these devices typically have the following specifications:

| | |
|---|---|
| ESR (Equivalent Series Resistance) | see Figure 32 |
| $C_O$ (shunt capacitance) | 7.0 pF maximum |
| $C_L$ (load capacitance) | 30 pF ±3 pF |
| Drive Level | 1 MW |

Frequency, tolerance, and temperature range are determined by the system requirements.

A ceramic resonator can be used in place of the crystal in cost-sensitive applications. When a ceramic resonator is used, C1 and C2 are normally selected as higher values, typically 47 pF. The manufacturer of the ceramic resonator should be consulted for recommendations on the values of these capacitors.

A more in-depth discussion of crystal specifications, ceramic resonators, and the selection of values for C1 and C2 can be found in Application Note AP-155, "Oscillators for Microcontrollers" in the Embedded Applications handbook.

To drive the CHMOS parts with an external clock source, apply the external clock signal to XTAL1 and leave XTAL2 floating as shown in Figure 31. This is an important difference from the HMOS parts. With HMOS, the external clock source is applied to XTAL2, and XTAL1 is grounded.

An external oscillator may encounter as much as a 100 pF load at XTAL1 when it starts up. This is due to interaction between the amplifier and its feedback capacitance. Once the external signal meets the $V_{IL}$ and $V_{IH}$ specifications the capacitance will not exceed 20 pF.

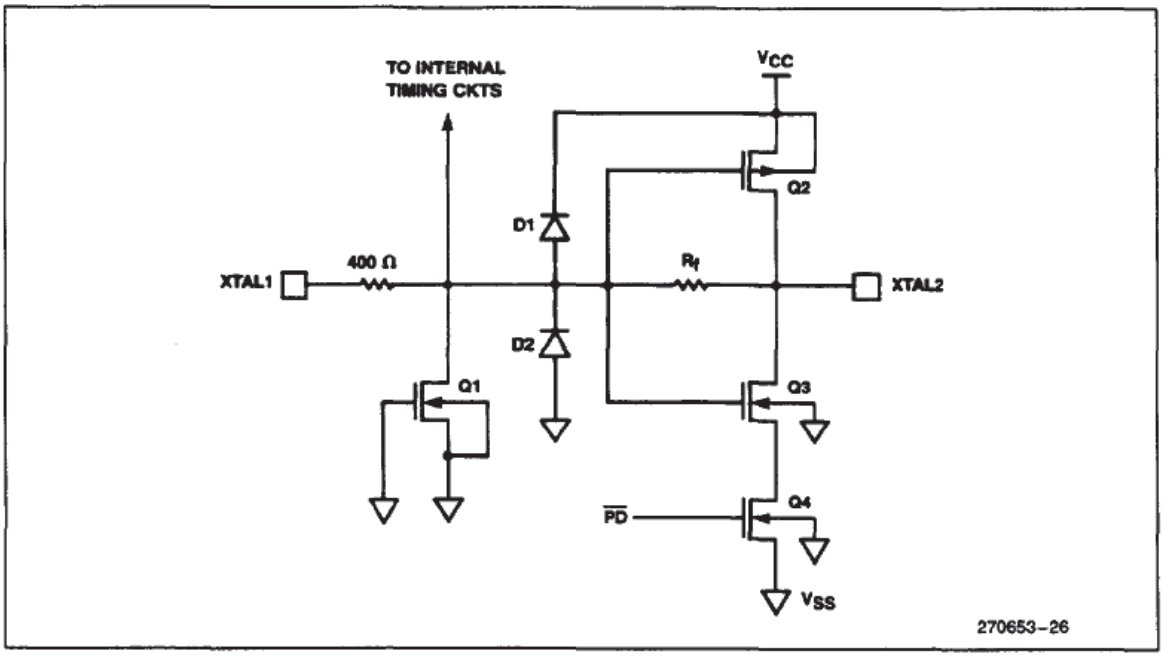



**Figure 29. On-Chip Oscillator Circuitry**

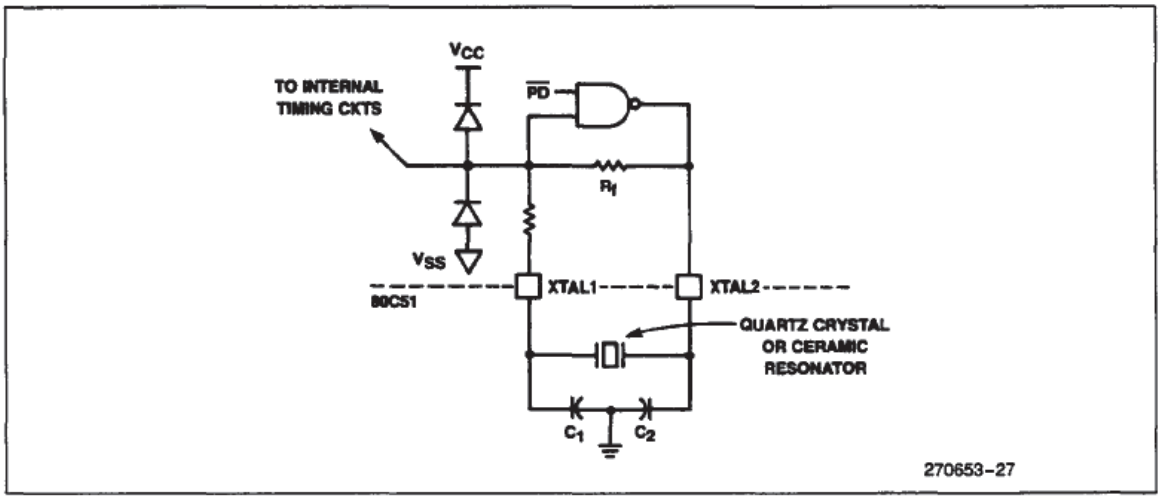



**Figure 30. Using the CHMOS On-Chip Oscillator**

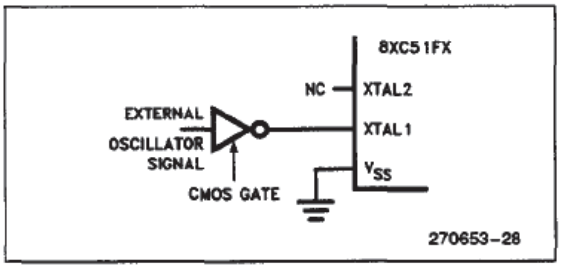



**Figure 31. Driving the CHMOS Parts with an External Clock Source**

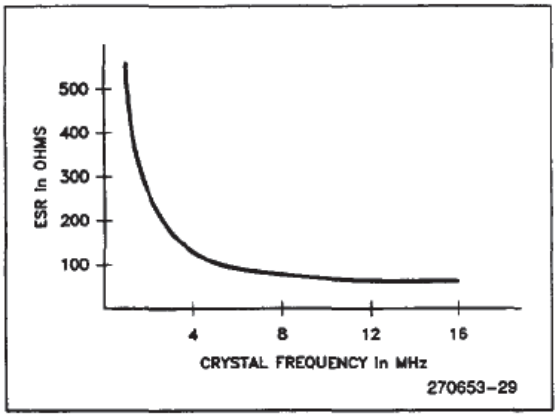



**Figure 32. ESR vs Frequency**

## 15.0 CPU TIMING

The internal clock generator defines the sequence of states that make up a machine cycle. A machine cycle consists of 6 states, numbered S1 through S6. Each state time lasts for two oscillator periods. Thus a machine cycle takes 12 oscillator periods or 1 microsecond if the oscillator frequency is 12 MHz. Each state is then divided into a Phase 1 and Phase 2 half.

Rise and fall times are dependent on the external loading that each pin must drive. They are approximately 10 nsec, measured between 0.8V and 2.0V.

Propagation delays are different for different pins. For a given pin they vary with pin loading, temperature, $V_{CC}$, and manufacturing lot. If the XTAL1 waveform is taken as the timing reference, propagation delays may vary from 25 to 125 nsec.

The AC Timings section of the data sheets do not reference any timing to the XTAL1 waveform. Rather, they relate the critical edges of control and input signals to each other. The timings published in the data sheets include the effects of propagation delays under the specified test condition.

## ADDITIONAL REFERENCES

The following application notes provide supplemental information to this document and can be found in the *Embedded Applications* handbook.

1. AP-125 "Designing Microcontroller Systems for Electrically Noisy Environments"
2. AP-155 "Oscillators for Microcontrollers"
3. AP-252 "Designing with the 80C51BH"
4. AP-410 "Enhanced Serial Port on the 83C51FA"
5. AP-415 "83C51FA/FB PCA Cookbook"
6. AB-41 "Software Serial Port Implemented with the PCA"
7. AP-425 "Small DC Motor Control"
8. The appropriate data sheet.

# 87C51GB Hardware Description

# 6

# 87C51GB Hardware Description

## CONTENTS



## CONTENTS

## 1.0 INTRODUCTION TO THE 8XC51GB

The 8XC51GB is a highly integrated 8-bit microcontroller based on the MCS®-51 architecture. As a member of the MCS-51 family, the 8XC51GB is optimized for control applications. Its key features are an analog to digital converter and two programmable counter arrays (PCA) capable of measuring and generating pulse information on ten I/O pins. Also included are an enhanced serial port for multi-processor communications, a serial expansion port, hardware watchdog timer, oscillator fail detection, an up/down timer/counter and a program lock scheme for the on-chip program memory. Since the 8XC51GB is CHMOS, it has two software selectable reduced power modes: Idle Mode and Power Down Mode.

The 8XC51GB used the standard 8051 instruction set and is functionally compatible with the existing MCS-51 family of products.

This document presents a comprehensive description of the on-chip hardware features of the 8XC51GB. It begins with a discussion of how the memory is organized, followed by the instruction set, and then discusses each of the peripherals listed below.

- Six 8-bit Bidirectional Parallel Ports
- Three 16-bit Timer/Counters with
  - One Up/Down Timer/Counter
  - Programmable Clock Output
- Analog to Digital Converter with
  - 8 channels
  - 8-bit resolution
  - compare mode
- Two Programmable Counter Arrays with
  - Compare/Capture
  - Software Timer
  - High Speed Output
  - Pulse Width Modulator
  - Watchdog Timer (PCA only)
- Full-Duplex Programmable Serial Port with
  - Framing Error Detection
  - Automatic Address Recognition
- Serial Expansion Port
  - four programmable modes
  - four selectable frequencies
- Hardware Watchdog Timer
- Reset
  - asynchronous
  - active low
- Oscillator Fail Detection
- Interrupt Structure with
  - 15 interrupt sources
  - Four priority levels
- Power-Saving Modes
  - Idle Mode
  - Power Down Mode

The table below summarizes the product names of the various 8XC51GB products currently available. Throughout this document, the products will generally be referred to as the 8XC51GB. Figure 1 shows a functional block diagram of the 8XC51GB.

| ROM Device | OTP Version | ROMless Version | ROM/ OTP Bytes | RAM Bytes |
|---|---|---|---|---|
| 87C51GB | 87C51GB | 80C51GB | 8K | 256 |

## 2.0 MEMORY ORGANIZATION

All MCS-51 devices have a separate address space for Program Memory and Data Memory. The logical separation of Program and Data Memory allows the Data Memory to be accessed by 8-bit addresses, which can be more quickly stored and manipulated by an 8-bit CPU. Nevertheless, 16-bit Data Memory addresses can also be generated through the DPTR register. Up to 64 Kbytes each of external Program and Data Memory can be addressed.

### 2.1 Program Memory

Program Memory can only be read, not written to. There can be up to 64 Kbytes of Program Memory. The read strobe for external Program Memory is the signal $\overline{PSEN}$ (Program Store Enable). $\overline{PSEN}$ is not activated for internal program fetches.

If the $\overline{EA}$ (External Access) pin is connected to $V_{SS}$, all program fetches are directed to external memory. For the ROMless devices, all program fetches must be to external memory. If the $\overline{EA}$ pin is connected to $V_{CC}$, then program fetches greater than 8K are to external addresses for the 8XC51GB products.

On the 87C51GB with $\overline{EA}$ connected to $V_{CC}$, program fetches to addresses 0000H through 1FFFH are to internal ROM, and fetches to addresses 2000H through FFFFH are to external memory.

### 2.2 Data Memory

The 8XC51GB implements 256 bytes of on-chip data RAM. The memory space is divided into three blocks,

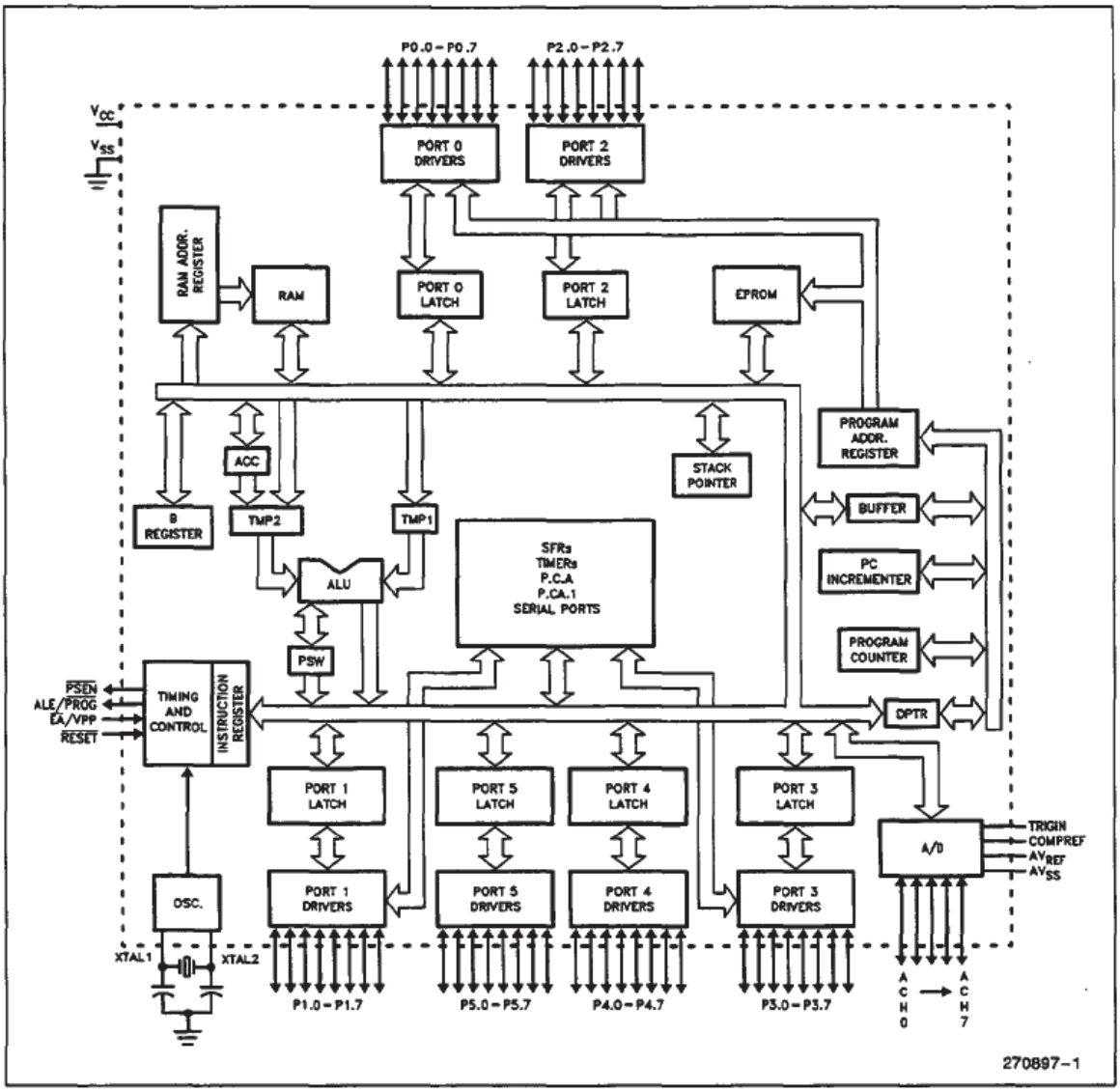



**Figure 1. 87C51GB Block Diagram**

which are generally referred to as the Lower 128, the Upper 128, and SFR space. The Upper 128 bytes occupy a parallel address space to the Special Function Registers. That means they have the same addresses, but they are physically separate from SFR space.

The Lower 128 bytes of RAM are present in all MCS-51 devices. All of the bytes in the Lower 128 can be accessed by either direct or indirect addressing. The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing.

When an instruction accesses an internal location above address 7FH, the CPU knows whether the access is to the upper 128 bytes of data RAM or to SFR space by the addressing mode used in the instruction. Instructions that use direct addressing access SFR space. For example,

```
MOV 0A0H, data
```

accesses the SFR at location 0A0H (which is P2). Instructions that use indirect addressing access the upper 128 bytes of data RAM. For example,

```
MOV @R0, data
```

where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H). Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

## 3.0 SPECIAL FUNCTION REGISTERS

A map of the on-chip memory area called by the SFR (Special Function Register) space is shown in Table 1. Special Function Registers (SFRs) include the Port latches, timers, peripheral controls, etc. These registers can only be accessed by direct addressing. Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 000B. The bit addresses in this area are 80H through 0FFH.

Not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

**Table 1. SFR Mapping and Reset Values**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| F8 | P5<br>00000000 | CH<br>00000000 | CCAP0H<br>XXXXXXXX | CCAP1H<br>XXXXXXXX | CCAP2H<br>XXXXXXXX | CCAP3H<br>XXXXXXXX | CCAP4H<br>XXXXXXXX | | FF |
| F0 | *B<br>00000000 | | | | AD7<br>00000000 | | | SEPSTAT<br>XXXXX000 | F7 |
| E8 | C1CON<br>00000000 | CL<br>00000000 | CCAP0L<br>XXXXXXXX | CCAP1L<br>XXXXXXXX | CCAP2L<br>XXXXXXXX | CCAP3L<br>XXXXXXXX | CCAP4L<br>XXXXXXXX | | EF |
| E0 | *ACC<br>00000000 | | | | AD6<br>00000000 | | | SEPDAT<br>XXXXXXXX | E7 |
| D8 | CCON<br>00X00000 | CMOD<br>00XXX000 | CCAPM0<br>X0000000 | CCAPM1<br>X0000000 | CCAPM2<br>X0000000 | CCAPM3<br>X0000000 | CCAPM4<br>X0000000 | | DF |
| D0 | *PSW<br>00000000 | | | | AD5<br>00000000 | | | SEPCON<br>XX000000 | D7 |
| C8 | T2CON<br>00000000 | T2MOD<br>XXXXXX00 | RCAP2L<br>00000000 | RCAP2H<br>00000000 | TL2<br>00000000 | TH2<br>00000000 | | | CF |
| C0 | P4<br>00000000 | | | | AD4<br>00000000 | | EXICON<br>X0000000 | ACMP<br>00000000 | C7 |
| B8 | *IP<br>X0000000 | SADEN<br>00000000 | C1CAP0H<br>XXXXXXXX | C1CAP1H<br>XXXXXXXX | C1CAP2H<br>XXXXXXXX | C1CAP3H<br>XXXXXXXX | C1CAP4H<br>XXXXXXXX | CH1<br>00000000 | BF |
| B0 | *P3<br>11111111 | | | | AD3<br>00000000 | IPAH<br>00000000 | IPA<br>00000000 | IPH<br>X0000000 | B7 |
| A8 | *IE<br>00000000 | SADDR<br>00000000 | C1CAP0L<br>XXXXXXXX | C1CAP1L<br>XXXXXXXX | C1CAP2L<br>XXXXXXXX | C1CAP3L<br>XXXXXXXX | C1CAP4L<br>XXXXXXXX | CL1<br>00000000 | AF |
| A0 | *P2<br>00000000 | | | | AD2<br>00000000 | OSCR<br>XXXXXXX0 | WDTRST<br>XXXXXXXX | IEA<br>00000000 | A7 |
| 98 | *SCON<br>00000000 | *SBUF<br>XXXXXXXX | C1CAPM0<br>X0000000 | C1CAPM1<br>X0000000 | C1CAPM2<br>X0000000 | C1CAPM3<br>X0000000 | C1CAPM4<br>X0000000 | C1MOD<br>XXXX0000 | 9F |
| 90 | *P1<br>00000000 | | | | AD1<br>00000000 | | | ACON<br>XX000000 | 97 |
| 88 | *TCON<br>00000000 | *TMOD<br>00000000 | *TL0<br>00000000 | *TL1<br>00000000 | *TH0<br>00000000 | *TH1<br>00000000 | | | 8F |
| 80 | *P0<br>11111111 | *SP<br>00000111 | *DPL<br>00000000 | *DPH<br>00000000 | AD0<br>00000000 | | | *PCON**<br>00XX0000 | 87 |

* = Found in the 8051 core (see 8051 Hardware Description for explanations of these SFRs).
** = See description of PCON SFR. Bit PCON.4 is not affected by reset.
X = Undefined.

User software should not write 1's to these unimplemented locations, since they may be used in future MCS-51 products to invoke new features. In that case the reset or inactive values of the new bits will always be 0, and their active values will be 1.

The functions of the SFRs are outlined below. More information on the use of specific SFRs for each peripheral is included in the description of that peripheral.

**Accumulator:** ACC is the Accumulator register. The mnemonics for Accumulator-Specific instructions, however, refer to the Accumulator simply as A.

**B Register:** The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

**Stack Pointer:** The Stack Pointer Register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. The stack may reside anywhere in on-chip RAM. On reset, the Stack Pointer is initialized to 07H causing the stack to begin at location 08H.

**Data Pointer:** The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address, but it may be manipulated as a 16-bit register or as two independent 8-bit registers.

**Program Status Word:** The PSW register contains program status information as detailed in Table 2.

**Ports 0 to 5 Registers:** P0, P1, P2, P3, P4, and P5 are the SFR latches of Ports 0 through 5 respectively.

**Timer Registers:** Register pairs (TH0, TL0), (TH1, TL1) and (TH2, TL2) are the 16-bit count registers for Timer/Counters 0, 1, and 2 respectively. Control and status bits are contained in registers TCON and TMOD for Timers 0 and 1 and in registers T2CON and T2MOD for Timer 2. The register pair (RCAP2H, RCAP2L) are the capture/reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

**Programmable Counter Array (PCA and PCA1) Registers:** The 16-bit PCA and PCA1 timer/counters consist of register CH (CH1) and CL (CL1). Registers CCON (C1CON) and CMOD (C1MOD) contain the control and status bits for the PCA (and PCA1). The CCAPMn (n = 0, 1, 2, 3, or 4) and the C1CAPMn registers control the mode for each of the five PCA and the five PCA1 modules. The register pairs (CCAPnH, CCAPnL and C1CAPnH, C1CAPnL) are the 16-bit compare/capture registers for each PCA and PCA1 module.

**Serial Port Registers:** The Serial Data Buffer, SBUF, is actually two separate registers: a transmit buffer and a receive buffer register. When data is moved to SBUF, it comes from the receive buffer. Register SCON contains the control and status bits for the Serial Port. Registers SADDR and SADEN are used to define the Given and the Broadcast addresses for the Automatic Address Recognition feature.

**Table 2. PSW: Program Status Word Register**

**PSW** Address = 0D0H Reset Value = 0000 0000B

Bit Addressable

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

| Symbol | Function |
|---|---|
| CY | Carry flag. |
| AC | Auxiliary Carry flag. (For BCD Operations) |
| F0 | Flag 0. (Available to the user for general purposes). |
| RS1 | Register bank select bit 1. |
| RS0 | Register bank select bit 0. |

| RS1 | RS0 | Working Register Bank and Address |
|---|---|---|
| 0 | 0 | Bank 0 (00H–07H) |
| 0 | 1 | Bank 1 (08H–0FH) |
| 1 | 0 | Bank 2 (10H–17H) |
| 1 | 1 | Bank 3 (18H–1FH) |

| | |
|---|---|
| OV | Overflow flag. |
| — | User definable flag. |
| P | Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the Accumulator, i.e., even parity. |

**Serial Expansion Port Registers:** The Serial Expansion Port is controlled through the register SEPCON. SEPDAT contains data for the Serial Expansion Port and SEPSTAT is used to monitor its status.

**Interrupt Registers:** The individual interrupt enable bits are in the IE and IEA registers. One of four priority levels can be selected for each interrupt using the IP, IPH, IPA and IPAH registers. The EXICON register controls the selection of the activation polarity for external interrupts two and three.

**Analog to Digital Converter Registers:** The results of A/D conversions are placed in registers AD0, AD1, AD2, AD3, AD4, AD5, AD6, and AD7 for analog channels 0 through 7 respectively. The register ACMP contains the results of the A/D comparison feature. ACON is the control register for A/D conversions.

**Power Control Register:** PCON controls the Power Reduction Modes, Idle and Power Down.

**Oscillator Fail Detect Register:** The OSCR register is used both to monitor the status of the OFD circuitry and to disable the feature.

**Watchdog Timer Register:** The WatchDog Timer ReSeT (WDTRST) register is used to keep the watchdog timer from periodically resetting the part.

**Table 3. Alternate Port Functions**

| Port Pin | Alternate Function |
|---|---|
| P0.0/AD0–P0.7/AD7 | Multiplexed Byte of Address/Data for external memory. |
| P1.0/T2 | Timer 2 External Clock Input/Clockout |
| P1.1/T2EX | Timer 2 Reload/Capture/Direction Control |
| P1.2/ECI | PCA External Clock Input |
| P1.3/CEX0 | PCA Module 0 Capture Input, Compare/PWM Output |
| P1.4/CEX1 | PCA Module 1 Capture Input, Compare/PWM Output |
| P1.5/CEX2 | PCA Module 2 Capture Input, Compare/PWM Output |
| P1.6/CEX3 | PCA Module 3 Capture Input, Compare/PWM Output |
| P1.7/CEX4 | PCA Module 4 Capture Input, Compare/PWM Output |
| P2.0/A8–P2.7/A15 | High Byte of Address for External Memory |
| P3.0/RXD | Serial Port Input |
| P3.1/TXD | Serial Port Output |
| P3.2/$\overline{INT0}$ | External Interrupt 0 |
| P3.3/$\overline{INT1}$ | External Interrupt 1 |
| P3.4/T0 | Timer 0 External Clock Input |
| P3.5/T1 | Timer 1 External Clock Input |
| P3.6/$\overline{WR}$ | Write Strobe for External Memory |
| P3.7/$\overline{RD}$ | Read Strobe for External Memory |
| P4.0/SEPCLK | Clock Source for SEP |
| P4.1/SEPDAT | Data I/O for SEP |
| P4.2/ECI1 | PCA1 External Clock Input |
| P4.3/C1EX0 | PCA1 Module 0, Capture Input, Compare/PWM Output |
| P4.4/C1EX1 | PCA1 Module 1, Capture Input, Compare/PWM Output |
| P4.5/C1EX2 | PCA1 Module 2, Capture Input, Compare/PWM Output |
| P4.6/C1EX3 | PCA1 Module 3, Capture Input, Compare/PWM Output |
| P4.7/C1EX4 | PCA1 Module 4, Capture Input, Compare/PWM Output |
| P5.2/INT2 | External Interrupt 2 |
| P5.3/INT3 | External Interrupt 3 |
| P5.4/INT4 | External Interrupt 4 |
| P5.5/INT5 | External Interrupt 5 |
| P5.6/INT6 | External Interrupt 6 |

**NOTE:**
The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise the port pin will not go high.

## 4.0 I/O PORTS

All six ports in the 8XC51GB are bidirectional. Each consists of a latch (Special Function Register P0 through P5), output driver and an input buffer. All the ports, except for Port 0, have Schmitt Trigger inputs.

The output drivers of Ports 0 and 2, and the input buffers of Port 0, are used in accesses to external memory. In this application, Port 0 outputs the low byte of the external memory address, time-multiplexed with the byte being written or read. Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise the Port 2 pins continue to emit the P2 SFR content.

All the Port 1, Port 3, Port 4 and most of Port 5 pins are multi-functional. They are not only port pins, but also serve the functions of various special features as shown in Table 3.

## 4.1 I/O Configurations

Functional diagrams of a bit latch and I/O buffer in each of the four ports are shown in Figure 2.

The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which clocks in a value from the internal bus in response to a "write to latch" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in response to a "read latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal. Those that read the latch are the Read-Modify-Write instructions.

The output drivers of Ports 0 and 2 are switchable to an internal ADDRESS and ADDRESS/DATA bus by an internal control signal for use in external memory accesses. During external memory accesses, the P2 SFR

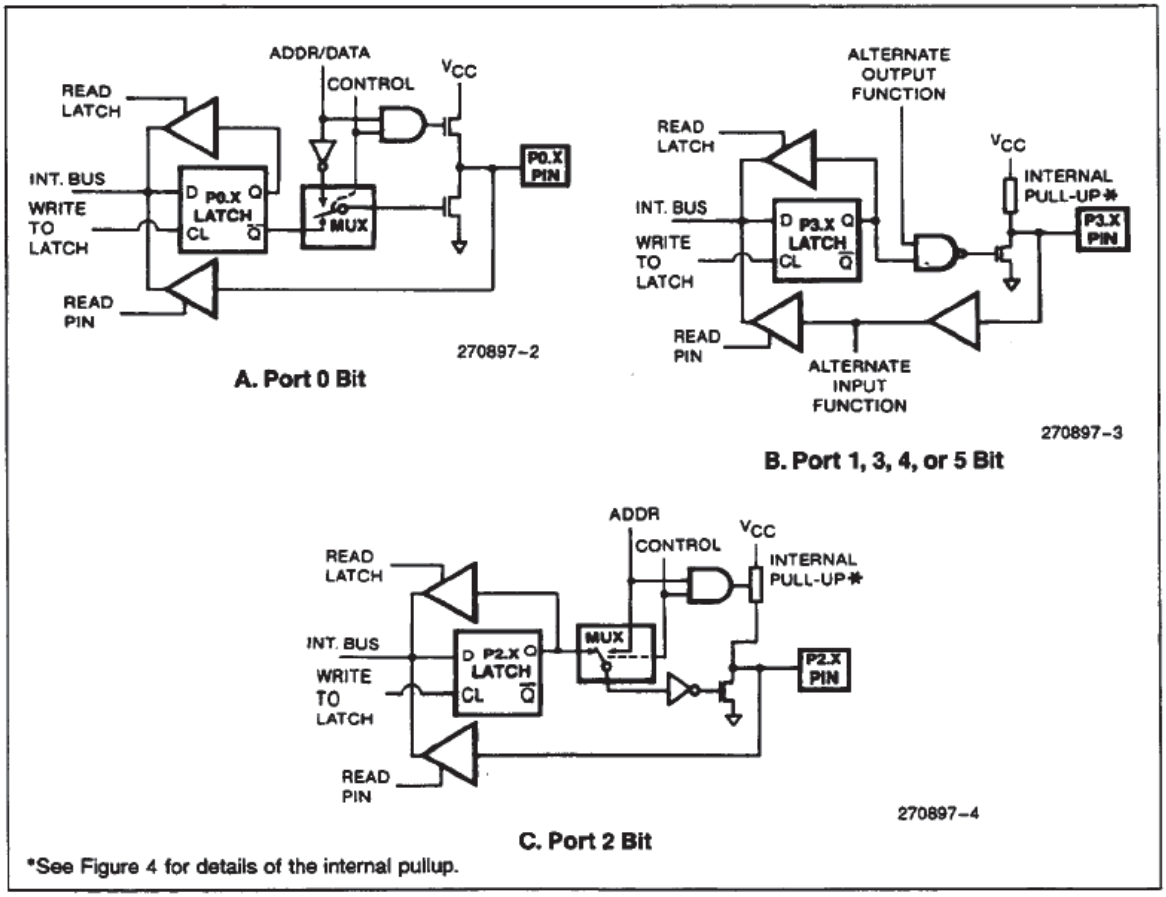



**Figure 2. 8XC51GB Port Bit Latches and I/O Buffers**

remains unchanged, but the P0 SFR gets 1s written to it.

If a P1 through P5 latch contains a 1, then the output level is controlled by the signal labeled "alternate output function." The pin level is always available to the pin's alternate input function, if any.

Ports 1 through 5 have internal pullups. Port 0 has open drain outputs. Each I/O line can be independently used as an input or an output (Ports 0 and 2 may not be used as general purpose I/O when being used as the ADDRESS/DATA BUS). To be used as an input, the port bit latch must contain a 1, which turns off the output driver FET. On Ports 1 through 5 the pin is pulled high by the internal pullup, but can be pulled low by an external source.

P1, P2, P4, and P5 reset to a low state. While in reset these pins can sink large amounts of current. If these ports are to be used as inputs and externally driven high while in reset, the user should be aware of possible contention. A simple solution is to use open collector interfaces with these port pins or to buffer the inputs.

Port 0 differs from the other ports in not having internal pullups. The pullup FET in the P0 output driver is used only when the port is emitting 1s during external memory accesses. Otherwise the pullup FET is off. Consequently P0 lines that are being used as output port lines are open drain. Writing a 1 to the bit latch leaves both output FETs off, which floats the pin and allows it to be used as a high-impedance input. Because Ports 1 through 5 have fixed internal pullups they are sometimes called "quasi-bidirectional" ports.

When configured as inputs they pull high and will source current ($I_{IL}$ in the data sheets) when externally pulled low. Port 0, on the other hand, is considered "true" bidirectional, because it floats when configured as an input.

The latches for ports 0 and 3 have 1s written to them by the reset function. If a 0 is subsequently written to a port latch, it can be reconfigured as an input by writing a 1 to it.

## 4.2 Writing to a Port

In the execution of an instruction that changes the value in a port latch, the new value arrives at the latch during State 6, Phase 2 of the final cycle of the instruction. However, port latches are sampled by their output buffers only during Phase 1 of any clock period. (During Phase 2 the output buffer holds the value it saw during the previous Phase 1). Consequently, the new value in the port latch won't actually appear at the output pin until the next Phase 1, which will be at S1P1 of the next machine cycle. Refer to Figure 3.

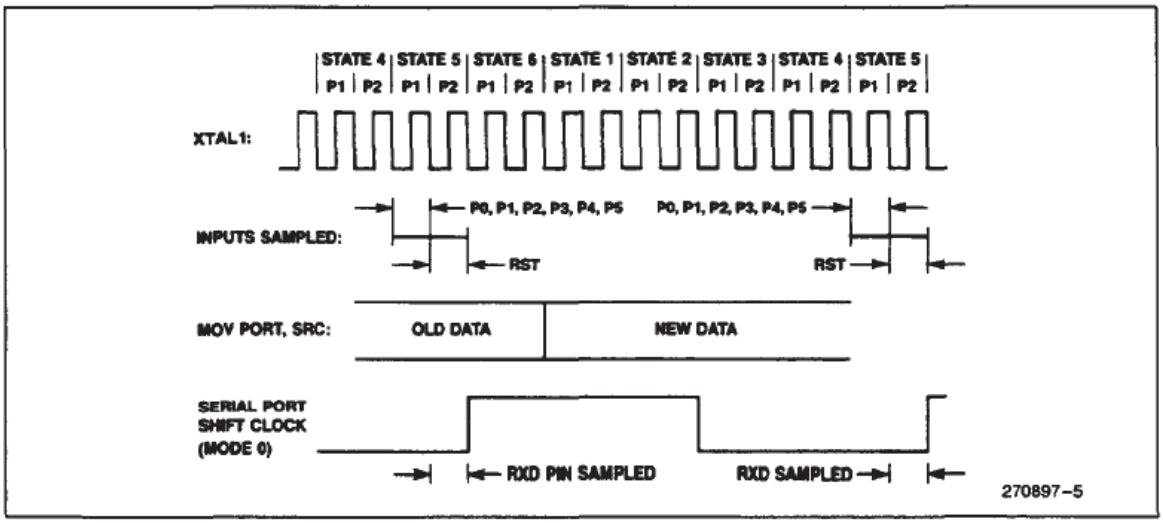



**Figure 3. Port Operation**

For more information on internal timings refer to the CPU Timing section.

If the change requires a 0-to-1 transition in Ports 1 through 5, an additional pullup is turned on during S1P1 and S1P2 of the cycle in which the transition occurs. This is done to increase the transition speed. The extra pullup can source about 100 times the current that the normal pullup can. The internal pullups are field-effect transistors, not linear resistors. The pullup arrangements are shown in Figure 4.

The pullup consists of three pFETs. Note that an n-channel FET (nFET) is turned on when a logical 1 is applied to its gate, and is turned off when a logical 0 is applied to its gate. A p-channel FET (pFET) is the opposite: it is on when its gate sees a 0, and off when its gate sees a 1.

pFET 1 is the transistor that is turned on for 2 oscillator periods after a 0-to-1 transition in the port latch. A 1 at the port pin turns on pFET3 (a weak pullup), through the inverter. This inverter and pFET form a latch which hold the 1.

If the pin is emitting a 1, a negative glitch on the pin from some external source can turn off pFET2, causing the pin to go into a float state. pFET2 is a very weak pullup which is on whenever the nFET is off, in traditional CMOS style. It's only about $1/_{10}$ the strength of pFET2. Its function is to restore a 1 to the pin in the event the pin had a 1 and lost it to a glitch.

## 4.3 Port Loading and Interfacing

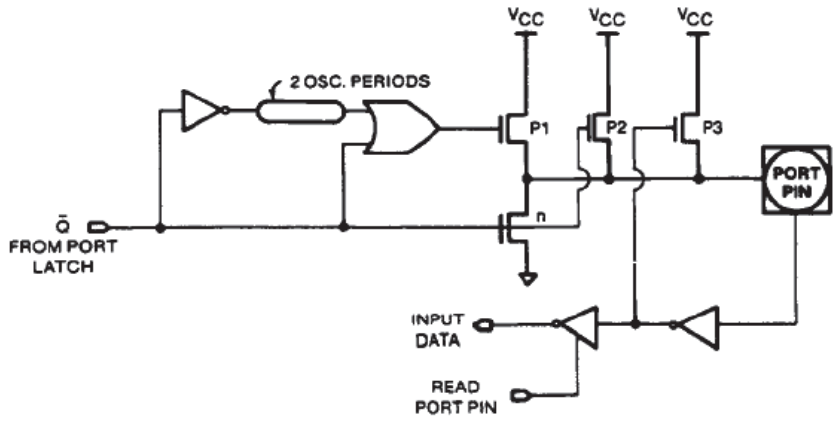
The output buffers of Ports 1 through 5 can each sink at least the amount of current specified by $V_{OL}$ in the data sheet. These port pins can be driven by open-collector and open-drain outputs although 0-to-1 transitions will not be fast since there is little current pulling the pin up. An input 0 turns off pullup pFET2, leaving only the very weak pullup pFET2 to drive the transition.

In external bus mode, Port 0 output buffers can each sink the amount of current specified at the test conditions for VOL1 in the data sheet. However, as port pins they require external pullups to be able to drive any inputs.

See the latest revision of the data sheet for design-in information.

## 4.4 Read-Modify-Write Instructions

Some instructions that read a port read the latch and others read the pin. Which ones do which? The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write" instructions. Listed on the following page, are the read-modify-write instructions. When the destination




**NOTE:**
CHMOS Configuration. pFET 1 is turned on for 2 osc. periods after $\overline{Q}$ mades a 0-to-1 transition. During this time, pFET 1 also turns on pFET 3 through the inverter to form a latch which holds the 1. pFET 2 is also on. Port 2 is similar except that it holds the strong pullup on while emitting 1s that are address bits. (See text, "Accessing External Memory".)

**Figure 4. Ports 1, 3, 4, and 5 Internal Pullup Configuration**

operand is a port, or a port bit, these instructions read the latch rather than the pin:

| | |
|---|---|
| ANL | (logical AND, e.g. ANL P1, A) |
| ORL | (logical OR, e.g. ORL P2, A) |
| XRL | (logical EX-OR, e.g. XRL P3, A) |
| JBC | (jump if bit = 1 and clear bit, e.g. JBC P1.1, LABEL) |
| CPL | (complement bit, e.g. CPL P3.0) |
| INC | (increment, e.g. INC P2) |
| DEC | (decrement, e.g. DEC P2) |
| DJNZ | (decrement and jump if not zero, e.g. DJNZ P3, LABEL) |
| MOV PX.Y, C | (move carry bit to bit Y of Port X) |
| CLR PX.Y | (clear bit Y of Port X) |
| SETB PX.Y | (set bit Y of Port X) |

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of a transistor. When a 1 is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as a 0. Reading the latch rather than the pin will return the correct value of 1.

## 4.5 Accessing External Memory

Accesses to external memory are of two types: accesses to external Program Memory and accesses to external Data Memory. Accesses to external Program Memory use signal $\overline{PSEN}$ (program store enable) as the read strobe. Accesses to external Data Memory use $\overline{RD}$ or $\overline{WR}$ (alternate functions of P3.7 and P3.6) to strobe the memory. Refer to Figures 5 through 7.

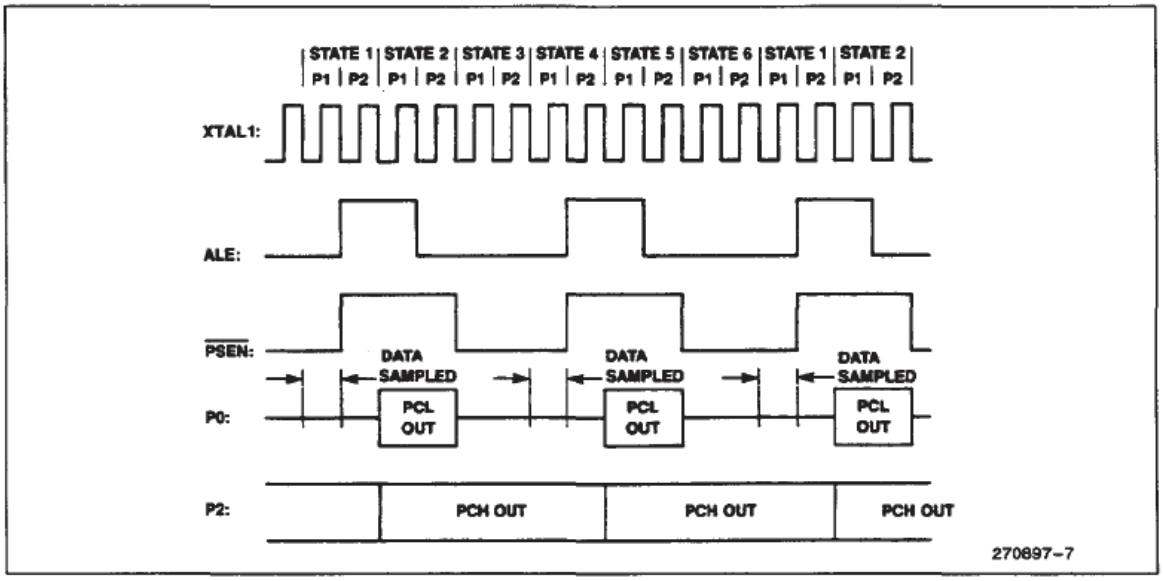



**Figure 5. External Program Memory Fetches**

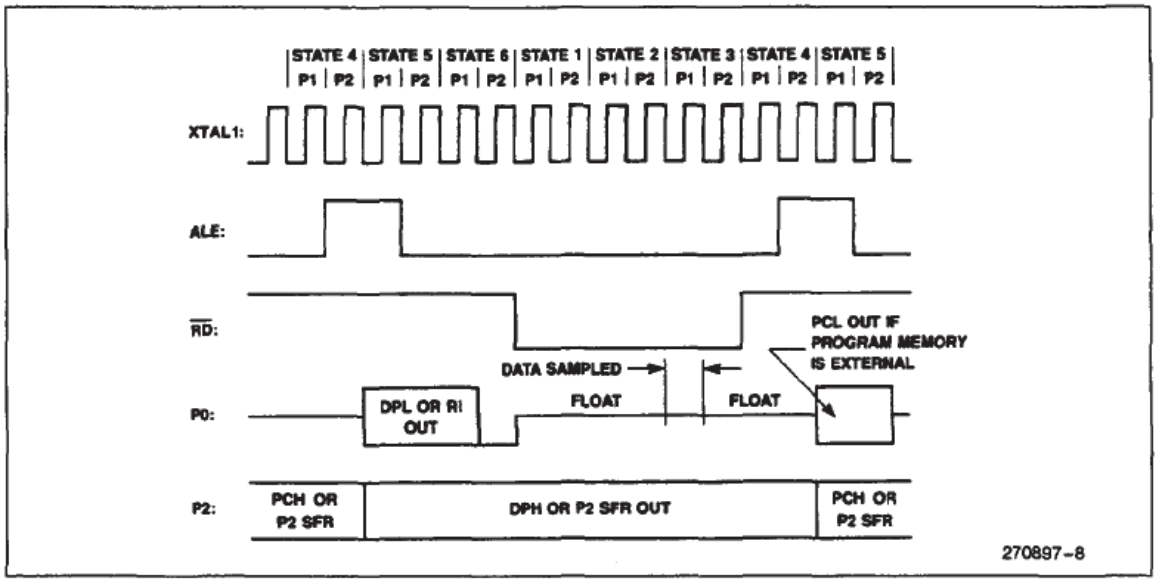



**Figure 6. External Data Memory Read Cycle**

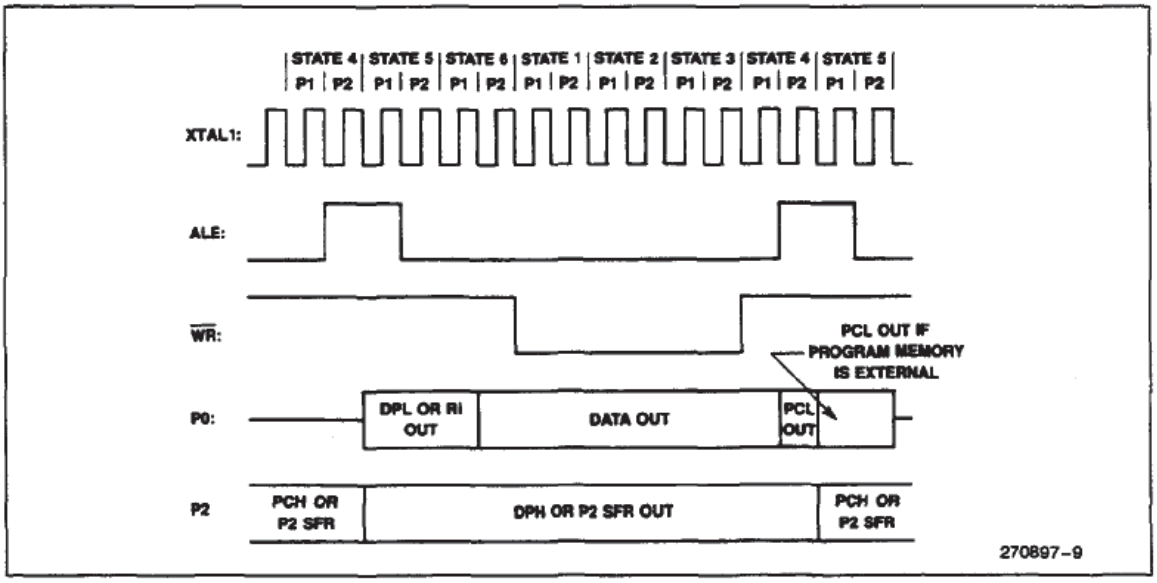



**Figure 7. External Data Memory Write Cycle**

Fetches from external Program Memory always use a 16-bit address. Accesses to external Data Memory can use either a 16-bit address (MOVX @ DPTR) or an 8-bit address (MOVX @ Ri).

Whenever a 16-bit address is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle. The Port 2 drivers use the strong pullups during the entire time that they are emitting address bits that are 1s. This occurs when the MOVX @ DPTR instruction is executed. During this time the Port 2 latch (the Special Function Register) does not have to contain 1s, and the contents of the Port 2 SFR are not modified. If the external memory cycle is not immediately followed by another external memory cycle, the undisturbed contents of the Port 2 SFR will reappear in the next cycle.

If an 8-bit address is being used (MOVX @ Ri), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. In this case, Port 2 pins can be used to page the external data memory.

In either case, the low byte of the address is time-multiplexed with the data byte on Port 0. The ADDRESS/DATA signal drives both FETs in the Port 0 output buffers. Thus, in external bus mode the Port 0 pins are not open-drain outputs and do not require external pullups. The ALE (Address Latch Enable) signal should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before $\overline{WR}$ is activated, and remains there until after $\overline{WR}$ is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe ($\overline{RD}$) is deactivated.

During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating the information in the Port 0 SFR. Also, a MOV P0 instruction must not take place during external memory accesses. If the user writes to Port 0 during an external memory fetch, the incoming code byte is corrupted. Therefore, do not write to Port 0 if external program memory is used.

External Program Memory is accessed under two conditions:

1. Whenever signal $\overline{EA}$ is high, or
2. Whenever the program counter (PC) contains an address greater than 1FFFH (8K).

This requires that the ROMless versions have $\overline{EA}$ wired to $V_{SS}$ to enable the lower 8K of program bytes to be fetched from external memory.

When the CPU is executing out of external Program Memory, all 8 bits of Port 2 are dedicated to an output function and may not be used for general purpose I/O. During external program fetches they output the high byte of the PC with the Port 2 drivers using the strong pullups to emit bits that are 1s.

## 5.0 TIMER/COUNTERS

The 8XC51GB has three 16-bit Timer/Counters: Timer 0, Timer 1, and Timer 2. Each consists of two 8-bit registers: THx and TLx with x = 0, 1, or 2. All three can be configured to operate either as timers or event counters.

In the Timer function, the TLx register is incremented every machine cycle. Thus, you can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin: T0, T1, or T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full machine cycle.

Timer 0 and Timer 1 have four operating modes:

Mode 0: 13-bit timer

Mode 1: 16-bit timer

Mode 2: 8-bit auto-reload timer

Mode 3: Timer 0 as two separate 8-bit timers

Also, its possible to use Timer 1 to generate baud rates.

Timer 2 has three modes of operation:

Timer 2 Capture

Timer 2 Auto-Reload (up or down counting), and

Timer 2 as a Baud Rate Generator

### 5.1 Timer 0 and Timer 1

The Timer/Counter function is selected by control bits C_Tx in TMOD (Table 4). These two Timer/Counters have four operating modes, which are selected by bit-pairs (M1x, M0x) also in TMOD. Mode 0, Mode 1, and Mode 2 are the same for both Timer/Counters. Mode 3 operation is different for the two timers.

**Table 4. TMOD: Timer/Counter Mode Control Register**

**TMOD** Address = 89H Reset Value = 0000 0000B

Not Bit Addressable

| | TIMER 1 | | | | TIMER 0 | | | |
|---|---|---|---|---|---|---|---|---|
| | GATE | C/$\overline{T}$ | M1 | M0 | GATE | C/$\overline{T}$ | M1 | M0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| GATE | Gating control when set. Timer/Counter 0 or 1 is enabled only while $\overline{INT0}$ or $\overline{INT1}$ pin is high and TR0 or TR1 control pin is set. When cleared, Timer 0 or 1 is enabled whenever TR0 or TR1 control bit is set. |
| C/$\overline{T}$ | Timer or Counter Selector. Clear for Timer operation (input from internal system clock). Set for Counter operation (input from T0 or T1 input pin). |

| M1 | M0 | Operating Mode |
|---|---|---|
| 0 | 0 | 8-bit Timer/Counter. THx with TLx as 5-bit prescaler. |
| 0 | 1 | 16-bit Timer/Counter. THx and TLx are cascaded; there is no prescaler. |
| 1 | 0 | 8-bit auto-reload Timer/Counter. THx holds a value which is to be reloaded into TLx each time it overflows. |
| 1 | 1 | (Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits. |
| 1 | 1 | (Timer 1) Timer/Counter stopped. |

## MODE 0

Either Timer 0 or Timer 1 in Mode 0 is an 8-bit counter with a divide-by-32 prescaler. In this mode, the Timer register is configured as a 13-bit register. Figure 8 shows the Mode 0 operation for either timer.

As the count rolls over from all 1s to all 0s, it sets the timer interrupt flag TF0 or TF1. The counted input is enabled to the timer when TR0 or TR1 = 1, and either GATEx = 0 or $\overline{INTx}$ pin = 1. (Setting GATEx = 1 allows the Timer to be controlled by external input $\overline{INTx}$ pin, to facilitate pulse width measurements).

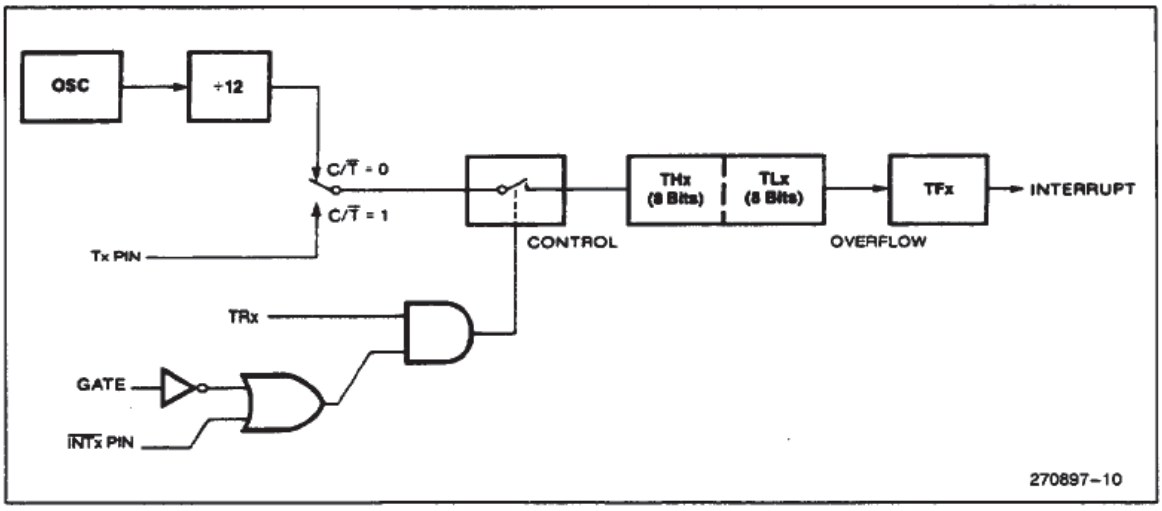



**Figure 8. Timer/Counter 0 or 1 in Mode 0: 13-Bit Counter**

TRx and TFx are control bits in the SFR TCON. The GATEx bits are in TMOD. There are two different GATE bits: one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

The 13-bit register consists of all 8 bits of THx and the lower 5 bits of TLx. The upper 3 bits of TLx are indeterminate and should be ignored. Setting the run flag (TRx) does not clear these registers.

### MODE 1

Mode 1 is the same as Mode 0, except that the Timer register uses all 16-bits. In this mode, THx and TLx are cascaded; there is no prescaler. Refer to Figure 9.

As the count rolls over from all 1s to all 0s, it sets the timer interrupt flag TF0 or TF1. The counted input is enabled to the timer when TR0 or TR1 = 1, and either GATEx = 0 or $\overline{INTx}$ pin = 1. (Setting GATEx = 1

**Table 5. TCON: Timer/Counter Control Register**

**TCON** Address = 88H Reset = 0000 0000B

Bit Addressable

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

| Symbol | Function |
|---|---|
| TF1 | Timer 1 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine. |
| TR1 | Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter 1 on/off. |
| TF0 | Timer 0 overflow Flag. Set by hardware on Timer/Counter 0 overflow. Cleared by hardware when processor vectors to interrupt routine. |
| TR0 | Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter 0 on/off. |
| IE1 | Interrupt 1 flag. Set by hardware when external interrupt 1 edge is detected (transmitted or level-activated). Cleared when interrupt processed only if transition-activated. |
| IT1 | Interrupt 1 Type control bit. Set/cleared by software to specifiy falling edge/low level triggered external interrupt 1. |
| IE0 | Interrupt 0 flag. Set by hardware when external interrupt 0 edge is detected (transmitted or level-activated). Cleared when interrupt processed only if transition-activated. |
| IT0 | Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupt 0. |

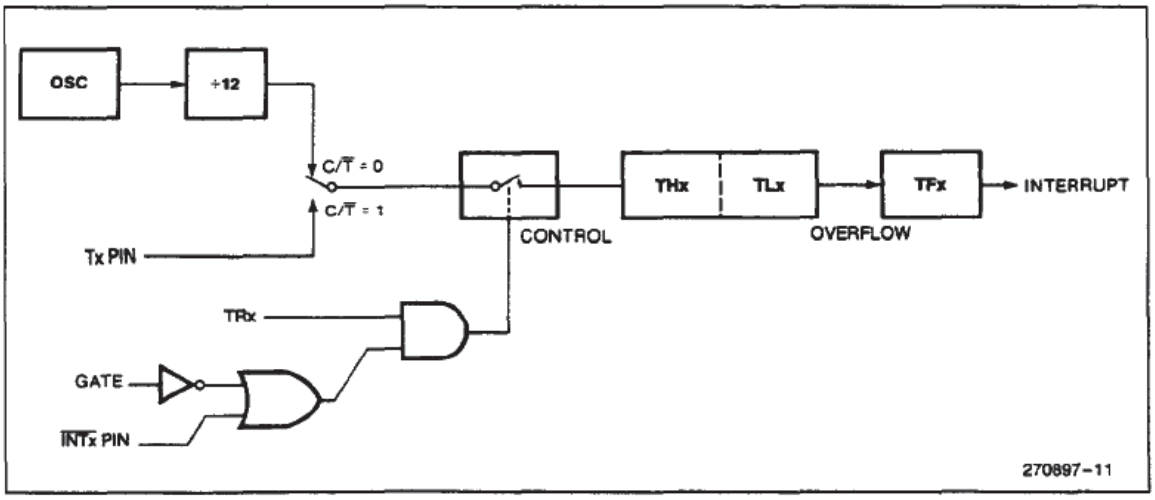



**Figure 9. Timer/Counter 0 or 1 In Mode 1: 16-Bit Counter**

allows the Timer to be controlled by external input $\overline{INTx}$ pin to facilitate pulse width measurements).

TRx and TFx are control bits in the SRF TCON. The GATEx bits are in TMOD. There are two different GATE bits: one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

### MODE 2

Mode 2 configures the Timer register as an 8-bit Counter (TLx) with automatic reload as shown in Figure 10. Overflow from TLx not only sets TFx, but also reloads TLx with the contents of THx, which is preset by software. The reload leaves THx unchanged.

The counted input is enabled to the timer when TR0 or TR1 = 1, and either GATEx = 0 or $\overline{INTx}$ pin = 1. (Setting GATEx = 1 allows the Timer to be controlled by external input $\overline{INTx}$ pin, to facilitate pulse width measurements).

TRx and TFx are control bits in the SFR TCON. The GATEx bits are in TMOD. There are two different GATE bits: one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

### MODE 3

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0.

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. TL0 uses the Timer 0 control bits: C_T0, GATE0, TR0, and TF0. TH0 is locked into a

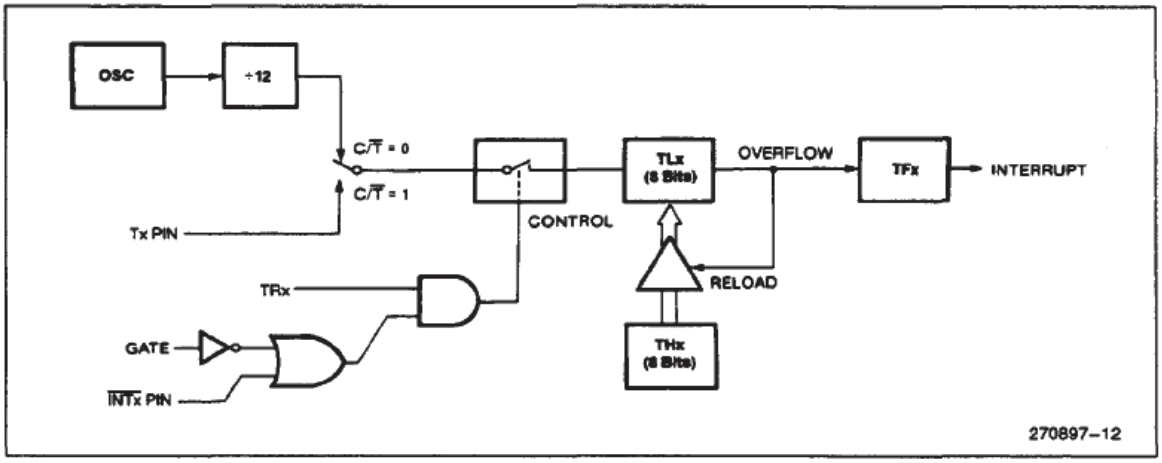



**Figure 10. Timer/Counter 1 Mode 2: 8-Bit Auto-Reload**

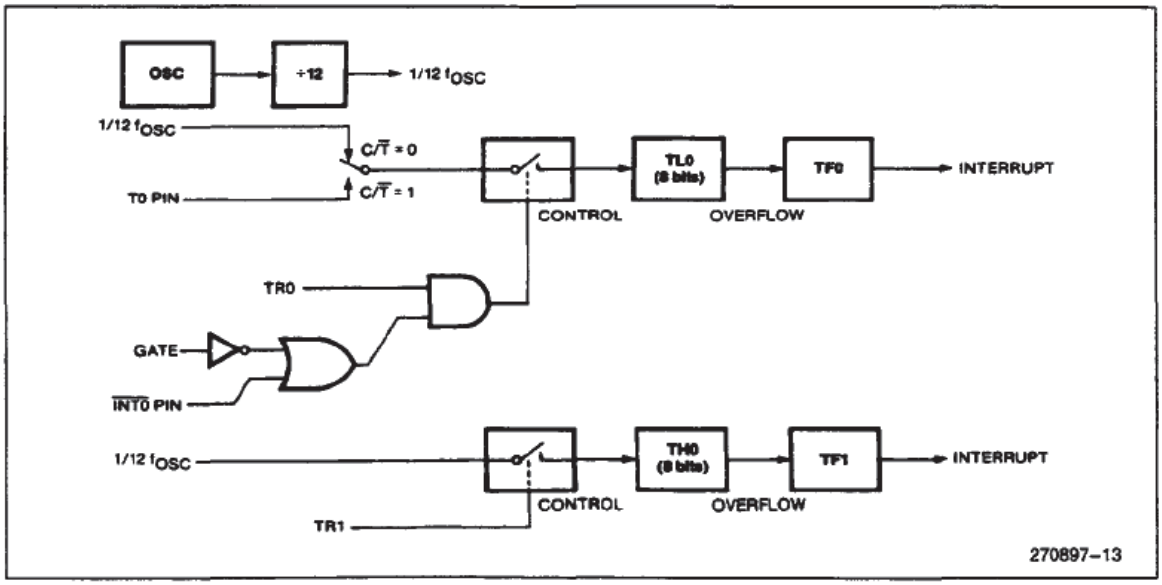



**Figure 11. Timer/Counter 0 Mode 3: Two 8-Bit Counters**

timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus TH0 now controls the Timer 1 interrupt. The logic for Mode 3 on Timer 0 is shown in Figure 11.

Mode 3 is provided for applications requiring an extra 8-bit timer or counter. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3, or can still be used by the serial port as a baud rate generator, or in any application not requiring an interrupt.

## 5.2 Timer 2

Timer 2 is a 16-bit Timer/Counter which can operate either as a timer or as an event counter. This is selected by bit C_T2 in the SFR T2CON (Table 7). It has the following three operating modes:

Timer 2 Capture,

Timer 2 Auto-Reload (up or down counting), and

Timer 2 as a Baud Rate Generator.

The modes are also selected by bits in T2CON as shown in Table 6.

**Table 6. Timer 2 Operating Modes**

| RCLK + TCLK | CP/$\overline{RL2}$ | T2*OE | TR2 | Mode |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 16-Bit Auto-Reload |
| 0 | 1 | 0 | 1 | 16-Bit Capture |
| 1 | X | X | 1 | Baud_Rate Generator |
| X | 0 | 1 | 1 | Clock-Out on P1.0* |
| X | X | X | 0 | Timer Off |

*Present only on the 87C51FC.

**Table 7. T2CON: Timer/Counter 2 Control Register**

**T2CON** Address = 0C8H Reset Value = 0000 0000B

Bit Addressable

| TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/$\overline{T2}$ | CP/$\overline{RL2}$ |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| TF2 | Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1. |
| EXF2 | Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1). |
| RCLK | Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock. |
| TCLK | Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock. |
| EXEN2 | Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX. |
| TR2 | Start/stop control for Timer 2. A logic 1 starts the timer. |
| C/$\overline{T2}$ | Timer or counter select, (Timer 2)<br>0 = Internal timer (OSC/12 or OSC/2 in baud rate generator mode.)<br>1 = External event counter (falling edge triggered). |
| CP/$\overline{RL2}$ | Capture/Reload flag. When set, captures will occur on negative transition at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow. |

The T2 Pin has another alternate function on the 87C51GB. It can be configured as a Programmable Clock Out.

### TIMER 2 CAPTURE MODE

In the capture mode there are two options selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16-bit timer on counter which, upon overflow, sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in the Timer 2 registers (TH2 and TL2) to be captured into registers RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set. The EXF2 bit, like TF2, can generate an interrupt. Figure 12 illustrates this.

### TIMER 2 AUTO-RELOAD (UP OR DOWN COUNTER)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by a bit named DCEN (Down Counter Enable) located in the SFR T2MOD (see Table 8). Upon reset the DCEN bit is set to 0 so that Timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down depending on the value of the T2EX pin.

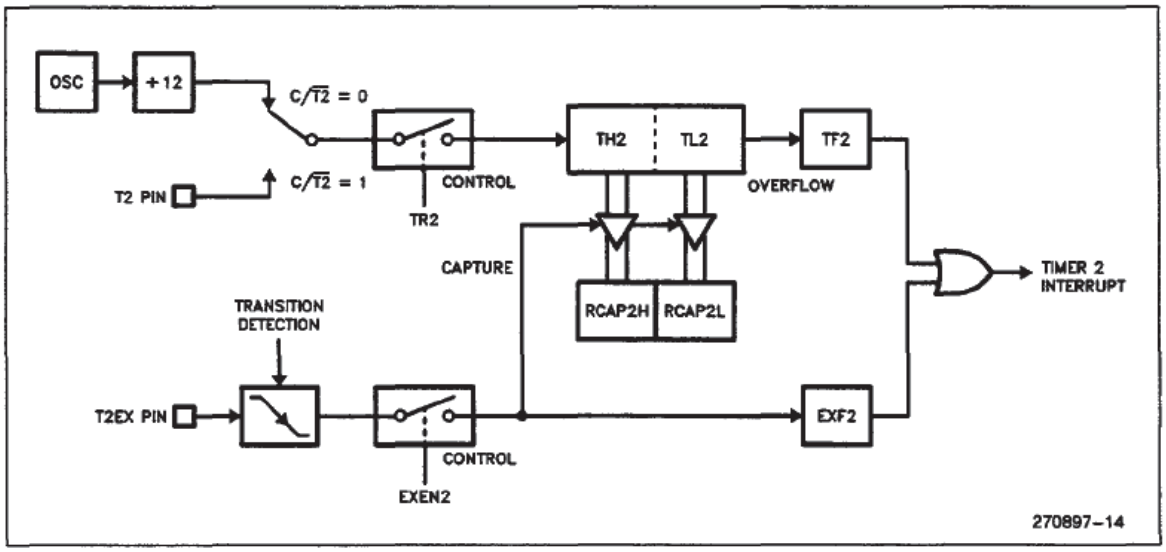



**Figure 12. Timer 2 in Capture Mode**

**Table 8. T2MOD: Timer 2 Mode Control Register**

**T2MOD** Address = 0C9H Reset Value = XXXX XX00B

Not Bit Addressable

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

| Symbol | Function |
|---|---|
| — | Not implemented, reserved for future use.* |
| T2OE | Timer 2 Output Enable bit. |
| DECN | Down Count Enable bit. When set, this allows Timer 2 to be configured as an up/down counter |

*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

In the auto-reload mode with DCEN = 0, there are two options selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in RCAP2H and RCAP2L are preset by software. If EXEN2 = 1, a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX. This transition also sets the EXF2 bit. Either the TF2 or EXF2 bit can generate the Timer 2 interrupt if it is enabled. Figure 13 shows timer 2 automatically counting up when DCEN = 0.

Setting the DCEN bit enables Timer 2 to count up or down as shown in Figure 14. In this mode the T2EX pin controls the direction of count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit which can then generate an interrupt if it is enabled. This overflow also causes the 16-bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

A logic 0 at T2EX makes Timer 2 count down. Now the timer underflows when TH2 and TL2 equal the values stored in RCAP2H and RCAP2L. The under-

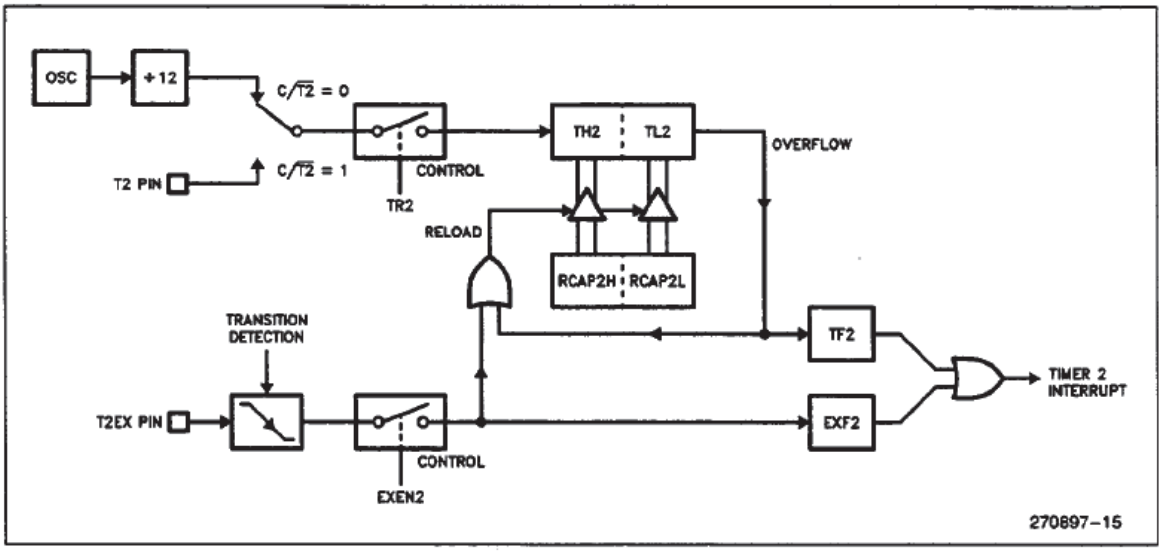



**Figure 13. Timer 2 Auto Reload Mode (DCEN = 0)**

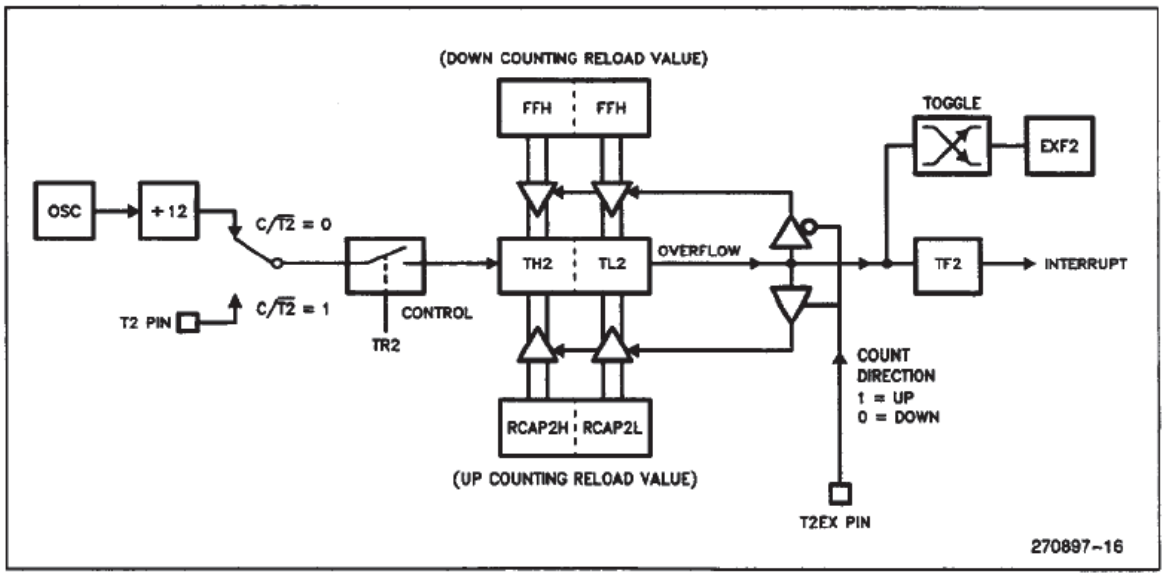



**Figure 14. Timer 2 Auto Reload Mode (DCEN = 1)**

flow sets the TF2 bit and causes 0FFFFH to be reloaded into the timer registers.

The EXF2 bit toggles whenever Timer 2 overflows or underflows. This bit can be used as a 17th bit of resolution if desired. In this operating mode, EXF2 does not generate an interrupt.

## 5.3 Programmable Clock Out

The 87C51GB has a new feature. A 50% duty cycle clock can be programmed to come out on P1.0. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed (1) to input the external clock for Timer/Counter 2, or (2) to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz at a 16 MHz operating frequency. Figure 15 shows Timer 2 in clock-out mode.

To configure the Timer/Counter 2 as a clock generator, bit C_T2 (in T2CON) must be cleared and bit T2OE (in T2MOD) must be set. Bit TR2 (in T2CON) also must be set to start the timer.

The Clock Out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L) as shown in this equation:

$$\text{Clock Out Frequency} = \frac{\text{Oscillator Frequency}}{4 \times [65536 - (\text{RCAP2H, RCAP2L})]}$$

In the Clock Out mode, Timer 2 roll-overs will not generate an interrupt. This is similar to when it is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and the clock-out frequency will be the same.

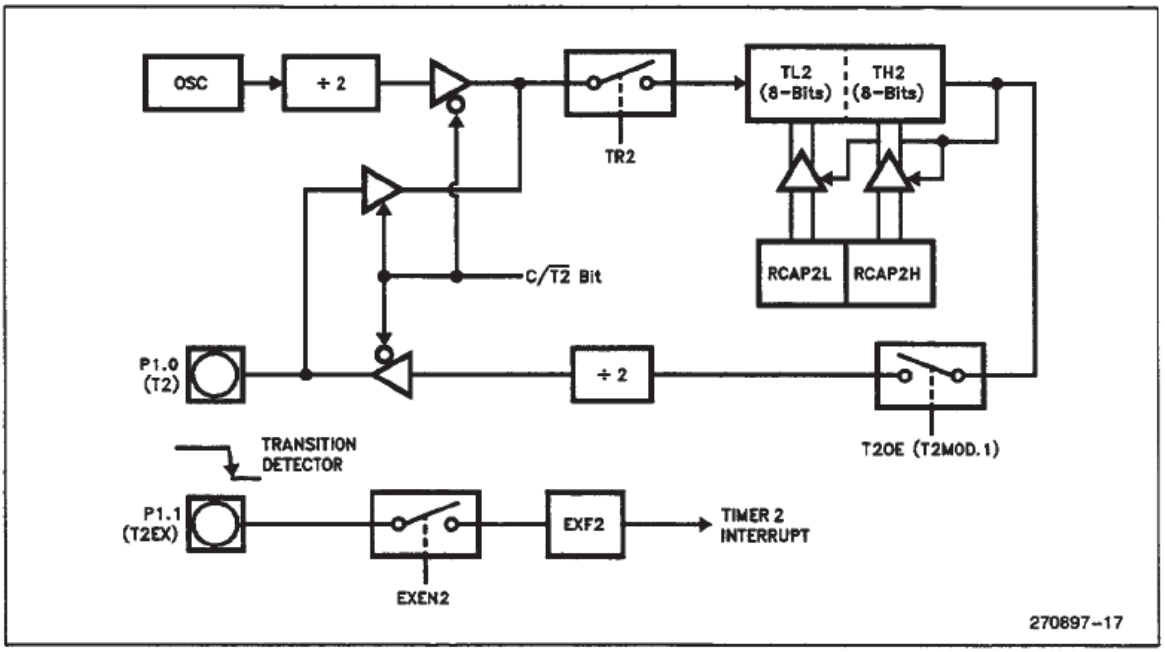



**Figure 15. Timer 2 in Clock-Out Mode**

## 6.0 A/D CONVERTER

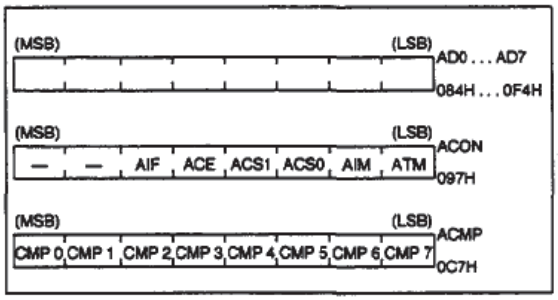
The A/D converter on the 8XC51GB consists of: 8 analog inputs (ACH0–ACH7), an external trigger input (TRIGIN), separate analog voltage supplies ($AV_{SS}$ and $AV_{REF}$), a comparison reference input (COMPREF) and internal circuitry. The internal circuitry includes: an 8 channel multiplexer, a 256 element resistive ladder, a comparator, sample-and-hold capacitor, successive approximation register, A/D trigger control, a comparison result register and 8 A/D result registers as shown in the A/D block diagram, Figure 16.

$AV_{REF}$ must be held within the tolerances stated on the 8XC51GB data sheet. The accuracy of the A/D cannot be improved, for instance, by tying $AV_{REF}$ to ½ the voltage on $V_{CC}$.

### 6.1 A/D Special Function Registers

The A/D has 10 SFRs associated with it. The SFRs are shown in Table 9.

**Table 9. A/D SFRs**




AD0 through AD7 contain the results of the 8 analog conversion. Each SFR is updated as each conversion is complete, starting with the lowest channel and ending with channel 7.

ACMP is the comparison result register. ACMP is organized differently than all the other SFRs in that CMP0 occupies the MSB and CMP7 the LSB. CMP0

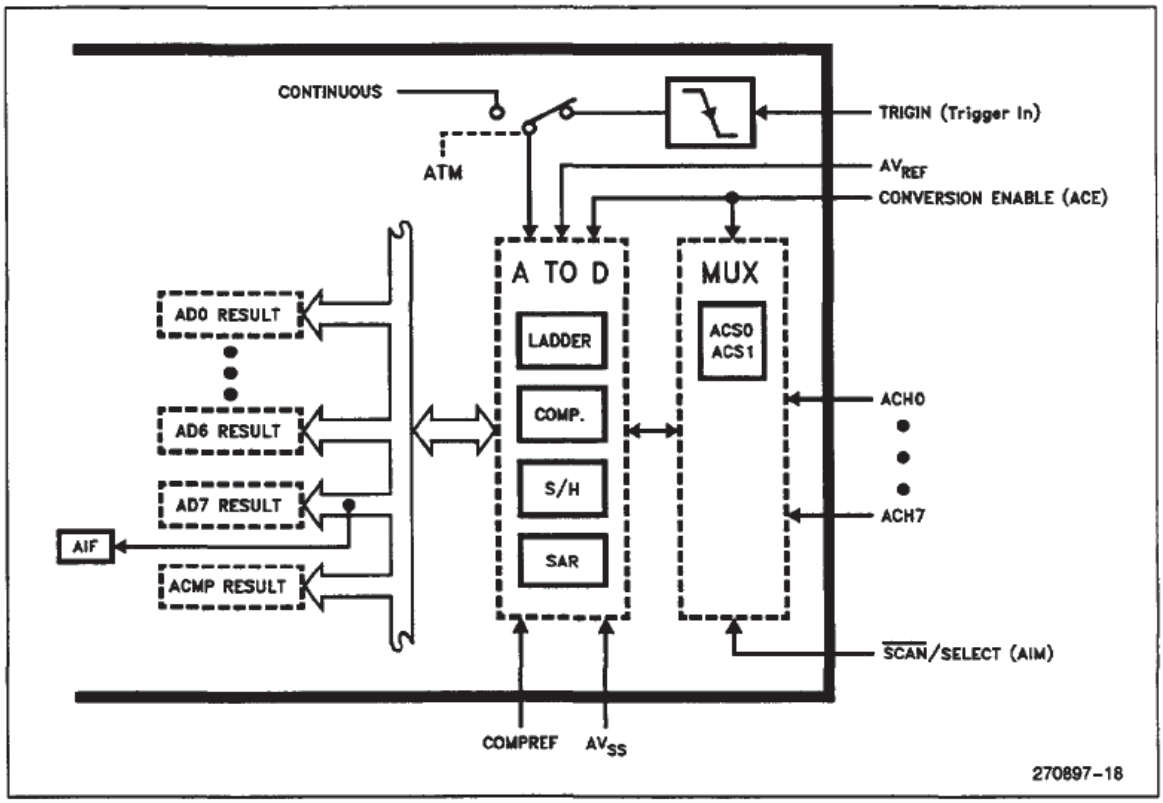



**Figure 16. A/D Block Diagram**

through CMP7 correspond to analog inputs 0 through 7. CMPn is set to a 1 if the analog input is greater than COMPREF. CMPn is cleared if the analog input is less than or equal to COMPREF.

ACON is the A/D control register and contains the A/D Interrupt Flag (AIF), A/D Conversion Enable (ACE), A/D Channel Select (ACS0 and ACS1), A/D Input Mode (AIM), and A/D Trigger Mode (ATM).

## 6.2 A/D Comparison Mode

The A/D Comparison mode is always active while the A/D converter is enabled. The Comparison mode is used to compare each analog input against an external reference voltage applied to COMPREF. Whenever the A/D converter is triggered, each bit in ACMP is updated as each analog conversion is completed, starting with channel 0 up to channel 7 regardless of whether Select or Scan mode is invoked. The comparison mode can provide a quicker "greater-than or less-than" decision than can be performed with software and it is more code efficient. It can also be used to convert the analog inputs into digital inputs with a variable threshold. If the comparison mode is not used, COMPREF should be tied to $V_{CC}$ or $V_{SS}$.

## 6.3 A/D Trigger Mode

The analog converter can be triggered either internally or externally. To enable internal trigger mode, ATM should be cleared.

When in internal trigger mode, A/D conversions begin in the machine cycle which follows the setting of the ACE bit. The lowest channel (see "A/D Input Modes" below) is converted first, followed by all the other channels in sequence. The AIF flag is set upon completion of the channel 7 conversion. AIF will flag an interrupt if the A/D interrupt is enabled. Once a conversion cycle is completed, a new cycle begins, starting with the lowest channel. If the user wishes each channel to be converted only once, the ACE bit should be cleared. Clearing ACE stops all A/D conversion activity. If a new A/D cycle begins, the result of the previous conversion will be overwritten.

In external mode, the A/D conversions begin when a falling edge is detected at the TRIGIN pin. There is no edge detector on the TRIGIN pin; is it sampled once every machine cycle.

A negative edge is recognized when TRIGIN is high in one machine cycle and low in the next. For this reason, TRIGIN should be held high for at least one machine cycle and low for one machine cycle. Once the falling edge is detected, the A/D conversions begin on the next machine cycle and complete when channel 7 is converted. After channel 7 is converted, AIF is set and the conversions halt until another trigger is detected while ACE = 1. External triggers are ignored while a conversion cycle is in progress.

## 6.4 A/D Input Modes

The 8XC51GB has two input modes: Scan mode and Select mode. Clearing AIM places the 8XC51GB in Scan mode. In Scan mode the analog conversions occur in the sequence ACH0, ACH1, ACH2, ACH3, ACH4, ACH5, ACH6, and ACH7. The result of each analog conversion is placed in the corresponding analog result register: AD0, AD1, AD2, AD3, AD4, AD5, AD6, and AD7.

Setting AIM activates Select mode. In Select mode, one of the lower 4 analog inputs (ACH0–ACH3) is converted four times. After the first four conversions are complete the cycle continues with ACH4 through ACH7. The results of the first four conversion are placed in the lower four result registers (AD0 through AD3). The rest of the conversions are placed in their matching result register. ACS0 and ACS1 determine which analog inputs are used as shown in Table 10.

**Table 10. A/D Channel Selection**

| ACS1 | ACS0 | Selected Channel |
|---|---|---|
| 0 | 0 | ACH0 |
| 0 | 1 | ACH1 |
| 1 | 0 | ACH2 |
| 1 | 1 | ACH3 |

## 6.5 Using the A/D with Fewer than 8 Inputs

There are several options for a user who wishes to convert fewer than eight analog input channels. If time is not critical the user can simply wait for the A/D interrupt to be generated by the AIF bit after channel 7 is converted and can ignore the results for unused channels. If a user needs to know the results of a conversion immediately after it occurs, a timer should be used to generate an interrupt. The amount of time required for each A/D conversion is specified in the 8XC51GB data sheet. The user could also periodically poll the result registers, provided he or she is looking only for a change in the analog voltage. Using the Select mode (see above) does not reduce the time required for a conversion cycle but will convert a given channel more frequently.

## 6.6 A/D in Power Down

The A/D on the 8XC51GB contains circuitry that limits the amount of current dissipated during Power Down mode to leakage current only. For this circuitry to function properly, $AV_{REF}$ should be tied to $V_{CC}$ during power down. The $I_{PD}$ specification in the data sheet includes the current for the entire chip. While $AV_{REF}$ is tied to $V_{CC}$ during Power Down, the voltage may be reduced to the minimum voltage as shown in the data sheet.

## 7.0 PROGRAMMABLE COUNTER ARRAY

Programmable Counter Arrays (PCAs) provide more timing capabilities with less CPU intervention than the standard timer/counters. Their advantages include reduced software overhead and improved accuracy. For example, a PCA can provide better resolution than Timers 0, 1, and 2 because the PCA clock rate can be three times faster. A PCA can also perform many tasks that these hardware timers cannot (i.e. measure phase differences between signals or generate PWMs).

The 8XC51GB has two PCAs called PCA and PCA1. The following text and figures address only PCA but are also applicable to PCA1 with the following exceptions:

1. PCA1, Module 4 does not support the Watchdog Timer
2. All the SFRs and bits have 1s added to their names (see Table 11).
3. Port 4 is the interface for PCA1:

| | |
|---|---|
| P4.2 | ECI1 |
| P4.3 | C1EX1 |
| P4.4 | C1EX2 |
| P4.5 | C1EX2 |
| P4.6 | C1EX3 |
| P4.7 | C1EX4 |

**Table 11. PCA and PCA1 SFRs**

| PCA | PCA1 |
|---|---|
| SFRs: | |
| CCON | C1CON |
| CMOD | C1MOD |
| CCAPM0 | C1CAPM0 |
| CCAPM1 | C1CAPM1 |
| CCAPM2 | C1CAPM2 |
| CCAPM3 | C1CAPM3 |
| CCAPM4 | C1CAPM4 |
| CL | CL1 |
| CCAP0L | C1CAP0L |
| CCAP1L | C1CAP1L |
| CCAP2L | C1CAP2L |
| CCAP3L | C1CAP3L |
| CCAP4L | C1CAP4L |
| CH | CH1 |
| CCAP0H | C1CAP0H |
| CCAP1H | C1CAP1H |
| CCAP2H | C1CAP2H |
| CCAP3H | C1CAP3H |
| CCAP4H | C1CAP4H |
| BITS: | |
| ECI | ECI1 |
| CEX0 | C1EX0 |
| CEX1 | C1EX1 |
| CEX2 | C1EX2 |
| CEX3 | C1EX3 |
| CEX4 | C1EX4 |
| CCF0 | C1CF0 |
| CCF1 | C1CF1 |
| CCF2 | C1CF2 |
| CCF3 | C1CF3 |
| CCF4 | C1CF4 |
| CR | CR1 |
| CF | CF1 |

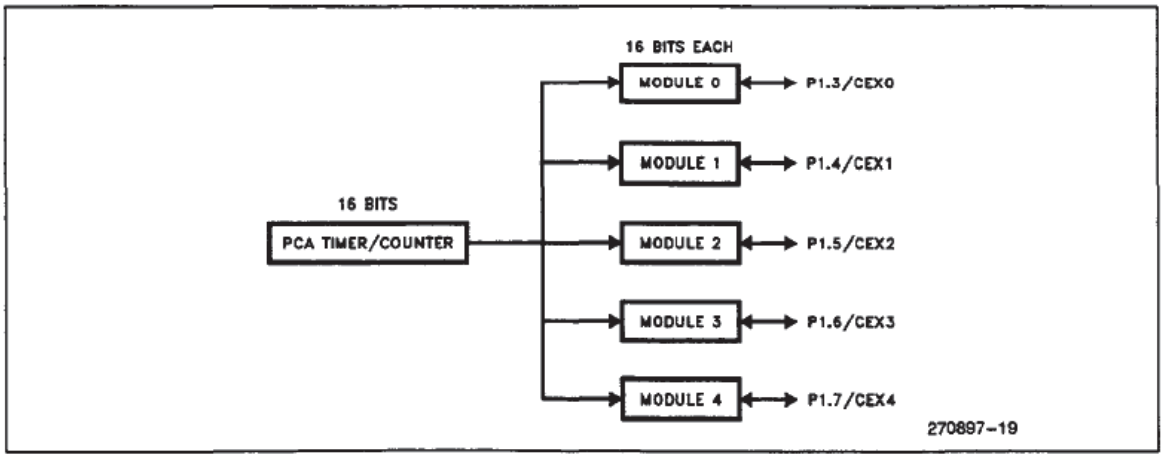



**Figure 17. PCA Block Diagram**

4. There has been one additional bit added to C1CON to allow both PCAs to be enabled simultaneously.

   The bit is called CRE and occupies bit position 5 of C1CN. Its bit address is 0EDH. When CRE is set, both CR and CR1 must be set to enable PCA1.

Each PCA consists of a 16-bit timer/counter and five 16-bit compare/capture modules as shown in Figure 17. The PCA timer/counter serves as a common time base for the five modules and is the only timer which can service the PCA. Its clock input can be programmed to count any one of the following signals:

Oscillator frequency / 12

Oscillator frequency / 4

Timer 0 overflow

External input on ECI (P1.2).

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

rising and/or falling edge capture

software timer

high speed output

pulse width modulator.

Module 4 can also be programmed as a watchdog timer.

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

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

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

## 7.1 PCA Timer/Counter

The PCA has a free-running 16-bit timer/counter consisting of registers CH and CL (the high and low bytes of the count value). These two registers can be read or written to at any time. Reading the PCA timer as a full 16-bit value simultaneously requires using one of the PCA modules in the capture mode and toggling a port pin in software.

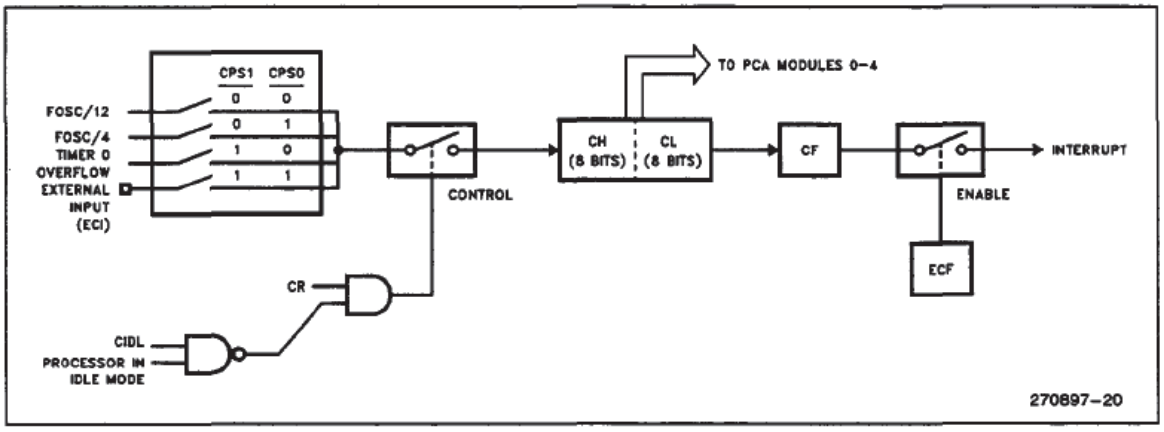



**Figure 18. PCA Timer/Counter**

The clock input can be selected from the following four modes:

**Oscillator frequency / 12:**

The PCA timer increments once per machine cycle. With a 16 MHz crystal, the timer increments every 750 ns.

**Oscillator frequency / 4:**

The PCA timer increments three times per machine cycle. With a 16 MHz crystal, the timer increments every 250 ns.

**Timer 0 overflows:**

The PCA timer increments whenever Timer 0 overflows. This mode allows a programmable input frequency to the PCA.

**External input:**

The PCA timer increments when a 1-to-0 transition is detected on the ECI pin (P1.2). The maximum input frequency in this mode is oscillator frequency / 8.

The mode register CMOD (Table 12) contains the Count Pulse Select bits (CPS1 and CPS0) to specify the clock input. This register also contains the ECF bit which enables the PCA counter overflow to generate the PCA interrupt. In addition, the user has the option of turning off the PCA timer during Idle Mode by setting the Counter Idle bit (CIDL). This can further reduce power consumption by an additional 30%.

The CCON (Table 13) register contains two more bits which are associated with the PCA timer/counter. The CF bit gets set by hardware when the counter overflows, and the CR bit is set or cleared to turn the counter on or off.

**Table 12. CMOD: PCA Counter Mode Register**

**CMOD** Address = 0D9H Reset Value = 00XX X000B

Not Bit Addressable

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

Bit

| Symbol | Function |
|---|---|
| CIDL | Counter Idle control: CIDL = 0 programs the PCA Counter to continue functioning during idle Mode. CIDL = 1 programs it to be gated off during idle. |
| WDTE | Watchdog Timer Enable: WDTE = 0 disables Watchdog Timer function on PCA Module 4. WDTE = 1 enables it. |
| — | Not implemented, reserved for future use.* |
| CPS1 | PCA Count Pulse Select bit 1. |
| CPS0 | PCA Count Pulse Select bit 0. |

| CPS1 | CPS0 | Selected PCA Input** |
|---|---|---|
| 0 | 0 | Internal clock, Fosc÷12 |
| 0 | 1 | Internal clock, Fosc÷4 |
| 1 | 0 | Timer 0 overflow |
| 1 | 1 | External clock at ECI/P1.2 pin (max. rate = Fosc÷8) |

| | |
|---|---|
| ECF | PCA Enable Counter Overflow interrupt: ECF = 1 enables CF bit in CCON to generate an interrupt. ECF = 0 disables that function of CF. |

**NOTE:**
*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.
**Fosc = oscillator frequency

**Table 13. CCON: PCA Counter Control Register**

**CCON** Address = 0D8H Reset Value = 00X0 0000B

Bit Addressable

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

| Symbol | Function |
|---|---|
| CF | PCA Counter Overflow flag. Set by hardware when the counter rolls over. CF flags an interrupt if bit ECF in CMOD is set. CF may be set by either hardware or software but can only be cleared by software. |
| CR | PCA Counter Run control bit. Set by software to turn the PCA counter on. Must be cleared by software to turn the PCA counter off. |
| — | Not implemented, reserved for future use*. |
| CCF4 | PCA Module 4 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software. |
| CCF3 | PCA Module 3 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software. |
| CCF2 | PCA Module 2 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software. |
| CCF1 | PCA Module 1 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software. |
| CCF0 | PCA Module 0 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software. |

***NOTE:**
User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

### READING THE PCA TIMER

Some applications may require that the full 16-bit PCA timer value be read simultaneously. Since the timer consists of two 8-bit registers (CH, CL), it would normally take two MOV instructions to read the whole timer value. An invalid read could occur if the registers rolled over in between the execution of the two MOVs.

However, with the PCA Capture Mode the 16-bit timer value can be loaded into the capture registers by toggling a port pin. For example, configure Module 0 to capture falling edges and initialize P1.3 to be high. Then, when the user wants to read the PCA timer, clear P1.3 and the full 16-bit timer value will be saved in the capture registers. It's still optional whether the user wants to generate an interrupt with the capture.

## 7.2 Compare/Capture Modules

Each of the five compare/capture modules has six possible functions it can perform:

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

In addition, module 4 can be used as a Watchdog Timer. The modules can be programmed in any combination of the different modes.

Each module has a mode register called CCAPMn (n = 0, 1, 2, 3, or 4) to select which function it will perform. The ECCFn bit enables the PCA interrupt when a module's event flag is set. The event flags (CCFn) are located in the CCON register and get set when a capture event, software timer, or high speed output event occurs for a given module.

Each module also has a pair of 8-bit compare/capture registers (CCAPnH and CCAPnL) associated with it. These registers store the time when a capture event occurred or when a compare event should occur. For the PWM mode, the high byte register CCAPnH controls the duty cycle of the waveform.

## 7.3 PCA Capture Mode

Both positive and negative transitions can trigger a capture with the PCA. This gives the PCA the flexibility to measure periods, pulse widths, duty cycles, and phase differences on up to five separate inputs. Setting the CAPPn and/or CAPNn bits in the CCAPMn mode register (Table 14) selects the input trigger—positive and/or negative transition—for module n. Refer to Figure 19.

Table 15 shows the combinations of bits in the CCAPMn register that are valid and have a defined function. Invalid combinations will produce undefined results.

**Table 14. CCAPMn: PCA Modules Compare/Capture Registers**

**CCAPMn** Address (n = 0–4): CCAPM0 0DAH, CCAPM1 0DBH, CCAPM2 0DCH, CCAPM3 0DDH, CCAPM4 0DEH — Reset Value = X000 0000B

Not Bit Addressable

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

| Symbol | Function |
|---|---|
| — | Not implemented, reserved for future use*. |
| ECOMn | Enable Comparator. ECOMn = 1 enables the comparator function. |
| CAPPn | Capture Positive, CAPPn = 1 enables positive edge capture. |
| CAPNn | Capture Negative, CAPNn = 1 enables negative edge capture. |
| MATn | Match. When MATn = 1, a match of the PCA counter with this module's compare/capture register causes the CCFn bit in CCON to be set, flagging an interrupt. |
| TOGn | Toggle. When TOGn = 1, a match of the PCA counter with this module's compare/capture register causes the CEXn pin to toggle. |
| PWMn | Pulse Width Modulation Mode. PWMn = 1 enables the CEXn pin to be used as a pulse width modulated output. |
| ECCFn | Enable CCF interrupt. Enables compare/capture flag CCFn in the CCON register to generate an interrupt. |

**NOTE:**
*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case,the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

**Table 15. PCA Module Modes (CCAPMn Register)**

| — | ECOMn | CAPPn | CAPNn | MATn | TOGn | PWMn | ECCFn | Module Function |
|---|---|---|---|---|---|---|---|---|
| X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | No operation |
| X | X | 1 | 0 | 0 | 0 | 0 | X | 16-bit capture by a postive-edge trigger on CEXn |
| X | X | 0 | 1 | 0 | 0 | 0 | X | 16-bit capture by a negative-edge trigger on CEXn |
| X | X | 1 | 1 | 0 | 0 | 0 | X | 16-bit capture by a transition on CEXn |
| X | 1 | 0 | 0 | 1 | 0 | 0 | X | 16-bit Software Timer |
| X | 1 | 0 | 0 | 1 | 1 | 0 | X | 16-bit High Speed Output |
| X | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 8-bit PWM |
| X | 1 | 0 | 0 | 1 | x | 0 | x | Watchdog Timer |

X = Don't Care

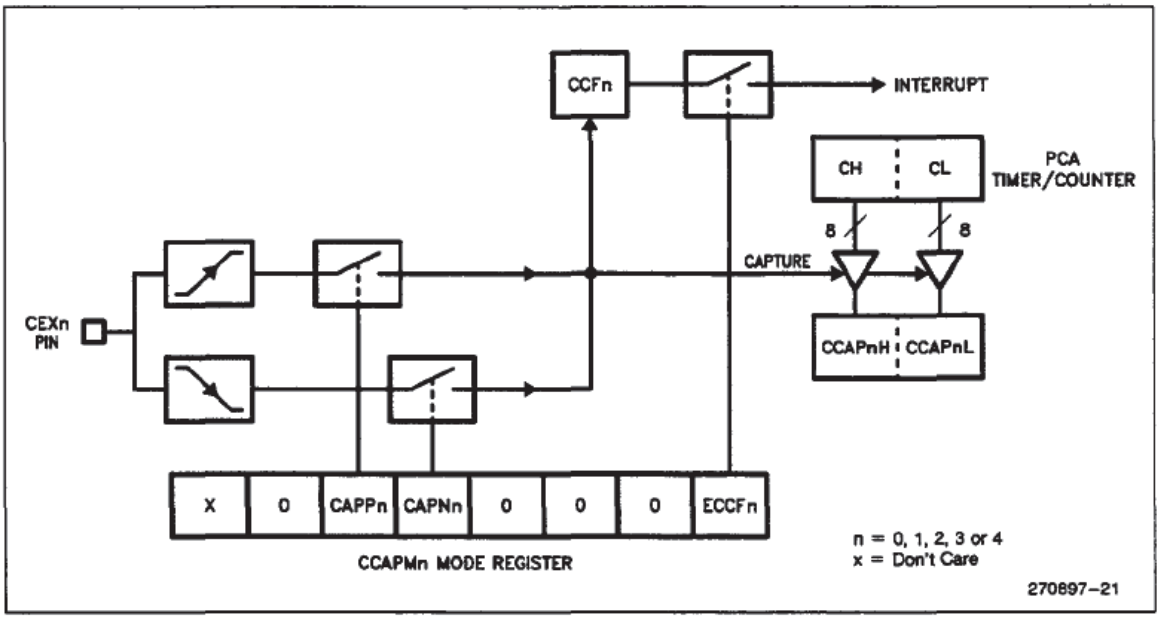



**Figure 19. PCA 16-Bit Capture Mode**

The external input pins CEX0 through CEX4 are sampled for a transition. When a valid transition is detected (positive and/or negative edge), hardware loads the 16-bit value of the PCA timer (CH, CL) into the module's capture registers (CCAPnH, CCAPnL). The resulting value in the capture registers reflects the PCA timer value at the time a transition was detected on the CEXn pin.

Upon a capture, the module's event flag (CCFn) in CCON is set, and an interrupt is flagged if the ECCFn bit in the mode register CCAPMn is set. The PCA interrupt will then be generated if it is enabled. Since the hardware does not clear an event flag when the interrupt is vectored to, the flag must be cleared in software.

In the interrupt service routine, the 16-bit capture value must be saved in RAM before the next capture event occurs. A subsequent capture on the same CEXn pin will write over the first capture value in CCAPnH and CCAPnL.

The time it takes to service this interrupt routine determines the resolution of back-to-back events with the same PCA module. To store two 8-bit registers and clear the event flags takes at least 9 machine cycles. That includes the call to the interrupt routine. At 12 MHz, this routine would take less than 10 $\mu$s. However, depending on the frequency and interrupt latency, the resolution will vary with each application.

## 7.4 Software Timer Mode

In most applications a software timer is used to trigger interrupt routines which must occur at periodic intervals. The user preloads a 16-bit value in a module's compare registers. When a match occurs between this compare value and the PCA timer value, an event flag is set and an interrupt can then be generated.

In the PCA compare mode, the 16-bit value of the PCA timer is compared with a 16-bit value pre-loaded in the module's compare registers (CCAPnH, CCAPnL) as seen in Figure 20. The comparison occurs three times per machine cycle in order to recognize the fastest possible clock input (i.e. $\frac{1}{4} \times$ oscillator frequency). Setting the ECOMn bit in the mode register CCAPMn enables the comparator function.

For the Software Timer mode, the MATn bit also needs to be set. When a match occurs between the PCA timer and the compare registers, a match signal is generated and the module's event flag (CCFn) is set. An interrupt is then flagged if the ECCFn bit is set. The PCA interrupt is generated only if it has been properly enabled. Software must clear the event flag before the next interrupt will be flagged.

During the interrupt routine, a new 16-bit compare value can be written to the compare registers (CCAPnH and CCAPnL). Notice, however, that a write to CCAPnL clears the ECOMn bit which temporarily disables the comparator function while these registers are being updated so an invalid match does not occur. A write to CCAPnH sets the ECOMn bit and re-enables the comparator. For this reason, user software should write to CCAPnL first, then CCAPnH.

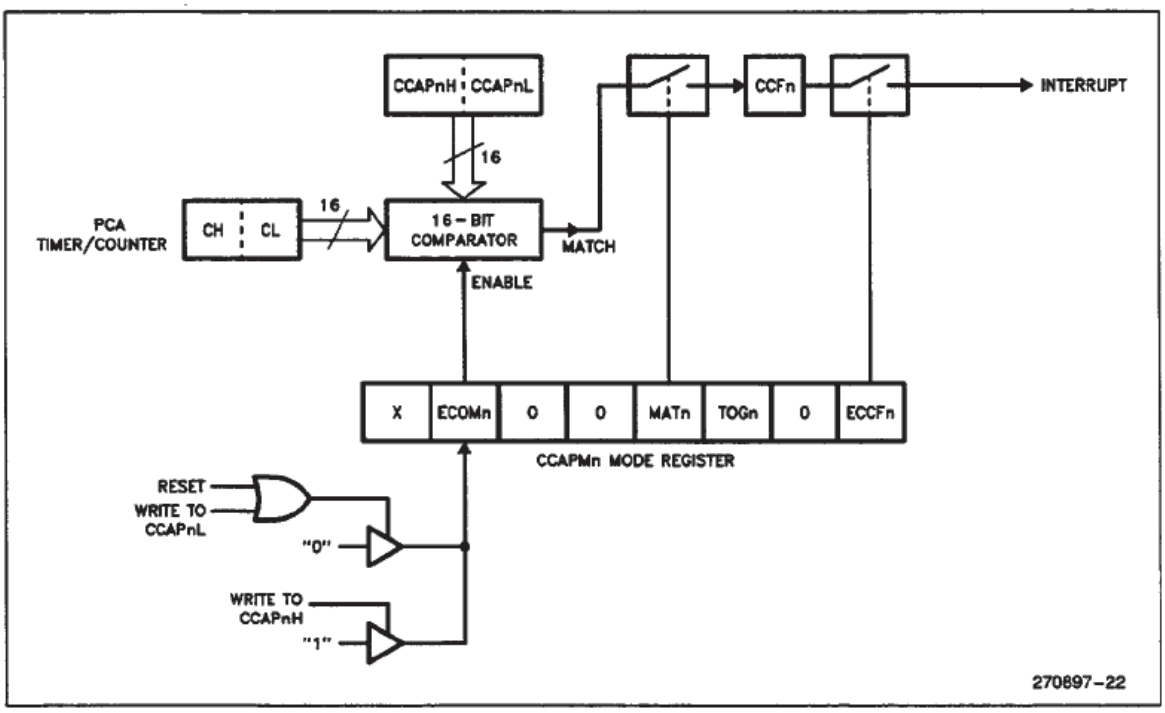



**Figure 20. PCA 16-Bit Comparator Mode: Software Timer**

## 7.5 High Speed Output Mode

The High Speed Output (HSO) mode toggles a CEXn pin when a match occurs between the PCA timer and a pre-loaded value in a module's compare registers. For this mode, the TOGn bit needs to be set in addition to the ECOMn and MATn bits in the CCAPMn mode register. By setting or clearing the pin in software, the user can select whether the CEXn pin will change from a logical 0 to a logical 1 or vice versa. The user also has the option of flagging an interrupt when a match event occurs by setting the ECCFn bit. See Figure 21.

The HSO mode is more accurate than toggling port pins in software because the toggle occurs before branching to an interrupt. That is, interrupt latency will not effect the accuracy of the output. In fact, the interrupt is optional. Only if the user wants to change the time for the next toggle is it necessary to update the compare registers. Otherwise, the next toggle will occur when the PCA timer rolls over and matches the last compare value.

Without any CPU intervention, the fastest waveform the PCA can generate with the HSO mode is a 30.5 Hz signal at 16 MHz.

## 7.6 Watchdog Timer Mode

A Watchdog Timer is a circuit that automatically invokes a reset unless the system being watched sends regular hold-off signals to the Watchdog. These circuits are used in applications that are subject to electrical noise, power glitches, electrostatic discharges, etc., or where high reliability is required.

The Watchdog Timer function is only available on PCA Module 4. If a Watchdog Timer is not needed, Module 4 can still be used in other modes.

As a Watchdog timer, every time the count in the PCA timer matches the value stored in module 4's compare registers, an internal reset is generated (see Figure 22). The bit that selects this mode is WDTE in the CMOD register. Module 4 must be set up in either compare mode as a "Software Timer" or High Speed Output.

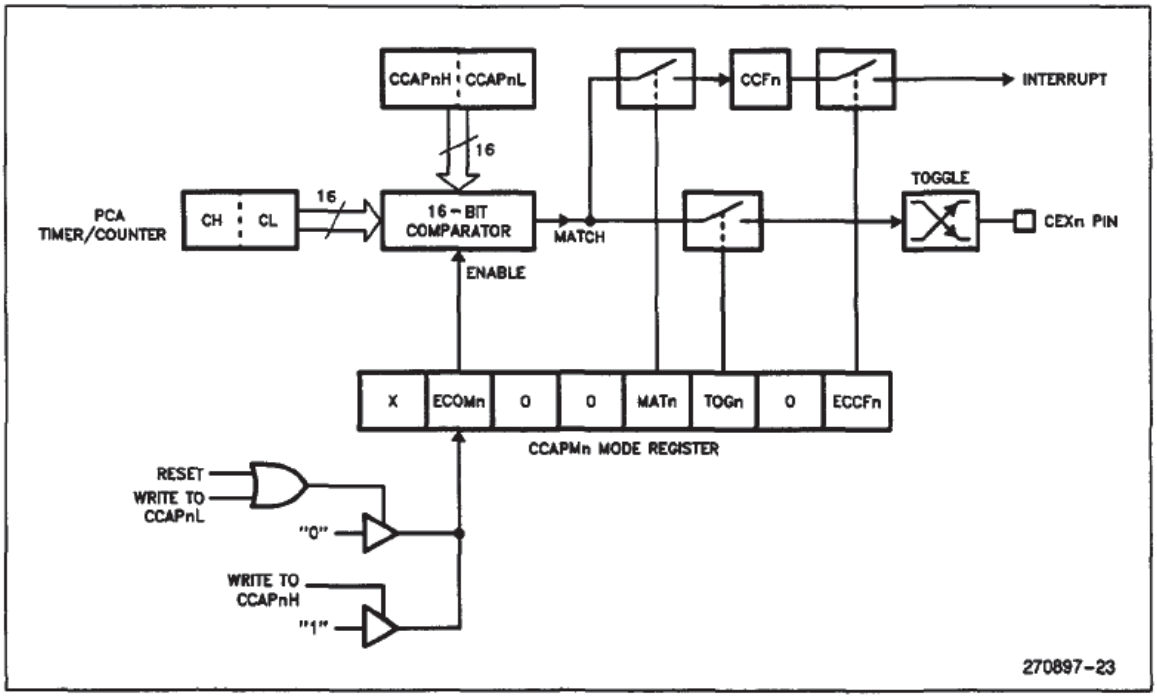



**Figure 21. PCA 16-Bit Comparator Mode: High Speed Output**

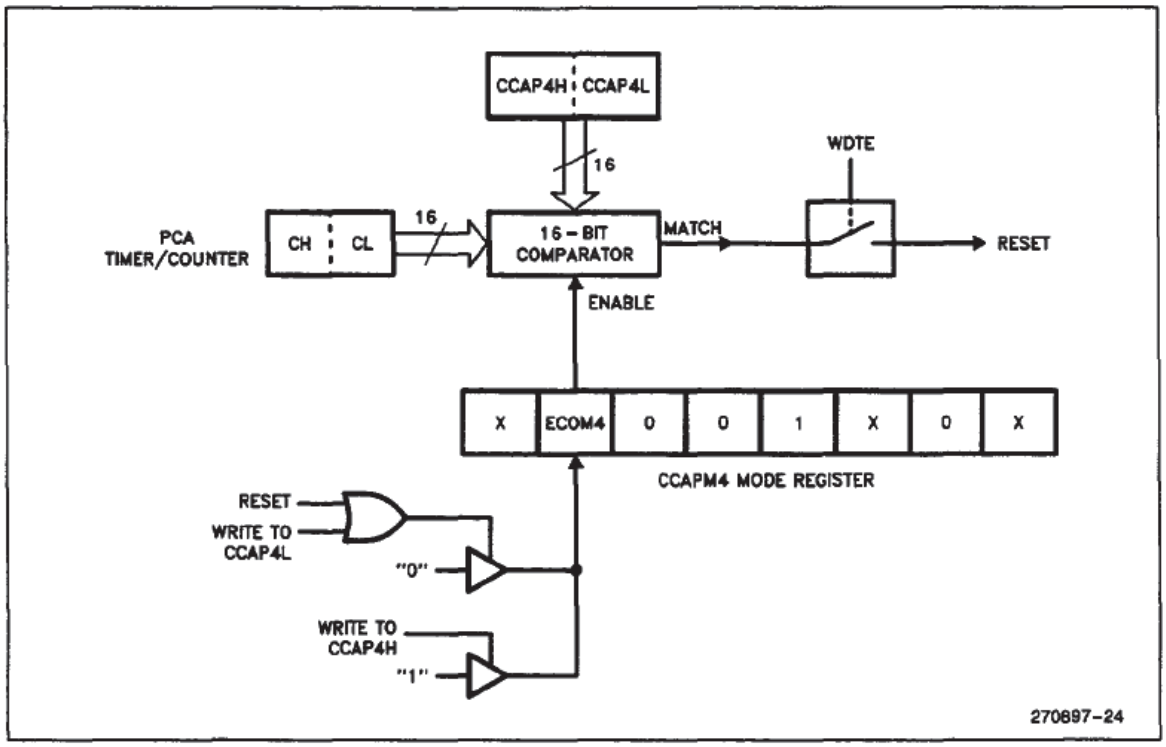



**Figure 22. Watchdog Timer Mode**

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

1. periodically change the compare value so it will never match the PCA timer,
2. periodically change the PCA timer value so it will never match the compare value,
3. disable the Watchdog by clearing the WDTE bit before a match occurs and then later re-enable it.

The first two options are more reliable because the Watchdog Timer is never disabled as in option #3. The second option is not recommended if other PCA modules are being used since this timer is the time base for all five modules. Thus, in most applications the first solution is the best option.

The watchdog routine should not be part of an interrupt service routine. Why? Because if the program counter goes astray and gets stuck in an infinite loop, interrupts will still be serviced, and the watchdog will not reset the controller. Thus, the purpose of the watchdog would be defeated. Instead, call this subroutine from the main program within 65536 counts of the PCA timer.

## 7.7 Pulse Width Modulator Mode

Any or all of the five PCA modules can be programmed to be a Pulse Width Modulator. The PWM output can be used to convert digital data to an analog signal by simple external circuitry. The frequency of the PWM depends on the clock source for the PCA timer. With a 16 MHz crystal the maximum frequency of the PWM waveform is 15.6 KHz. Table 16 shows the various frequencies that are possible.

**Table 16. PWM Frequencies**

| PCA Timer Mode | PWM Frequency | |
|---|---|---|
| | 12 MHz | 16 MHz |
| 1/12 Osc. Frequency | 3.9 KHz | 5.2 KHz |
| 1/4 Osc. Frequency | 11.8 KHz | 15.6 KHz |
| Timer 0 Overflow:<br>8-bit<br>16-bit<br>8-bit Auto-Reload | <br>15.5 Hz<br>0.06 Hz<br>3.9 KHz to 15.3 Hz | <br>20.3 Hz<br>0.08 Hz<br>5.2 KHz to 20.3 Hz |
| External Input (Max) | 5.9 KHz | 7.8 KHz |

For this mode, the ECOMn bit and the PWMn bits in the CCAPMn mode register need to be set. The PCA generates 8-bit PWMs by comparing the low byte of the PCA timer (CL) with the low byte of the module's compare registers (CCAPnL). When CL < CCAPnL the output is low. When CL > CCAPnL the output is high. Refer to Figure 23.

The value in CCAPnL controls the duty cycle of the waveform. To change the value in CCAPnL without output glitches, the user must write to the high byte register (CCAPnH). This value is then shifted by hardware into CCAPnL when CL rolls over from 0FFH to 00H which corresponds to the next period of the output.

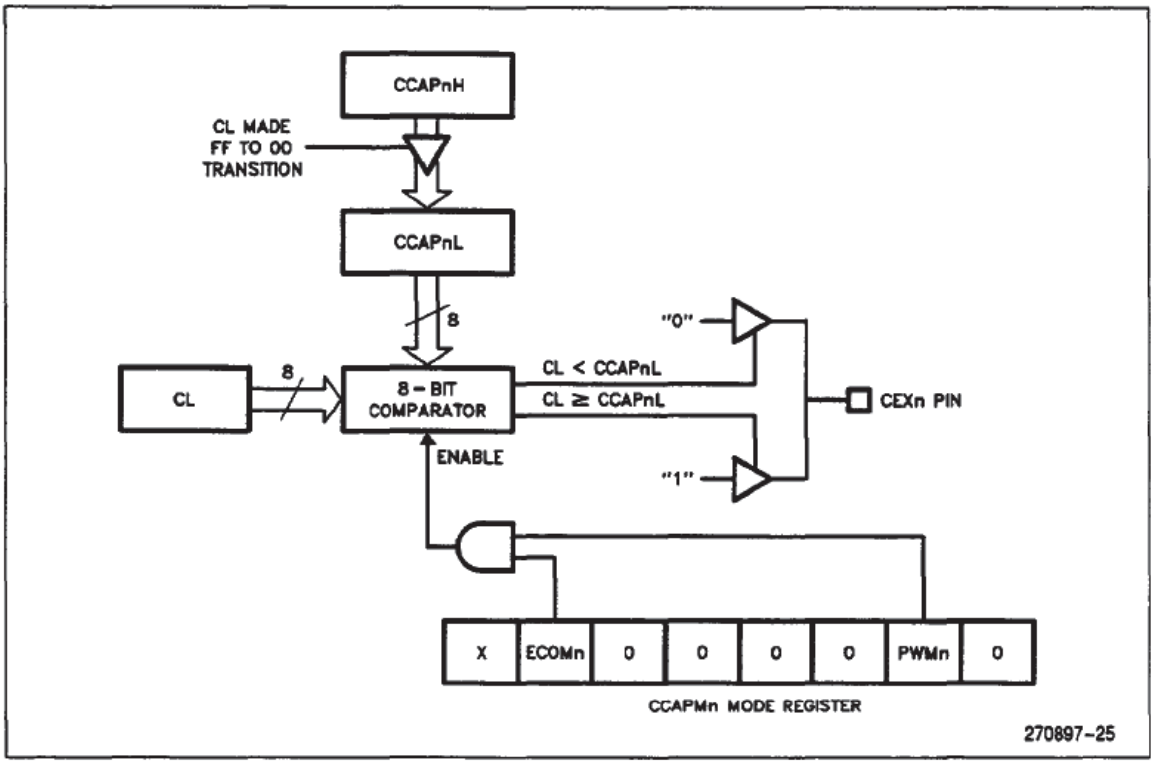



**Figure 23. PCA 8-Bit PWM Mode**

CCAPnH can contain any integer from 0 to 255 to vary the duty cycle from a 100% to 0.4%. A 0% duty cycle can be obtained by writing directly to the port pin with the CLR bit instruction. To calculate the CCAPnH value for a given duty cycle, use the following equation:

$$CCAPnH = 256 \times (1 - \text{Duty Cycle})$$

where CCAPnH is an 8-bit integer and Duty Cycle is expressed as a fraction. See Figure 24.

## 8.0 SERIAL PORT

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register. (However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost).

The serial port receive and transmit registers are both accessed through Special Function Register SBUF. Actually, SBUF is two separate registers, a transmit buffer and a receive buffer. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

The serial port control and status register is the Special Function Register SCON (Table 17). This register contains the mode selection bits (SM0 and SM1); the SM2 bit for the multiprocessor modes; the Receive Enable bit (REN); the 9th data bit for transmit and receive (TB8 and RB8); and the serial port interrupt bits (TI and RI).

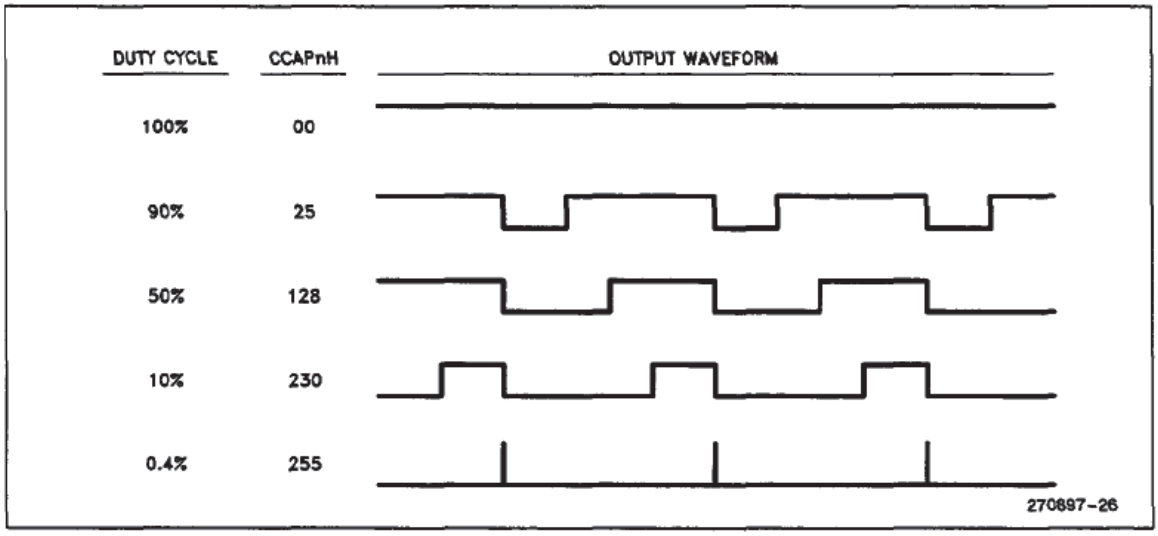



**Figure 24. CCAPnH Varies Duty Cycle**

**Table 17. SCON: Serial Port Control Register**

**SCON** Address = 98H Reset Value = 0000 0000B

Bit Addressable

| | SM0/FE | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |
|---|---|---|---|---|---|---|---|---|
| Bit: | 7 (SMOD0=0/1)* | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| FE | Framing Error bit. This bit is set by the receiver when an invalid stop bit is detected. The FE bit is not cleared by valid frames but should be cleared by software. The SMOD0* bit must be set to enable access to the FE bit. |
| SM0 | Serial Port Mode Bit 0, (SMOD0 must = 0 to access SM0) |
| SM1 | Serial Port Mode Bit 1 |

| SM0 | SM1 | Mode | Description | Baud Rate** |
|---|---|---|---|---|
| 0 | 0 | 0 | shift register | $F_{OSC}/12$ |
| 0 | 1 | 1 | 8-bit UART | variable |
| 1 | 0 | 0 | 9-bit UART | $F_{OSC}/64$ or $F_{OSC}/32$ |
| 1 | 1 | 3 | 9-bit UART | variable |

| | |
|---|---|
| SM2 | Enables the Automatic Address Recognition feature in Modes 2 or 3. If SM2 = 1 then RI will not be set unless the received byte is a Given or Broadcast Address. In Mode 1, if SM2 = 1 then RI will not be activated unless a valid stop bit was received, and the received byte is a Given or Broadcast Address. In Mode 0, SM2 should be 0. |
| REN | Enables serial reception. Set by software to enable reception. Cleared by software to disable reception. |
| TB8 | The 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired. |
| RB8 | In modes 2 and 3, the 9th data bit that was received. In Mode 1 if SM2=0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used. |
| TI | Transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software. |
| RI | Receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0 or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software. |

**NOTE:**
*SMOD0 is located at PCON6.
**$F_{OSC}$ = oscillator frequency

The serial port can operate in 4 modes:

Mode 0: Shift Register, fixed frequency

Mode 1: 8-Bit UART, variable frequency

Mode 2: 9-Bit UART, fixed frequency

Mode 3: 9-Bit UART, variable frequency

The baud rate in some modes is fixed and in others is generated by Timer 1 or Timer 2.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1.

Mode 0: Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at 1/12 the oscillator frequency.

Mode 1: 10 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in SCON. The baud rate in Mode 1 is variable: you can use either Timer 1 to generate baud rates and/or Timer 2 to generate baud rates. Figure 25 shows the mode 1 Data Frame.

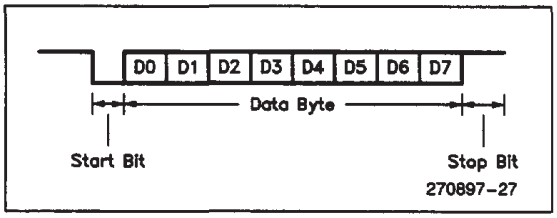



**Figure 25. Mode 1 Data Frame**

Mode 2: 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On Transmit, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P in the PSW) could be moved into TB8. On receive, the 9th data bit goes into RB8 in SCON, while the stop bit is ignored. (The validity of the stop bit can be checked with Framing Error Detection.) The baud rate is programmable to either 1/32 or 1/64 the oscillator frequency. See Figure 26.

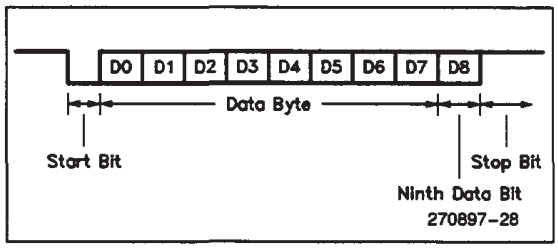



**Figure 26. Mode 2 Data Frame**

Mode 3: 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except the baud rate. The baud rate in Mode 3 is variable: you can use Timer 1 and/or Timer 2 to generate baud rates. See Figure 27.

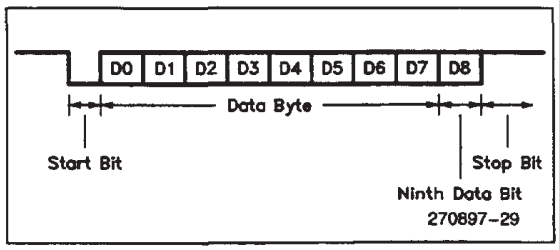



**Figure 27. Mode 3 Data Frame**

## 8.1 Framing Error Detection

Framing Error Detection allows the serial port to check for valid stop bits in modes 1, 2, or 3. A missing stop bit can be caused, for example, by noise on the serial lines, or transmission by two CPUs simultaneously.

If a stop bit is missing, a Framing Error bit (FE) is set. The FE bit can be checked in software after each reception to detect communication errors. Once set, the FE bit must be cleared in software. A valid stop bit will not clear FE.

The FE bit is located in SCON and shares the same bit address as SM0. Control bit SMOD0 in the PCON register determines whether the SM0 or FE bit is accessed. If SMOD0 = 0, then accesses to SCON.7 are to SM0. If SMOD0 = 1, then accesses to SCON.7 are to FE.

## 8.2 Multiprocessor Communications

Modes 2 and 3 provide a 9-bit mode to facilitate multiprocessor communication. The 9th bit allows the controller to distinguish between address and data bytes. The 9th bit is set to 1 for address and set to 0 for data bytes. When receiving, the 9th bit goes into RB8 in SCON. When transmitting, the ninth bit TB8 is set or cleared in software.

The serial port can be programmed such that when the stop bit is received the serial port interrupt will be activated only if the received byte is an address byte (RB8 = 1). This feature is enabled by setting the SM2 bit in SCON. A way to use this feature in multiprocessor systems is as follows.

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. Remember, an address byte has its 9th bit set to 1, whereas a data byte has its 9th bit set to 0. All the slave processors should have their SM2 bits set to 1 so they will only be interrupted by an address byte. In fact, the 8XC51GB has an Automatic Address Recognition feature which allows only the addressed slave to be interrupted. That is, the address comparison occurs in hardware, not software. (On the 8051 serial port, an address byte interrupts all slaves for an address comparison.)

The addressed slave then clears its SM2 bit and prepares to receive the data bytes that will be coming. The other slaves are unaffected by these data bytes. They are still waiting to be addressed since their SM2 bits are all set.

## 8.3 Automatic Address Recognition

Automatic Address Recognition reduces the CPU time required to service the serial port. Since the CPU is only interrupted when it receives its own address, the software overhead to compare addresses is eliminated. Automatic address recognition is enabled by setting the SM2 bit in SCON. With this feature enabled in one of the 9-bit modes, the Receive Interrupt (RI) flag will only get set when the received byte corresponds to either a Given or Broadcast address.

The master can selectively communicate with groups of slaves by using the Given Address. Addressing all slaves at once is possible with the Broadcast Address. These addresses are defined for each slave by two Special Function Registers: SADDR and SADEN.

A slave's individual address is specified in SADDR. SADEN is a mask byte that defines don't-cares to form the Given Address. These don't-cares allow flexibility in the user-defined protocol to address one or more slaves at a time. The following is an example of how the user could define Given Addresses to selectively address different slaves.

Slave 1:

| | | |
|---|---|---|
| SADDR = | 1111 | 0001 |
| SADEN = | 1111 | 1010 |
| GIVEN = | 1111 | 0X0X |

Slave 2:

| | | |
|---|---|---|
| SADDR = | 1111 | 0011 |
| SADEN = | 1111 | 1001 |
| GIVEN = | 1111 | 0XX1 |

The SADEN bytes are selected such that each slave can be addressed separately. Notice that bit 1 (LSB) is a don't-care for Slave 1's Given Address, but bit 1 = 1 for Slave 2. Thus, to selectively communicate with just Slave 1 the master must send an address with bit 1 = 0 (e.g. 1111 0000). Similarly, bit 2 = 0 for Slave 1, but is a don't-care for Slave 2. Now to communicate with just Slave 2 an address with bit 2 = 1 must be used (e.g. 1111 0111). Finally, for a master to communicate with both slaves at once the address must have bit 1 = 1 and bit 2 = 0.

Notice, however, that bit 3 is a don't-care for both slaves. This allows two different addresses to select both slaves (1111 0001 or 1111 0101). If a third slave was added that required its bit 3 = 0, then the latter address could be used to communicate with Slave 1 and 2 but not Slave 3.

The master can also communicate with all slaves at once with the Broadcast Address. It is formed from the logical OR of the SADDR and SADEN registers with zeros defined as don't-cares. The don't-cares also allow flexibility in defining the Broadcast Address, but in most applications a Broadcast Address will be 0FFH.

The feature works the same way in the 8-bit mode (Mode 1) as in the 9-bit modes, except that the stop bit takes the place of the 9th data bit. If SM2 is set, the RI flag is set only if the received byte matches the Given or Broadcast Address and is terminated by a valid stop bit. Setting the SM2 bit has no effect in Mode 0.

On reset, the SADDR and SADEN registers are initialized to 00H, which defines the Given and Broadcast Addresses as XXXX XXXX (all don't-cares). This assures the 8XC51GB serial port to be backwards compatibility with other MCS-51 products which do not implement Automatic Addressing.

## 8.4 Baud Rates

The baud rate in Mode 0 is fixed:

$$\text{Mode 0 Baud Rate} = \frac{\text{Oscillator Frequency}}{12}$$

The baud rate in Mode 2 depends on the value of bit SMOD1 in Special Function Register PCON. If SMOD1 = 0 (which is the value on reset), the baud rate is 1/64 the oscillator frequency. If SMOD1 = 1, the baud rate is 1/32 the oscillator frequency.

$$\text{Mode 2 Baud Rate} = 2^{\text{SMOD1}} \times \frac{\text{Oscillator Frequency}}{64}$$

The baud rates in Mode 1 and Mode 3 are determined by the Timer 1 overflow rate, or by Timer 2 overflow rate, or by both (one for transmit and the other for receive).

## 8.5 Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD1 as follows:

$$\text{Modes 1 and 3 Baud Rate} = 2^{\text{SMOD1}} \times \frac{\text{Timer 1 Overflow Rate}}{32}$$

Figure 28 shows how commonly used Baud Rates may be generated. The Timer 1 interrupt should be disabled in this application. Timer 1 can be configured for either "timer" or "counter" operation, and in any of its 3 running modes. In most applications, it is configured for "timer" operation in the auto-reload mode (high

nibble of TMOD = 0010B). In this case, the baud rate is given by the formula:

$$\text{Modes 1 and 3 Baud Rate} = 2^{\text{SMOD1}} \times \frac{\text{Oscillator Frequency}}{32 \times 12 \times [256 - (\text{TH1})]}$$

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the Timer 1 interrupt to do a 16-bit software reload.

## 8.6 Timer 2 to Generate Baud Rates

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON. Note that the baud rates for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode as shown in Figure 29.

| Baud Rate | Fosc | SMOD | Timer 1 | | |
|---|---|---|---|---|---|
| | | | C_T | Mode | Reload Value |
| Mode 0 Max: 1 MHz | 12 MHz | X | X | X | X |
| Mode 2 Max: 375K | 12 MHz | 1 | X | X | X |
| Modes 1 & 3: 62.5K | 12 MHz | 1 | 0 | 2 | FFH |
| 19.2K | 11.059 MHz | 1 | 0 | 2 | FDH |
| 9.6K | 11.059 MHz | 0 | 0 | 2 | FDH |
| 4.8K | 11.059 MHz | 0 | 0 | 2 | FAH |
| 2.4K | 11.059 MHz | 0 | 0 | 2 | F4H |
| 1.2K | 11.059 MHz | 0 | 0 | 2 | E8H |
| 137.5 | 11.986 MHz | 0 | 0 | 2 | 1DH |
| 110 | 6 MHz | 0 | 0 | 2 | 72H |
| 110 | 12 MHz | 0 | 0 | 1 | FEEBH |

**Figure 28. Timer 1 Generated Commonly Used Baud Rates**

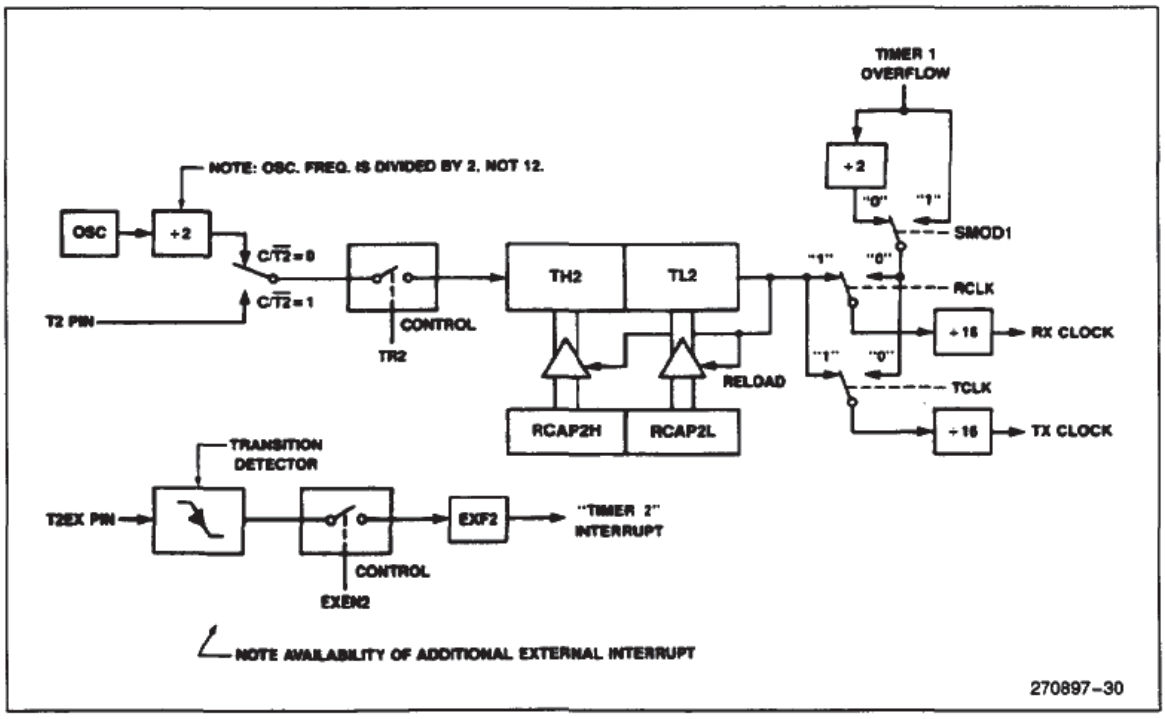



**Figure 29. Timer 2 in Baud Rate Generator Mode**

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

The baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate as follows:

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

Timer 2 can be configured for either "timer" or "counter" operation. In most applications, it is configured for "timer" operation (C_T2 = 0). The "Timer" operation is different for Timer 2 when it's being used as a baud rate generator. Normally, as a timer, it increments every machine cycle (1/12 the oscillator frequency). As a baud rate generator, however, it increments every state time (½ the oscillator frequency). The baud rate formula is given below:

$$\text{Modes 1 and 3 Baud Rate} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - (\text{RCAP2H, RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is valid only if RCLK and/or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Therefore, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Note too, that if EXEN2 is set, a 1-to-0 transition on the T2EX pin will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

Table 18 lists commonly used baud rates and how they can be obtained from Timer 2.

It should be noted that when Timer 2 is running (TR2 = 1) in "timer" function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the Timer is being incremented every state time, and the results of a read or write may not be accurate. The RCAP2 registers may be read, but shouldn't be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

**Table 18. Timer 2 Generated Baud Rates**

| Baud Rate | Fosc | Timer 2 | |
|---|---|---|---|
| | | RCAP2H | RCAP2L |
| 375K | 12 MHz | FFH | FFH |
| 9.6K | 12 MHz | FFH | D9H |
| 4.8K | 12 MHz | FFH | B2H |
| 2.4K | 12 MHz | FFH | 64H |
| 1.2K | 12 MHz | FEH | C8H |
| 300 | 12 MHz | FBH | 1EH |
| 110 | 12 MHz | F2H | AFH |
| 300 | 6 MHz | FDH | 8FH |
| 110 | 6 MHz | F9H | 57H |

## 9.0 SERIAL EXPANSION PORT

The Serial Expansion Port (SEP) allows a wide variety of serially hosted peripherals to be connected to the 8XC51GB. The SEP has four programmable modes and four clock options. There is a single bi-directional data pin (P4.1) and a clock output pin (P4.0). Data transfers consist of 8 clocks with 8 bits of data received or transmitted. When not transmitting or receiving the data and clock pins are inactive. There are 3 SFRs associated with the SEP as shown in Figure 30.

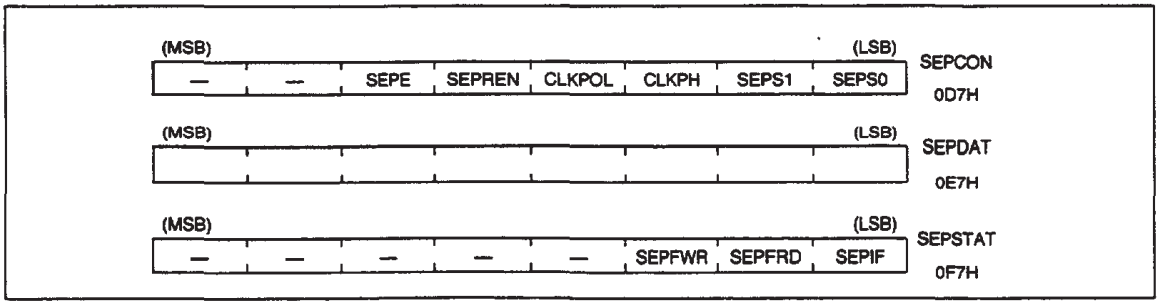



**Figure 30. SEP SFRs**

None of the SEP SFRs are bit addressable. However, the individual bits of SEPSTAT and SEPCON are significant and have symbolic names associated with them as shown. The meaning of these bits are:

SEPE — SEP Enable bit
SEPREN — SEP Receive ENable
CLKPOL — CLocK POLarity
CLKPH — CLocK PHase
SEPS1 — SEP Speed select 1
SEPS0 — SEP Speed select 0
SEPFWR — SEP Fault during WRite
SEPFRD — SEP Fault during ReaD
SEPIF — SEP Interrupt Flag

## 9.1 Programmable Modes and Clock Options

The four programmable modes determine the inactive level of the clock pin and which edge of the clock is used for transmission or reception. These four modes are shown in Figure 31. Table 19 shows how the modes are determined.

**Table 19. Determination of SEP Modes**

| CLKPOL | CLKPH | SEP Mode |
|---|---|---|
| 0 | 0 | SEPMODE0 |
| 0 | 1 | SEPMODE1* |
| 1 | 0 | SEPMODE2 |
| 1 | 1 | SEPMODE3* |

The four clock options determine the rate at which data is shifted out of or into the SEP. All four rates are fractions of the oscillator frequency. Table 20 shows the various rates that can be selected for the SEP.

**Table 20. SEP Data Rates**

| SEPS1 | SEPS0 | Data Rate |
|---|---|---|
| 0 | 0 | Fosc/12 |
| 0 | 1 | Fosc/24 |
| 1 | 0 | Fosc/48 |
| 1 | 1 | Fosc/96 |

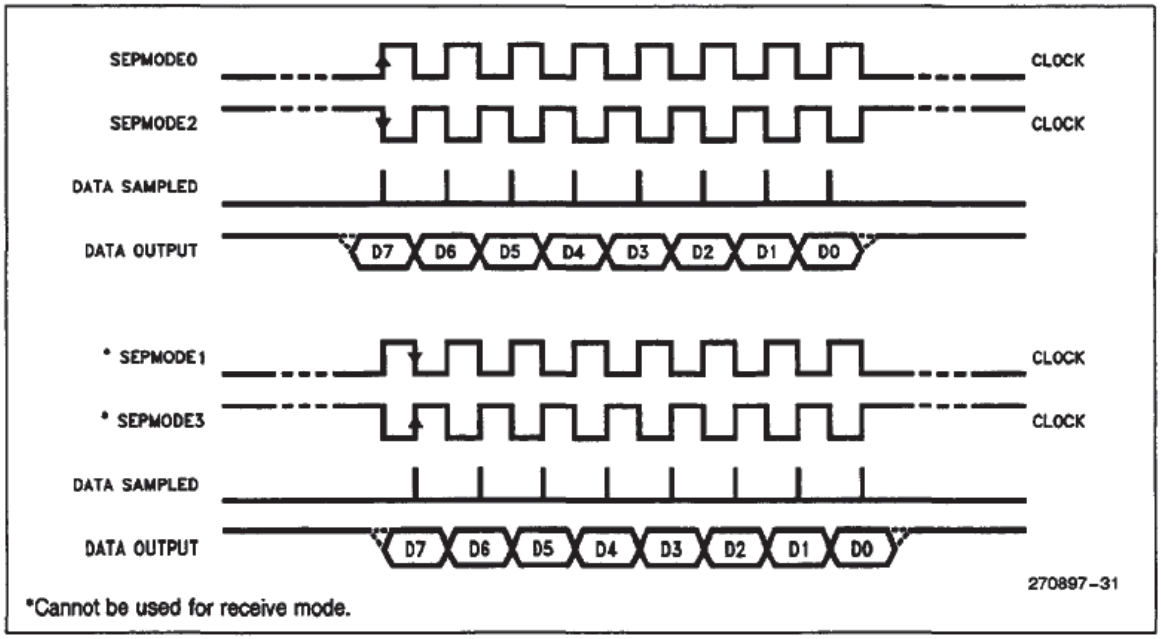



**Figure 31. SEP Modes**

## 9.2 SEP Transmission or Reception

To transmit or receive a byte the user should initialize the SEP mode (CLKPOL and CLKPH), clock frequency (SEPS1 and SEPS0), and enable the SEP (SEPE). A transmission then occurs if the user loads data into SEPDATA. A reception occurs if the user sets SEPREN while SEPDATA is empty and a transmission is not in progress. When 8 bits have been received SEPREN will be cleared by hardware. Once the transmission or reception is complete, SEPIF will be set. SEPIF remains set until cleared by software. SEPIF is also the source of the SEP interrupt. Data is transmitted and received MSB first.

If the user attempts to read or write the SEPDATA register or write to the SEPCON register while the SEP is transmitting or receiving an error bit is set. The SEPFWR bit is set if the action occurred while the SEP was transmitting. The SEPFRD bit is set if the action occurred while the SEP was receiving. There is no interrupt associated with these error bits. The bit remains set until cleared by software. The attempted read or write of the register is ignored. The reception of transmission that was in progress will not be affected.

# 10.0 HARDWARE WATCHDOG TIMER

The hardware WatchDog Timer (WDT) resets the 8XC51GB when it overflows. The WDT is intended as a recovery method in situations where the CPU may be subjected to a software upset. The WDT consists of a 14-bit counter and the WatchDog Timer ReSeT (WDTRST) SFR. The WDT is *always* enabled and increments while the oscillator is running. There is no way to disable the WDT. This means that the user must still service the WDT while testing or debugging an application. The WDT is loaded with 0 when the 8XC51GB exits reset. The WDT described in this section is not the Watchdog Timer associated with PCA module 4. The WDT does not drive the Reset pin.

## 10.1 Using the WDT

Since the WDT is automatically enabled while the processor is running, the user only needs to be concerned with servicing it. The 14-bit counter overflows when it reaches 16383 (3FFFH). The WDT increments once every machine cycle. This means the user must reset the WDT at least every 16383 machine cycles. If the user does not wish to use the functionality of the WDT in an application, a timer interrupt can be used to reset the WDT. To reset the WDT the user must write 01EH and 0E1H to WDTRST. WDTRST is a write only register. The WDT count cannot be read or written. Using a timer interrupt is not recommended in applications that make use of the WDT because interrupts may still be serviced, even after a software upset. To make the best use of the WDT, it should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset.

## 10.2 WDT During Power Down and Idle

In Power Down mode the oscillator stops, which means the WDT also stops. While in Power Down the user does not need to service the WDT. There are two methods of exiting Power Down: by a reset or via a level activated external interrupt which is enabled prior to entering Power Down. If Power Down is exited with reset, servicing of the WDT should occur as it normally does whenever the 8XC51GB is reset. Exiting Power Down with an interrupt is significantly different. The interrupt is held low which brings the device out of Power Down and starts the oscillator. The user must hold the interrupt low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service routine for the interrupt used to exit Power Down.

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

In Idle mode, the oscillator continues to run. To prevent the WDT from resetting the 8XC51GB while in Idle, the user should always set up a timer that will periodically exit Idle, service the WDT, and re-enter Idle mode.

# 11.0 OSCILLATOR FAIL DETECT

The Oscillator Fail Detect (OFD) circuitry keeps the 8XC51GB in reset when the oscillator speed is below the OFD trigger frequency. The OFD trigger frequency is shown in the data sheet as a minimum and maximum. If the oscillator frequency is below the minimum, the device is held in reset. If the oscillator frequency is greater than the maximum, the device will not be held in reset. If the frequency is between the minimum and maximum, it is indeterminate whether the device will be held in reset or not.

The OFD is automatically enabled when the device comes out of reset or when Power Down is exited with a reset or an interrupt.

The OFD is intended to function only in situations where there is a gross failure of the oscillator, such as a

broken crystal. To fulfill this need the OFD trigger frequency is significantly below the normal operating frequency. The OFD will not reset the 8XC51GB if the oscillator frequency should change to another point within the operating range.

## 11.1 OFD During Power Down

In Power Down, the 8XC51GB oscillator stops in order to conserve power. To prevent the 8XC51GB from immediately resetting itself out of power down the OFD must be disabled prior to setting the PD bit. Writing the sequence "0E1H, 01EH" to the OSCillatoR (OSCR) SFR, turns the OFD off. Once disabled, the OFD can only be re-enabled by a reset or exit from Power Down with an interrupt. The status of the OFD (whether on or off) can be determined by reading OSCR. The LSB indicates the status of the OFD. The upper 7 bits of OSCR will always be 1s when read. If OSCR = 0FFH, the OFD is enabled. If OSCR = 0FEH, the OFD is disabled.

## 12.0 INTERRUPTS

The 8XC51GB has a total of 15 interrupt vectors: seven external interrupts ($\overline{INT0}$, $\overline{INT1}$, INT2, INT3, INT4, INT5, and INT6), three timer interrupts (Timers 0, 1, and 2), two PCA interrupts (PCA0 and PCA1), the A/D interrupt, the SEP interrupt, and the serial port interrupt. Figure 32 shows the interrupt sources.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

## 12.1 External Interrupts

External Interrupts $\overline{INT0}$ and $\overline{INT1}$ can each be either level-activated or negative edge-triggered, depending on bits IT0 and IT1 in register TCON. If ITx = 0, external interrupt x is triggered by a detected low at the $\overline{INTx}$ pin. If ITx = 1, external interrupt x is negative edge-triggered.

INT2 and INT3 can each be either negative or positive edge-triggered, depending on bits IT2 and IT3 in register EXICON. If ITx = 0, external interrupt x is negative edge-triggered. If ITx = 1, external interrupt x is positive edge-triggered.

INT4, INT5, and INT6 are positive edge-triggered only.

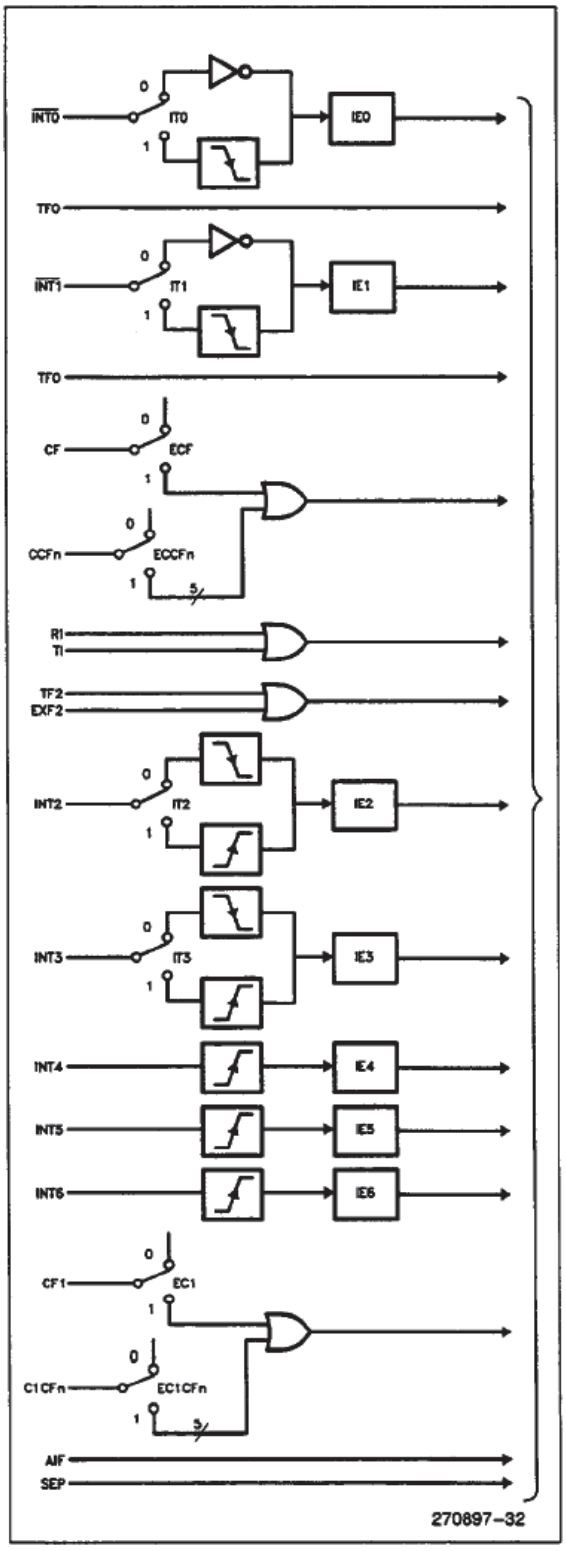



**Figure 32. Interrupt Sources**

**Table 21. EXICON: External Interrupt Control Register**

| EXICON | | | | | | Reset Value = X000 0000B | | |
|---|---|---|---|---|---|---|---|---|
| | Address = 0C6H | | | | | Not Bit Addressable | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EXICON | — | IE6 | IE5 | IE4 | IE3 | IE2 | IT3 | IT2 |

| Symbol | Function |
|---|---|
| — | Not implemented, reserved for future use.* |
| IE6 | Interrupt 6 Edge flag. This bit is set by hardware when an external interrupt edge is detected. |
| IE5 | Interrupt 5 Edge flag. This bit is set by hardware when an external interrupt edge is detected. |
| IE4 | Interrupt 4 Edge flag. This bit is set by hardware when an external interrupt edge is detected. |
| IE3 | Interrupt 3 Edge flag. This bit is set by hardware when an external interrupt edge is detected. |
| IE2 | Interrupt 2 Edge flag. This bit is set by hardware when an external interrupt edge is detected. |
| IT3 | Interrupt 3 Type control bit. This bit is set or cleared by software to control whether INT3 is positive or negative transition activated. When IT3 is high, IE3 is set by a positive transition on pin INT3. When IT3 is low, IE3 is set by a negative transition on pin INT3. |
| IT2 | Interrupt 2 Type control bit. This bit is set or cleared by software to control whether INT2 is positive or negative transition activated. When IT2 is high, IE2 is set by a positive transition on pin INT2. When IT2 is low, IE2 is set by a negative transition on pin INT2. |

*Using software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from reserved bit is indeterminate.

The flags that actually generate the interrupts are bits IE0 and IE1 in TCON and IE2, IE3, IE4, IE5, and IE6 in EXICON. These flags are cleared by hardware when the service routine is vectored to if the interrupt was transition-activated. If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware. The external interrupts are enabled through bits EX0 and EX1 in the IE register and EX2, EX3, EX4, EX5, and EX6 in the IEA register.

Since the external interrupt pins are sampled once each machine cycle, an input high or low should hold for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one cycle, and then hold it low for at least one cycle to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.

If external interrupt $\overline{\text{INT0}}$ or $\overline{\text{INT1}}$ is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

## 12.2 Timer Interrupts

Timer 0 and Timer 1 interrupts are generated by TF0 and TF1 in register TCON, which are set by a rollover in their respective Timer/Counter registers; the exception is Timer 0 in Mode 3. When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to. These timer interrupts are enabled by bits ET0 and ET1 in the IE register.

Timer 2 interrupt is generated by the logical OR of bits TF2 and EXF2 in register T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared in software. The Timer 2 interrupt is enabled by the ET2 bit in the IE register.

## 12.3 PCA Interrupt

The PCA interrupts are generated by the logical OR of five event flags (CCFn, C1CFn) and the PCA timer overflow flag (CF, CF1) in the registers CCON and C1CON. None of these flags are cleared by hardware when the service routine is vectored to. Normally the service routine will have to determine which bit flagged the interrupt and clear that bit in software. This allows the user to define the priority of servicing each PCA module.

The PCA interrupt is enabled by bit EC in the IE register. The PCA1 interrupt is enabled by bit EC1 in the IEA register. In addition, the CF (CF1) flag and each of the CCFn (C1CFn) flags must also be individually enabled by bits ECF (ECF1) and ECCFn (EC1CFn) in registers CMOD (C1MOD) and CCAPMn (C1CAPMn), respectively, in order for that flag to be able to cause an interrupt.

## 12.4 Serial Port Interrupt

The serial port interrupt is generated by the logical OR of bits RI and TI in register SCON. Neither of these flags is cleared by hardware when the service routine is vectored to. The service routine will normally have to determine whether it was RI or TI that generated the interrupt, and the bit will have to be cleared in software. The serial port interrupt is enabled by bit ES in the IE register.

## 12.5 Interrupt Enable

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in the *Interrupt Enable* (IE and IEA) registers as shown in Table 22. Note that IE also contains a global disable bit, EA. If EA is set (1), the interrupts are individually enabled or disabled by their corresponding bits in IE and IEA. If EA is clear (0), all interrupts are disabled. Figure 33 shows the interrupt control system.

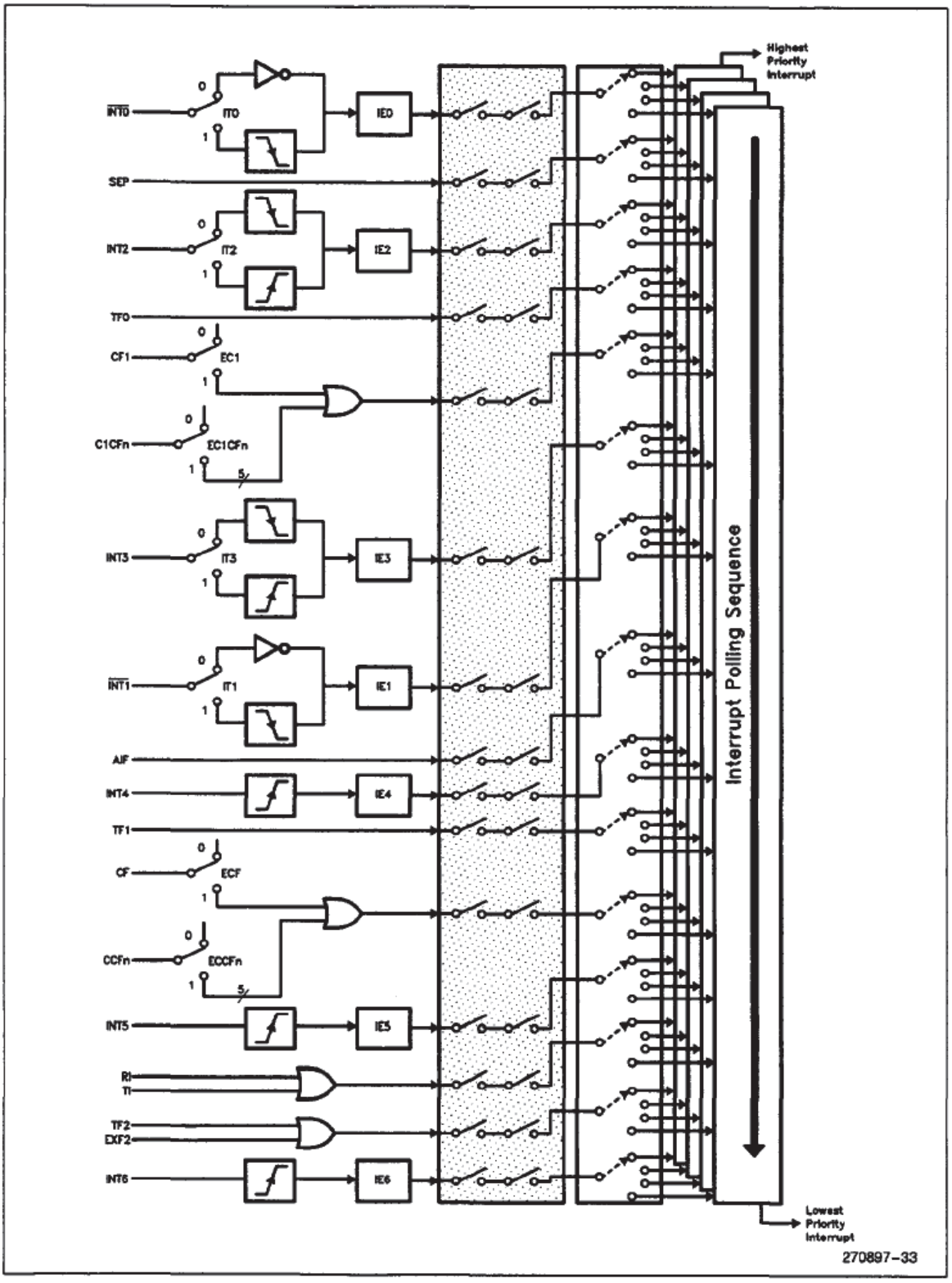



**Figure 33. Interrupt Control System**

**Table 22. Interrupt Enable Registers**

**IE** Address = 0A8H — Reset Value = 0000 0000B

Bit Addressable

| | EA | EC | ET2 | ES | ET1 | EX1 | ET0 | EX0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**IEA** Address = 0A7H — Reset Value = 0000 0000B

Not Bit Addressable

| | EAD | EX6 | EX5 | EX4 | EX3 | EX2 | EC1 | ESEP |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Enable bit = 1 enables the interrupt
Enable bit = 0 disables the interrupt

| Symbol | Function |
|---|---|
| EA | Global disable bit. If EA=0, all Interrupts are disabled. If EA=1, each Interrupt can be individually enabled or disabled by setting or clearing its enable bit. |
| EC | PCA interrupt enable bit. |
| ET2 | Timer 2 interrupt enable bit |
| ES | Serial Port interrupt enable bit. |
| ET1 | Timer 1 interrupt enable bit. |
| EX1 | External interrupt 1 enable bit. |
| ET0 | Timer 0 interrupt enable bit. |
| EX0 | External interrupt 0 enable bit. |
| EAD | A/D converter interrupt enable bit. |
| EX6 | External interrupt 6 enable bit. |
| EX5 | External interrupt 5 enable bit. |
| EX4 | External interrupt 4 enable bit. |
| EX3 | External interrupt 3 enable bit. |
| EX2 | External interrupt 2 enable bit. |
| EC1 | PCA1 interrupt enable bit. |
| ESEP | Serial Expansion Port interrupt enable bit. |

## 12.6 Interrupt Priorities

Each interrupt source on the 8XC51GB can be individually programmed to one of four priority levels, by setting or clearing the bits in the Interrupt Priority (IP and IPA) registers and the Interrupt Priority High (IPH and IPAH) registers. See Table 23. The IPH registers have the same bit map as the IP registers with an "H" added to each bit's name. This gives each interrupt source two bits for setting the priority levels. The LSB of the Priority Select Bits is in the IP SFR, and the MSB is in the IPH SFR.

A low-priority interrupt can itself be interrupted by a higher priority interrupt, but not by another interrupt of the same priority. The highest priority interrupt cannot be interrupted by any other interrupt source.

If two or more requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence shown in Table 24.

**Table 23. Interrupt Priority Registers**

**IP** Address = 0B8H — Reset Value = X000 0000B

*Bit Addressable*

| — | PPC | PT2 | PS | PT1 | PX1 | PT0 | PX0 |
|---|---|---|---|---|---|---|---|

**IPA** Address = 0B6H — Reset Value = 0000 0000B

Not Bit Addressable

| PAD | PX6 | PX5 | PX4 | PX3 | PX2 | PC1 | PSEP |
|---|---|---|---|---|---|---|---|

**IPH** Address = 0B7H — Reset Value = X000 0000B

Not Bit Addressable

| — | PPPC | PT2H | PSH | PT1H | PX1H | PT0H | PX0H |
|---|---|---|---|---|---|---|---|

**IPHA** Address = 0B5H — Reset Value = 0000 0000B

Not Bit Addressable

| PADH | PX6H | PX5H | PX4H | PX3H | PX2H | PC1H | PSEPH |
|---|---|---|---|---|---|---|---|

| Priority Bit | Priority Bit H | Priority |
|---|---|---|
| 0 | 0 | Lowest |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | Highest |

| Symbol | Function |
|---|---|
| — | Not Implemented, reserved for future use* |
| PPC, PPCH | PCA interrupt priority bits |
| PT2, PT2H | Timer 2 interrupt priority bits |
| PS, PSH | Serial Port interrupt priority bits |
| PT1, PT1H | Timer 1 interrupt priority bits |
| PX1, PX1H | External interrupt 1 interrupt priority bits |
| PT0, PT0H | Timer 0 interrupt priority bits |
| PX0, PX0H | External interrupt 0 interrupt priority bits |
| PAD, PADH | A/D converter interrupt priority bits |
| PX6, PX6H | External interrupt 6 interrupt priority bits |
| PX5, PX5H | External interrupt 5 interrupt priority bits |
| PX4, PX4H | External interrupt 4 interrupt priority bits |
| PX3, PX3H | External interrupt 3 interrupt priority bits |
| PX2, PX2H | External interrupt 2 interrupt priority bits |
| PC1, PC1H | PCA1 interrupt priority bits |
| PSEP, PSEPH | Serial Expansion Port interrupt priority bits |

**NOTE:**
*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

**Table 24. Interrupt Polling Sequence**

| | |
|---|---|
| 1 (Highest) | $\overline{\text{INT0}}$ |
| 2 | SEP |
| 3 | INT2 |
| 4 | Timer 0 |
| 5 | PCA1 |
| 6 | INT3 |
| 7 | $\overline{\text{INT1}}$ |
| 8 | A/D |
| 9 | INT4 |
| 10 | Timer 1 |
| 11 | PCA |
| 12 | INT5 |
| 13 | PCA |
| 14 | Timer 2 |
| 15 (Lowest) | INT6 |

Note that the "priority within level" structure is only used to resolve simultaneous requests of the same priority level.

## 12.7 Interrupt Processing

The interrupt flags are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. The Timer 2 overflow interrupt is slightly different, as described in the Interrupt Response Time section. If one of the flags was in a set condition at S5P2 of the preceding 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 equal or higher priority level is already in progress.
2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE or IP registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any write to IE or IP, then at least one more instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at S5P2 of the previous machine cycle. If the interrupt flag for a level-sensitive external interrupt is active but not being responded to for one of the above conditions and is not still active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The polling cycle/LCALL sequence is illustrated in the Interrupt Response Timing Diagram.

Note that if an interrupt of a higher priority level goes active prior to S5P2 of the machine cycle labeled C3 in the diagram, then in accordance with the above rules it will be vectored to during C5 and C6, without any instruction of the lower priority routine having been executed. This is the fastest possible response when C2 is the final cycle of an instruction other than RETI or write IE or IP.

Thus the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine. The hardware-generated LCALL pushes the contents of the Program Counter onto 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. Table 25 shows the interrupt vector addresses.

**Table 25. Interrupt Vector Addresses**

| Interrupt Source | Interrupt Request Bits | Cleared by Hardware | Vector Address |
|---|---|---|---|
| $\overline{\text{INT0}}$ | IE0 | No (level)<br>Yes (trans.) | 0003H |
| Timer 0 | TF0 | Yes | 000BH |
| $\overline{\text{INT1}}$ | IE1 | No (level)<br>Yes (trans.) | 0013H |
| Timer 1 | TF1 | Yes | 001BH |
| Serial Port | RI, TI | No | 0023H |
| Timer 2 | TF2, EXF2 | No | 002BH |
| PCA | CF, CCFn<br>(n = 0–4) | No | 0033H |
| A/D | AIF | No | 003BH |
| PCA1 | CF1, C1CCFn<br>(n = 0–4) | No | 0043H |
| SEP | SEPIF | No | 004BH |
| INT2 | IE2 | Yes | 0053H |
| INT3 | IE3 | Yes | 005BH |
| INT4 | IE4 | Yes | 0063H |
| INT5 | IE5 | Yes | 006BH |
| INT6 | IE6 | Yes | 0073H |

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stack and reloads the Program Counter. Execution of the interrupted program continues from where it left off.

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking interrupt was still in progress.

The starting addresses of consecutive interrupt service routines are only 8 bytes apart. That means if consecutive interrupts are being used (IE0 and TF0, for example, or TF0 and IE1), and if the first interrupt routine is more than 7 bytes long, then that routine will have to execute a jump to some other memory location where the service routine can be completed without overlapping the starting address of the next interrupt routine.

## 12.8 Interrupt Response Time

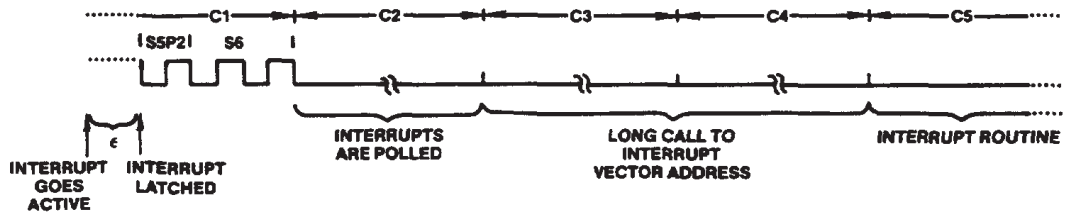
The $\overline{INT0}$ and $\overline{INT1}$ levels are inverted and latched into the Interrupt Flags IE0, and IE1 at S5P2 of every machine cycle. The level of interrupts 2 through 6 are also latched into the appropriate flags (IE2–IE6) in S5P2. Similarly, the Timer 2 flag EXF2 and the Serial Port flags RI and TI are set at S5P2. The values are not actually polled by the circuitry until the next machine cycle.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag TF2 is set at S2P2 and is polled in the same cycle in which the timer overflows.

If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapses between activation of an external interrupt request and the beginning of execution of the service routine's first instruction. See Figure 34.

A longer response time would result if the request is blocked by one of the 3 conditions discussed in the Interrupt Processing section. If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is not in its final cycle, the additional wait time cannot be more than 3 cycles, since the longest instructions (MUL and DIV) are only 4 cycles long, and if the instruction in progress is RETI or write to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one or more cycles to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV).

Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 9 cycles.




*This is the fastest possible response when C2 is the final cycle of an instruction other than RETI or write IE or IP.*

**Figure 34. Interrupt Response Timing Diagram**

## 13.0 RESET

The reset input is the $\overline{\text{RESET}}$ pin, which has a Schmitt Trigger input. A reset is accomplished by holding the $\overline{\text{RESET}}$ pin low for at least two machine cycles (24 oscillator periods). On the 8XC51GB, reset is asynchronous to the CPU clock. This means that the oscillator does not have to be running for the I/O pins to be in their reset condition. However, $V_{CC}$ has to be within the specified operating conditions.

Once Reset has reached a high level, the 8XC51GB may remain in its reset state for up to 5 machine cycles. This is caused by the OFD circuitry.

While the $\overline{\text{RESET}}$ pin is low, the port pins, ALE and $\overline{\text{PSEN}}$ are weakly pulled high. After $\overline{\text{RESET}}$ is pulled high, it will take up to 5 machine cycles for ALE and $\overline{\text{PSEN}}$ to start clocking. For this reason, other devices can not be synchronized to the internal timings of the 8XC51GB.

Driving the ALE and $\overline{\text{PSEN}}$ pins to 0 while reset is active could cause the device to go into an indeterminate state.

The internal reset algorithm redefines most of the SFRs. Refer to individual SFRs for their reset values. The internal RAM is not affected by reset. On power up the RAM content is indeterminate.

### 13.1 Power-On Reset

For CHMOS devices, when $V_{CC}$ is turned on, an automatic reset can be obtained by connecting the $\overline{\text{RESET}}$ pin to $V_{SS}$ through a 1 $\mu$F capacitor. The CHMOS devices do not require an external resistor like the HMOS devices because they have an internal pullup on the $\overline{\text{RESET}}$ pin. Figure 35 shows this.

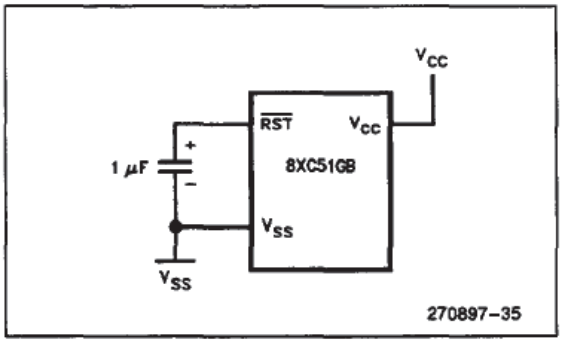



**Figure 35. Power-On Reset Circuitry**

When power is turned on, the circuit holds the $\overline{\text{RESET}}$ pin high for an amount of time that depends on the capacitor value and the rate at which it charges. To ensure a valid reset the $\overline{\text{RESET}}$ pin must be held low long enough to allow the oscillator to start up plus two machine cycles.

On power up, $V_{CC}$ should rise within approximately ten milliseconds. The oscillator start-up time will depend on the oscillator frequency. For a 10 MHz crystal, the start-up time is typically 1 ms. For a 1 MHz crystal, the start-up time is typically 10 ms.

Powering up the device without a valid reset could cause the CPU to start executing instructions from an indeterminate location. This is because the SFRs, specifically the Program Counter, may not get properly initialized.

## 14.0 POWER-SAVING MODES

For applications where power consumption is critical, the 8XC51GB provides two power reducing modes of operation: Idle and Power Down. The input through which backup power is supplied during these operations is $V_{CC}$. The Idle and Power Down modes are activated by setting bits IDL and PD, respectively, in the SFR PCON (Table 26). Figure 36 shows the Idle and Power Down circuitry.

In the Idle mode (IDL = 1), the oscillator continues to run and the Interrupt, Serial Port, PCA, and Timer blocks continue to be clocked, but the clock signal is gated off to the CPU. In Power Down (PD = 1), the oscillator is frozen.

**Table 26. PCON: Power Control Register**

**PCON** Address = 87H Reset Value = 00XX 0000B

Not Bit Addressable

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

| Symbol | Function |
|---|---|
| SMOD1 | Double Baud rate bit. When set to a 1 and Timer 1 is used to generate baud rates, and the Serial Port is used in modes 1, 2, or 3. |
| SMOD0 | When set, Read/Write accesses to SCON.7 are to the FE bit. When clear, Read/Write accesses to SCON.7 are to the SM0 bit. |
| — | Not implemented, reserved for future use.* |
| POF | Power Off Flag. Set by hardware on the rising edge of $V_{CC}$. Set or cleared by software. This flag allows detection of a power failure caused reset. $V_{CC}$ must remain above 3V to retain this bit. |
| GF1 | General-purpose flag bit. |
| GF0 | General-purpose flag bit. |
| PD | Power Down bit. Setting this bit activates Power Down operation. |
| IDL | Idle mode bit. Setting this bit activates idle modes operation. If 1s are written to PD and IDL at the same time, PD takes precedence. |

**NOTE:**
*User software should not write 1s to unimplemented bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

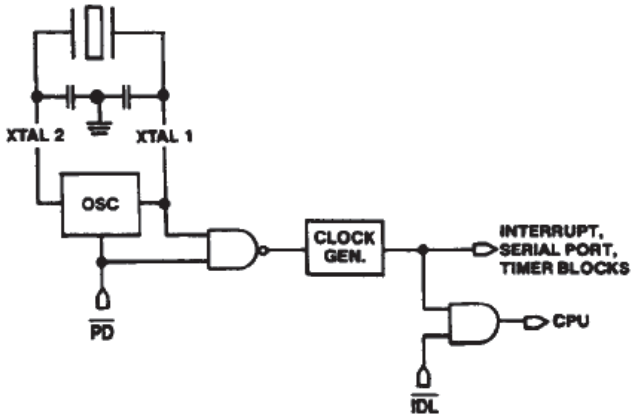



**Figure 36. Idle and Power Down Hardware**

## 14.1 Idle Mode

An instruction that sets the IDL bit causes that to be the last instruction executed before going into the Idle mode. In the Idle mode, the internal clock signal is gated off to the CPU, but not to the Interrupt, Timer, and Serial Port functions. The PCA and PCA1 timers can be programmed either to pause or continue operating during Idle with the CIDL (C1IDL) bit in CMOD (C1MOD). The CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical states they had at the time Idle was activated. ALE and $\overline{PSEN}$ hold at logic high levels. Refer to Table 27.

**Table 27. Status of the External Pins during Idle Mode**

| Program Memory | ALE | $\overline{PSEN}$ | Port 0 | Port 1 | Port 2 | Ports 3, 4, 5 |
|---|---|---|---|---|---|---|
| Internal | 1 | 1 | Data | Data | Data | Data |
| External | 1 | 1 | Float | Data | Address | Data |

There are two ways to terminate the Idle Mode. Activation of any enabled interrupt will cause the IDL bit to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that put the device into Idle.

The flag bits (GF0 and GF1 in PCON) can be used to give an indication if an interrupt occurred during normal operation or during Idle. For example, an instruction that activates Idle can also set one or both flag bits. When Idle is terminated by an interrupt, the interrupt service routine can examine the flag bits.

The other way of terminating the Idle mode is with a hardware reset. Since the clock oscillator is still running, the hardware reset needs to be held active for only two machine cycles (24 oscillator periods) to complete the reset.

The signal at the $\overline{RESET}$ pin clears the IDL bit directly and asynchronously. At this time the CPU resumes program execution from where it left off; that is, at the instruction following the one that invoked the Idle Mode. As shown in the Reset Timing diagram, two or three machine cycles of program execution may take place before the internal reset algorithm takes control. On-chip hardware inhibits access to the internal RAM during this time, but access to the port pins is not inhibited. To eliminate the possibility of unexpected outputs at the port pins, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external Data RAM.

## 14.2 Power Down Mode

An instruction that sets the PD bit causes that to be the last instruction executed before going into the Power Down mode. In this mode the on-chip oscillator is stopped. With the clock frozen, all functions are stopped, but the on-chip RAM and Special Function Registers are held. The port pins output the values held by their respective SFRs, and ALE and $\overline{PSEN}$ output lows. In Power Down, $V_{CC}$ can be reduced to as low as 2V. Care must be taken, however, to ensure that $V_{CC}$ is not reduced before Power Down is invoked. If the Oscillator Fail Detect circuitry is not disabled before entering powerdown, the part will reset itself (see Section 11.0 "Oscillator Fail Detect"). Table 28 shows the status of external pins during Power Down mode.

**Table 28. Status of the External Pins during Power Down Mode**

| Program Memory | ALE | $\overline{PSEN}$ | Port 0 | Port 1 | Port 2 | Ports 3, 4, 5 |
|---|---|---|---|---|---|---|
| Internal | 0 | 0 | Data | Data | Data | Data |
| External | 0 | 0 | Float | Data | Data | Data |

The 8XC51GB can exit Power Down with either a hardware reset or external interrupt. Reset redefines most of the SFRs but does not change the on-chip RAM. An external interrupt allows both the SFRs and the on-chip RAM to retain their values.

To properly terminate Power Down the reset or external interrupt should not be executed before $V_{CC}$ is restored to its normal operating level and must be held active long enough for the oscillator to restart and stabilize (normally less than 10 ms).

With an external interrupt, $\overline{INT0}$ or $\overline{INT1}$ must be enabled and configured as level-sensitive. Holding the pin low restarts the oscillator and bringing the pin back high completes the exit. After the RETI instruction is executed in the interrupt service routine, the next instruction will be the one following the instruction that put the device in Power Down.

## 14.3 Power Off Flag

The Power Off Flag (POF) located at PCON.4 is set by hardware when $V_{CC}$ rises from 0V to 5V. POF can also be set or cleared by software. This allows the user to distinguish between a "cold start" reset and a "warm start" reset.

A cold start reset is one that is coincident with $V_{CC}$ being turned on to the device after it was turned off. A warm start reset occurs while $V_{CC}$ is still applied to the device and could be generated, for example, by a Watchdog Timer or an exit from Power Down.

Immediately after reset, the user's software can check the status of the POF bit. POF = 1 would indicate a cold start. The software then clears POF and commences its tasks. POF = 0 immediately after reset would indicate a warm start.

$V_{CC}$ must remain above 3V for POF to retain a 0.

## 15.0 EPROM/OTP PROGRAMMING

The 8XC51GB uses the fast "Quick-Pulse" Programming algorithm. The devices program at $V_{PP}$ = 12.75V (and $V_{CC}$ = 5.0V) using a series of five 100 μs PROG pulses per byte programmed.

### 15.1 Program Memory Lock

In some microcontroller applications it is desirable that the Program Memory be secure from software piracy. The 8XC51GB has a three-level program lock feature which protects the code of the on-chip EPROM/OTP or ROM.

Within the EPROM/OTP/ROM are 64 bytes of Encryption Array that are initially unprogrammed (all 1s). The user can program the Encryption Array to encrypt the program code bytes during EPROM/OTP/ROM verification. The verification procedure is performed as usual except that each code byte comes out exclusive-NOR'ed (XNOR) with one of the key bytes. Therefore, to read the ROM code the user has to know the 64 key bytes in their proper sequence.

Unprogrammed bytes have the value 0FFH. So if the Encryption Array is left unprogrammed, all the key bytes have the value 0FFH. Since any code byte XNORed with 0FFH leaves the byte unchanged, leaving the Encryption Array unprogrammed in effect bypasses the encryption feature.

#### PROGRAM LOCK BITS

Also included in the Program Lock scheme are three Lock Bits which can be programmed to disable certain functions as shown in Table 29.

To obtain maximum security of the on-board program and data, all 3 Lock Bits and the Encyption Array must be programmed.

Erasing the EPROM also erases the Encryption Array and the Lock Bits, returning the part to full functionality.

**Table 29. EPROM/OTP Lock Bits**

| Program Lock Bits | | | Logic Enabled |
|---|---|---|---|
| LB1 | LB2 | LB3 | |
| U | U | U | No Program Lock features enabled. (Code Verify will still be encrypted by the Encryption Array.) |
| P | U | U | MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory. $\overline{EA}$ is sampled and latched on reset, and further programming of EPROM is disabled. |
| P | P | U | Same as above, but Verify is also disabled (option available on EPROM only). |
| P | P | P | Same as above and all external program execution is inhibited and internal RAM cannot be read externally. |

**NOTE:**
All other combinations of lock bits may produce indeterminate results and should not be used.

## 16.0 ONCE MODE

The ONCE (ON-Circuit Emulation) mode facilitates testing and debugging of systems using the 8XC51GB without having to remove the device from the circuit. The ONCE mode is invoked by:

1. Pulling ALE low while the device is in reset and $\overline{PSEN}$ is high;
2. Holding ALE low as RST is deactivated.

While the device is in ONCE mode, the Port 0 pins go into a float state, and the other port pins, ALE, and $\overline{PSEN}$ are weakly pulled high. The oscillator circuit remains active. While the device is in this mode, an emulator or test CPU can be used to drive the circuit.

Normal operation is restored after a valid reset is applied.

## 17.0 ON-CHIP OSCILLATOR

The on-chip oscillator for the CHMOS devices consists of a single stage linear inverter intended for use as a

crystal-controlled, positive reactance oscillator. In this application the crystal is operating in its fundamental response mode as an inductive reactance in parallel resonance with capacitance external to the crystal. Figure 37 shows the on-chip oscillator circuitry.

The oscillator on the CHMOS devices can be turned off under software control by setting the PD bit in the PCON register (Figure 38). The feedback resistor $R_f$ shown in the figure consists of parallel n- and p-channel FETs controlled by the PD bit, such that $R_f$ is opened when PD = 1. The diodes D1 and D2, which act as clamps to $V_{CC}$ and $V_{SS}$, are parasitic to the $R_f$ FETs.

The crystal specifications and capacitance values (C1 and C2 in Figure 39) are not critical. 30 pF can be used in these positions at any frequency with good quality crystals. In general, crystals used with these devices typically have the following specifications:

| | |
|---|---|
| ESR (Equivalent Series Resistance) | |
| CO (shunt capacitance) | 7.0 pF maximum |
| CL (load capacitance) | 30 pF ±3 pF |
| Drive Level | 1 MW |

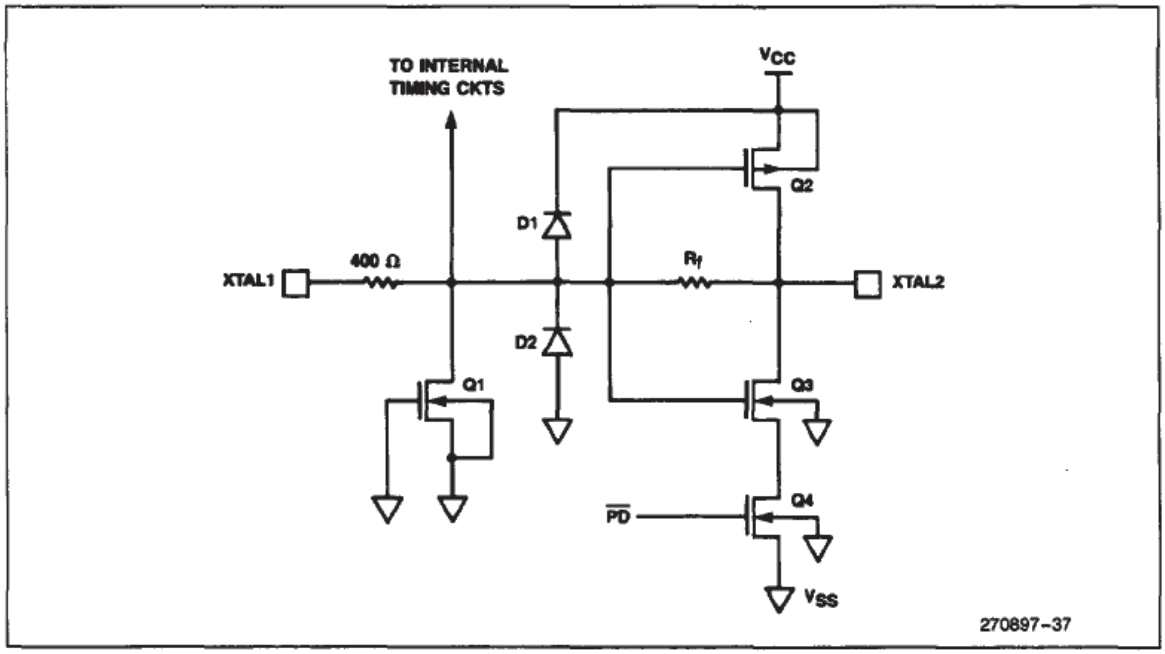



**Figure 37. On-Chip Oscillator Circuitry**

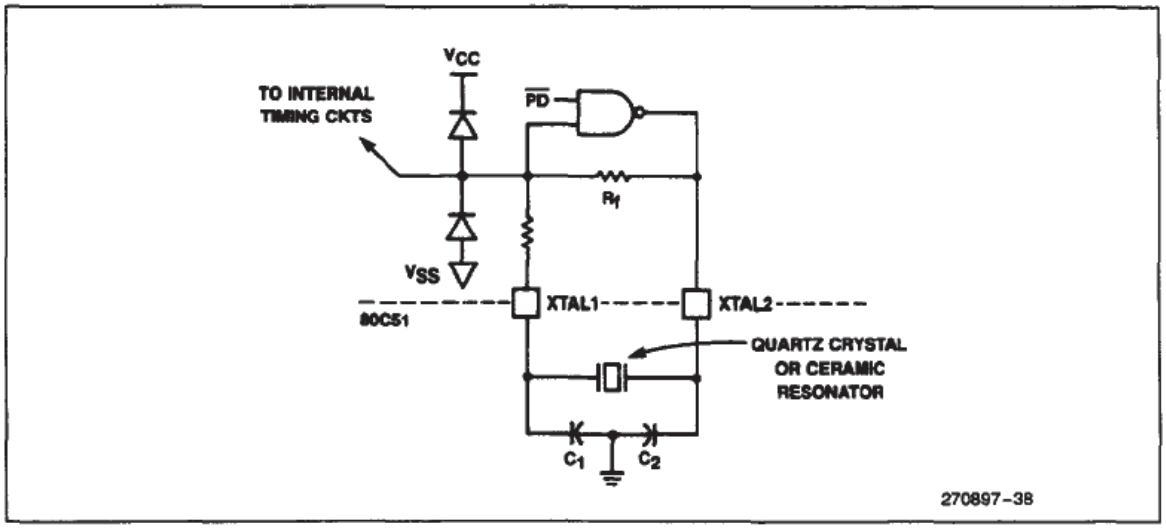



**Figure 38. Using the CHMOS On-Chip Oscillator**

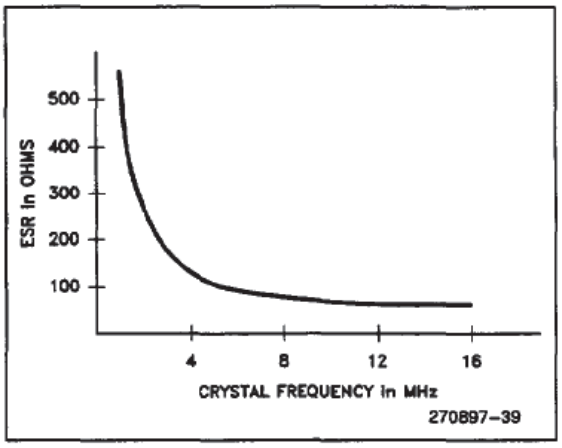



**Figure 39. ESR vs Frequency**

Frequency, tolerance, and temperature range are determined by the system requirements.

A ceramic resonator can be used in place of the crystal in cost-sensitive applications. When a ceramic resonator is used, C1 and C2 are normally selected as higher values, typically 47 pF. The manufacturer of the ceramic resonator should be consulted for recommendations on the values of these capacitors.

A more in-depth discussion of crystal specifications, ceramic resonators, and the selection of values for C1 and C2 can be found in Application Note AP-155, "Oscillators for Microcontrollers" in the Embedded Control Applications handbook.

To drive the CHMOS parts with an external clock source, apply the external clock signal to XTAL1 and leave XTAL2 floating. Refer to the External Clock Source diagram. This is an important difference from the HMOS parts. With HMOS, the external clock source is applied to XTAL2, and XTAL1 is grounded. See Figure 40.

There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheets must be observed. Refer to the External Clock Specifications for this information.

An external oscillator may encounter as much as a 100 pF load at XTAL1 when it starts up. This is due to interaction between the amplifier and its feedback capacitance. Once the external signal meets the $V_{IL}$ and $V_{IH}$ specifications, the capacitance will not exceed 20 pF.

## 18.0 CPU TIMING

The internal clock generator defines the sequence of states that make up a machine cycle. A machine cycle consists of 6 states, numbered S1 through S6. Each state time lasts for two oscillator periods. Thus a machine cycle takes 12 oscillator periods or 1 μs if the oscillator frequency is 12 MHz. Each state is then divided into a Phase 1 and Phase 2 half.

Rise and fall times are dependent on the external loading that each pin must drive. They are approximately 10 ns, measured between 0.8V and 2.0V.

Propagation delays are different for different pins. For a given pin they vary with pin loading, temperature, $V_{CC}$, and manufacturing lot. If the XTAL1 waveform is taken as the timing reference, propagation delays may vary from 25 ns to 125 ns.

The AC Timings section of the data sheets do not reference any timing to the XTAL1 waveform. Rather, they relate the critical edges of control and input signals to each other. The timings published in the data sheets include the effects of propagation delays under the specified test condition.

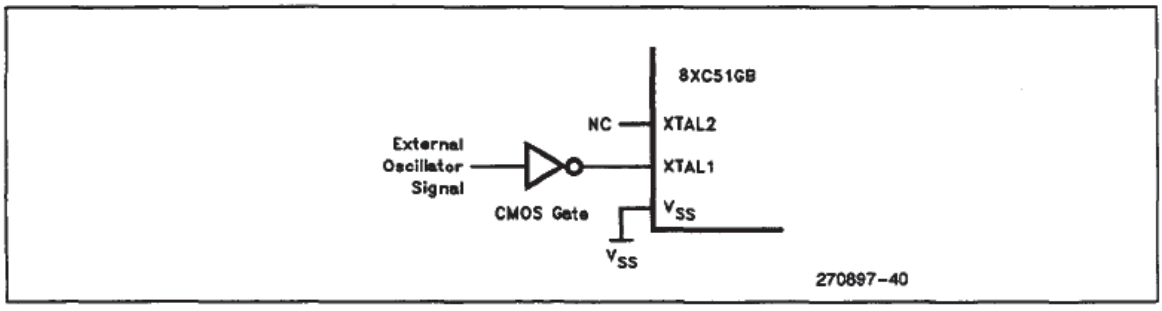



**Figure 40. Driving the CHMOS Devices with an External Clock Sources**

# 83C152 Hardware Description

7

# 83C152 HARDWARE DESCRIPTION

## CONTENTS

# 83C152 HARDWARE DESCRIPTION

## 1.0 INTRODUCTION

The 83C152 Universal Communications Controller is an 8-bit microcontroller designed for the intelligent management of peripheral systems or components. The 83C152 is a derivative of the 80C51BH and retains the same functionality. The 83C152 is fabricated on the same CHMOS III process as the 80C51BH. What makes the 83C152 different is that it has added functions and peripherals to the basic 80C51BH architecture that are supported by new Special Function Registers (SFRs). These enhancements include: a high speed multi-protocol serial communication interface, two channels for DMA transfers, HOLD/HLDA bus control, a fifth I/O port, expanded data memory, and expanded program memory.

In addition to a standard UART, referred to here as Local Serial Channel (LSC), the 83C152 has an onboard multi-protocol communication controller called the Global Serial Channel (GSC). The GSC interface supports SDLC, CSMA/CD, user definable protocols, and a subset of HDLC protocols. The GSC capabilities include: address recognition, collision resolution, CRC generation, flag generation, automatic retransmission, and a hardware based acknowledge feature. This high speed serial channel is capable of implementing the Data Link Layer and the Physical Link Layer as shown in the OSI open systems communication model. This model can be found in the document "Reference Model for Open Systems Interconnection Architecture", ISO/TC97/SC16 N309.

The DMA circuitry consists of two 8-bit DMA channels with 16-bit addressability. The control signals; Read ($\overline{RD}$), Write ($\overline{WR}$), hold and hold acknowledge (HOLD/HLDA) are used to access external memory. The DMA channels are capable of addressing up to 64K bytes (16 bits). The destination or source address can be automatically incremented. The lower 8 bits of the address are multiplexed on the data bus Port 0 and the upper eight bits of address will be on Port 2. Data is transmitted over an 8-bit address/data bus. Up to 64K bytes of data may be transmitted for each DMA activation.

The new I/O port (P4) functions the same as Ports 1-3, found on the 80C51BH.

Internal memory has been doubled in the 83C152. Data memory has been expanded to 256 bytes, and internal program memory has been expanded to 8K bytes.

There are also some specific differences between the 83C152 and the 80C51BH. The first is that the numbering system between the 83C152 and the 80C51BH is slightly different. The 83C152 and the 80C51BH are factory masked ROM devices. The 80C152 and the 80C31BH are ROMless devices which require the use of external program memory. The second difference is that RESET is active low in the 83C152 and active high in the 80C51BH. This is very important to designers who may currently be using the 80C51BH and planning to use the 83C152, or are planning on using both devices on the same board. The third difference is that GF0 and GF1, general purpose flags in PCON, have been renamed GFIEN and XRCLK. GFIEN enables idle flags to be generated in SDLC mode, and XRCLK enables the receiver to be externally clocked. All of the previously unused bits are now being used and interrupt vectors have been added to support the new enhancements. Programmers using old code generated for the 80C51BH will have to examine their programs to ensure that new bits are properly loaded, and that the new interrupt vectors will not interfere with their program.

Throughout the rest of this manual the 80C152 and the 83C152 will be referred to generically as the "C152".

The C152 is based on the 80C51BH architecture and utilizes the same 80C51BH instruction set. Figure 1.1 is a block diagram of the C152. Readers are urged to compare this block diagram with the 80C51BH block diagram. There have been no new instructions added. All the new features and peripherals are supported by an extension of the Special Function Registers (SFRs). Very little of the information pertaining specifically to the 80C51BH core will be discussed in this chapter. The detailed information on such functions as: the instruction set, port operation, timer/counters, etc., can be found in the MCS®-51 Architecture chapter in the Intel Embedded Controller Handbook. Knowledge of the 80C51BH is required to fully understand this manual and the operation of the C152. To gain a basic understanding on the operation of the 80C51BH, the reader should familiarize himself with the entire MCS-51 chapter of the Embedded Controller Handbook.

Another source of information that the reader may find helpful is Intel's LAN Components User's Manual, order number 230814. Inside are descriptions of various protocols, application examples, and application notes dealing with different serial communication environments.

## 2.0 COMPARISON OF 80C152 AND 80C51BH FEATURES

### 2.1 Memory Space

A good understanding of the memory space and how it is used in the operation of MCS-51 products is essential. All the enhancements on the C152 are implemented by accessing Special Function Registers (SFRs), added data memory, or added program memory.

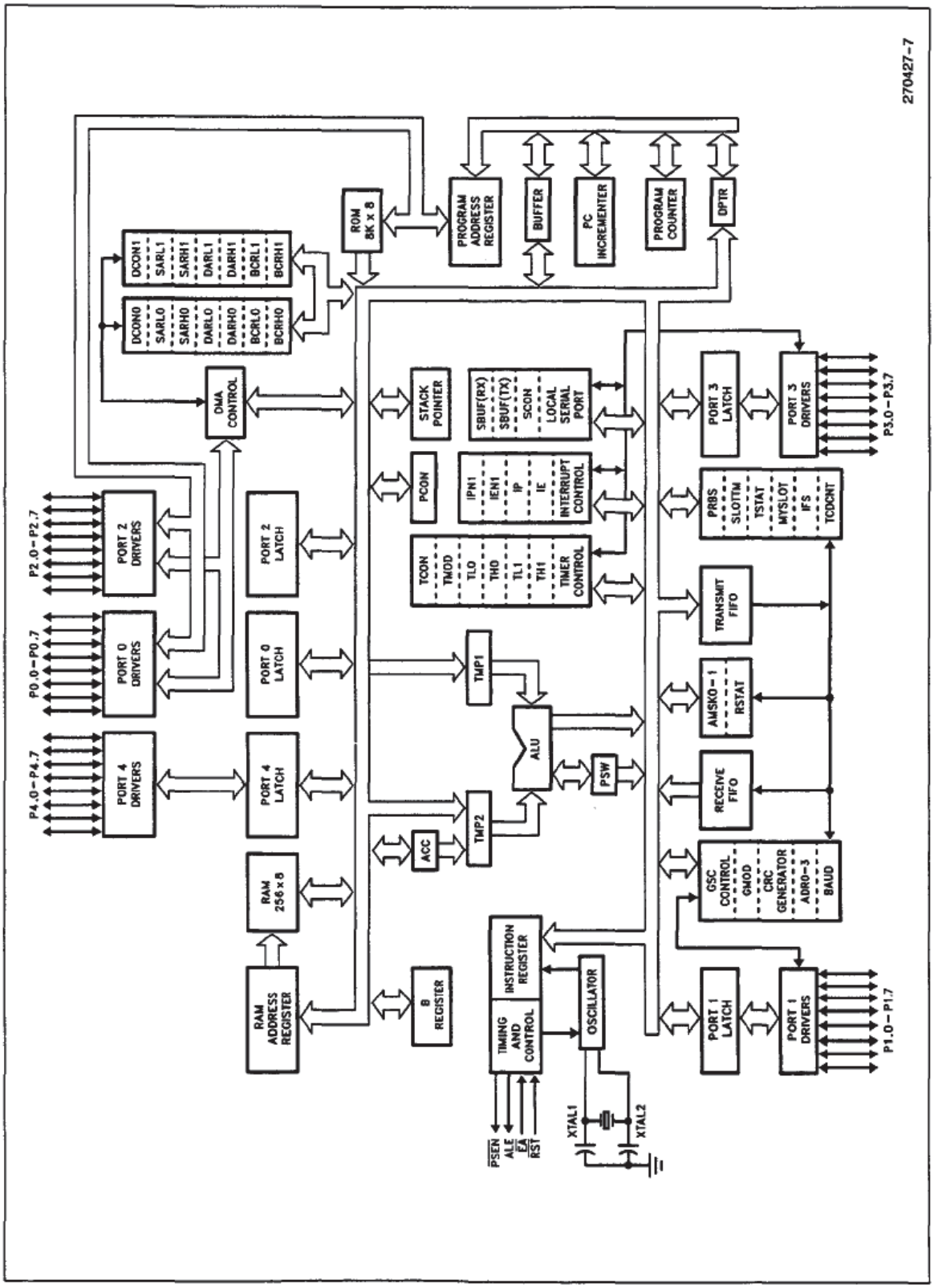



**Figure 1.1. Block Diagram**

### 2.1.1 SPECIAL FUNCTION REGISTERS (SFRs)

The following list contains all the SFRs, their names and function. All of the SFRs of the 80C51BH are retained and for a detailed explanation of their operation, please refer to the chapter, "Hardware Description of the 8051 and 8052" that is found in the Embedded Controller Handbook. An overview of the new SFRs is found in Section 2.1.1.1, with a detailed explanation in Section 3.7, Section 4.5, and 6.0.

#### 2.1.1.1 New SFRs

The following descriptions are quick overviews of the new SFRs, and not intended to give a complete understanding of their use. The reader should refer to the detailed explanation in Section 3 for the GSC SFRs, and Section 4 for the DMA SFRs.

ADR 0,1,2,3 - (95H, 0A5H, 0B5H, 0C5H) Contains the four bytes for address matching during GSC operation.

AMSK0 - (0D5H) Selects "don't care" bits to be used with ADR0.

AMSK1 - (0E5H) Selects "don't care" bits to be used with ADR1.

BAUD - (94H) Contains the programmable value for the baud rate generator for the GSC. The baud rate will equal (fosc)/((BAUD + 1) × 8).

BCRL0 - (0E2H) Contains the low byte of a count-down counter that determines when the DMA access for Channel 0 is complete.

BCRH0 - (0E3H) Contains the high byte for count-down counter for Channel 0.

BCRL1 - (0F2H) Same as BCRL0 except for DMA Channel 1.

BCRH1 - (0F3H) Same as BCRH0 except for DMA Channel 1.

BKOFF - (0C4H) An 8-bit count-down timer used with the CSMA/CD resolution algorithm.

DARL0 - (0C2H) Contains the low byte of the destination address for DMA Channel 0.

DARH0 - (0C3H) Contains the high byte of the destination address for DMA Channel 0.

DARL1 - (0D2H) Same as DARL0 except for DMA Channel 1.

DARH1 - (0D3H) Same as DARH0 except for DMA Channel 1.

DCON0 - (92H) Contains the Destination Address Space bit (DAS), Increment Destination Address bit (IDA), Source Address Space bit (SAS), Increment Source Address bit (ISA), DMA Channel Mode bit (DM), Transfer Mode bit (TM), DMA Done bit (DONE), and the GO bit (GO). DCON0 is used to control DMA Channel 0.

DCON1 - (93H) Same as DCON0 except this is for DMA Channel 1.

GMOD - (84H) Contains the Protocol bit (PR), the Preamble Length (PL1,0), CRC Type (CT), Address Length (AL), Mode select (M1,0), and External Transmit Clock (TXC). This register is used for GSC operation only.

IEN1 - (0C8H) Interrupt enable register for DMA and GSC interrupts.

IFS - (0A4H) Determines the number of bit times separating transmitted frames.

IPN1 - (0F8H) Interrupt priority register for DMA and GSC interrupts.

MYSLOT - (0F5H) Contains the Jamming mode bit (DCJ), the Deterministic Collision Resolution Algorithm bit (DCR), and the DCR slot address for the GSC.

P4 - (0C0H) Contains the memory "image" of Port 4.

PRBS - (0E4H) Contains a pseudo-random number to be used in CSMA/CD backoff algorithms. May be read or written to by user software.

RFIFO - (F4H) RFIFO is used to access a 3-byte FIFO that contains the receive data from the GSC.

RSTAT - (0E8H) Contains the Hardware Based Acknowledge Enable bit (HABEN), Global Receive Enable bit (GREN), Receive FIFO Not Empty bit (RFNE), Receive Done bit (RDN), CRC Error bit (CRCE), Alignment Error bit (AE), Receiver Collision/Abort detect bit (RCABT), and the Overrun bit (OVR), used with both DMA and GSC.

SARL0 - (0A2H) Contains the low byte of the source address for DMA transfers.

SARH0 - (0A3H) Contains the high byte of the source address for DMA transfers.

SARL1 - (0B2H) Same as SARL0 but for DMA Channel 1.

SARH1 - (0B3H) Same as SARH1 but for DMA Channel 1.

SLOTTM - (0B4H) Determines the length of the slot time in CSMA/CD.

TCDCNT - (0D4H) Contains the number of collisions in the current frame if using CSMA/CD GSC.

| Old(O)/New(N) | Name | Addr | Function |
|---|---|---|---|
| O | A | 0E0H | ACCUMULATOR |
| N | ADR0 | 095H | GSC MATCH ADDRESS 0 |
| N | ADR1 | 0A5H | GSC MATCH ADDRESS 1 |
| N | ADR2 | 0B5H | GSC MATCH ADDRESS 2 |
| N | ADR3 | 0C5H | GSC MATCH ADDRESS 3 |
| N | AMSK0 | 0D5H | GSC ADDRESS MASK 0 |
| N | AMSK1 | 0E5H | GSC ADDRESS MASK 1 |
| O | B | 0F0H | B REGISTER |
| N | BAUD | 094H | GSC BAUD RATE |
| N | BCRL0 | 0E2H | DMA BYTE COUNT 0 (LOW) |
| N | BCRH0 | 0E3H | DMA BYTE COUNT 0 (HIGH) |
| N | BCRL1 | 0F2H | DMA BYTE COUNT 1 (LOW) |
| N | BCRH1 | 0F3H | DMA BYTE COUNT 1 (HIGH) |
| N | BKOFF | 0C4H | GSC BACKOFF TIMER |
| N | DARL0 | 0C2H | DMA DESTINATION ADDR 0 (LOW) |
| N | DARH0 | 0C3H | DMA DESTINATION ADDR 0 (HIGH) |
| N | DARL1 | 0D2H | DMA DESTINATION ADDR 1 (LOW) |
| N | DARH1 | 0D3H | DMA DESTINATION ADDR 1 (HIGH) |
| N | DCON0 | 092H | DMA CONTROL 0 |
| N | DCON1 | 093H | DMA CONTROL 1 |
| O | DPH | 083H | DATA POINTER (HIGH) |
| O | DPL | 082H | DATA POINTER (LOW) |
| N | GMOD | 084H | GSC MODE |
| O | IE | 0A8H | INTERRUPT ENABLE REGISTER 0 |
| N | IEN1 | 0C8H | INTERRUPT ENABLE REGISTER 1 |
| N | IFS | 0A4H | GSC INTERFRAME SPACING |
| O | IP | 0B8H | INTERRUPT PRIORITY REGISTER 0 |
| N | IPN1 | 0F8H | INTERRUPT PRIORITY REGISTER 1 |
| N | MYSLOT | 0F5H | GSC SLOT ADDRESS |
| O | P0 | 080H | PORT 0 |
| O | P1 | 090H | PORT 1 |
| O | P2 | 0A0H | PORT 2 |
| O | P3 | 0B0H | PORT 3 |
| N | P4 | 0C0H | PORT 4 |
| N | P5 | 091H | PORT 5 |
| N | P6 | 0A1H | PORT 6 |
| O | PCON | 087H | POWER CONTROL |
| N | PRBS | 0E4H | GSC PSEUDO-RANDOM SEQUENCE |
| O | PSW | 0D0H | PROGRAM STATUS WORD |
| N | RFIFO | 0F4H | GSC RECEIVE BUFFER |
| N | RSTAT | 0E8H | RECEIVE STATUS (DMA & GSC) |
| N | SARL0 | 0A2H | DMA SOURCE ADDR 0 (LOW) |
| N | SARH0 | 0A3H | DMA SOURCE ADDR 0 (HIGH) |
| N | SARL1 | 0B2H | DMA SOURCE ADDR 1 (LOW) |
| N | SARH1 | 0B3H | DMA SOURCE ADDR 1 (HIGH) |
| O | SBUF | 099H | LOCAL SERIAL CHANNEL (LSC) BUFFER |
| O | SCON | 098H | LOCAL SERIAL CHANNEL (LSC) CONTROL |
| N | SLOTTM | 0B4H | GSC SLOT TIME |
| O | SP | 081H | STACK POINTER |
| N | TCDCNT | 0D4H | GSC TRANSMIT COLLISION COUNTER |
| O | TCON | 088H | TIMER CONTROL |
| N | TFIFO | 085H | GSC TRANSMIT BUFFER |
| O | TH0 | 08CH | TIMER 0 (HIGH) |
| O | TH1 | 08DH | TIMER 1 (HIGH) |
| O | TL0 | 08AH | TIMER 0 (LOW) |
| O | TL1 | 08BH | TIMER 1 (LOW) |
| O | TMOD | 089H | TIMER MODE |
| N | TSTAT | 0D8H | TRANSMIT STATUS (DMA & GSC) |

TFIFO - (85H) TFIFO is used to access a 3-byte FIFO that contains the transmission data for the GSC.

TSTAT - (0D8H) Contains the DMA Service bit (DMA), Transmit Enable bit (TEN), Transmit FIFO Not Full bit (TFNF), Transmit Done bit (TDN), Transmit Collision Detect bit (TCDT), Underrun bit (UR), No Acknowledge bit (NOACK), and the Receive Data Line Idle bit (LNI). This register is used with both DMA and GSC.

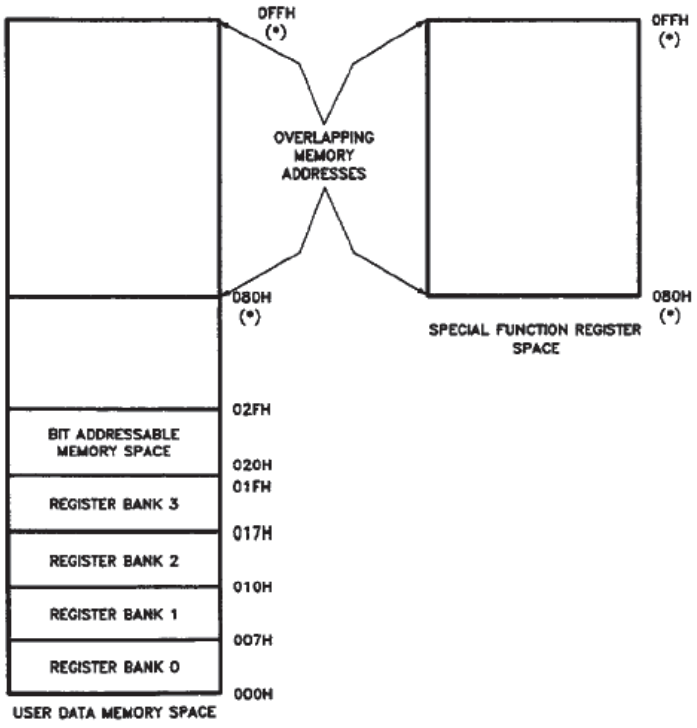
The general purpose flag bits (GF0 and GF1) that exist on the 80C51BH are no longer available on the C152. GF0 has been renamed GFIEN (GSC Flag Idle Enable) and is used to enable idle fill flags. Also GF1 has been renamed XRCLK (External Receive Clock Enable) and is used to enable the receiver to be clocked externally.

### 2.1.2 DATA MEMORY

Internal data memory consists of 256 bytes as shown in Figure 2.1. The first 128 bytes are addressed exactly like an 80C51BH, using direct addressing.

The addresses of the second 128 bytes of data memory happen to overlap the SFR addresses. The SFRs and their memory locations are shown in Figure 2.2. This means that internal data memory spaces have the same address as the SFR address. However, each type of memory is addressed differently. To access data memory above 80H, indirect addressing or the DMA channels must be used. To access the SFRs, direct addressing is used. When direct addressing is used, the address is the source or destination, e.g. MOV A, 10H, moves the contents of location 10H into the accumulator. When indirect addressing is used, the address of the destination or source exists within another register, e.g. MOV A, @R0. This instruction moves the contents of the memory location addressed by R0 into the accumulator. Directly addressing the locations 80H to 0FFH will access the SFRs. Another form of indirect addressing is with the use of Stack Pointer Operations. If the Stack Pointer contains an address and a PUSH or POP instruction is executed, indirect addressing is actually used. Directly accessing an unused SFR address will give undefined results.

Physically, there are separate SFR memory and data memory spaces allocated on the chip. Since there are separate spaces, the SFRs do not diminish the available data memory space.




*NOTE:
User data memory above 80H must be addressed indirectly. Using direct addressing above 80H accesses the Special Function Registers.

**Figure 2.1. Data Memory Map**

External data memory is accessed like an 80C51BH, with "MOVX" instructions. Addresses up to 64K may be accessed when using the Data Pointer (DPTR). When accessing external data memory with the DPTR, the address appears on Port 0 and 2. When using the DPTR, if less than 64K of external data memory is used, the address is emitted on all sixteen pins. This means that when using the DPTR, the pins of Port 2 not used for addresses cannot be used for general purpose I/O. An alternative to using 16-bit addresses with the DPTR is to use R0 or R1 to address the external data memory. When using the registers to address external data memory, the address range is limited to 256 bytes. However, software manipulation of I/O Port 2 pins as normal I/O, allows this 256 bytes restriction to be expanded via bank switching. When using R0 or R1 as data pointers, Port 2 pins that are not used for addressing, can be used as general purpose I/O.

### 2.1.2.1 Bit Addressable Memory

The C152 has several memory spaces in which the bits are directly addressed by their location. The directly addressable bits and their symbolic names are shown in Figure 2.3A, 2.3B, and 2.3C.

Bit addresses 0 to 7FH reside in on-board user data RAM in byte addresses 20H to 2FH (see Figure 2.3A).

Bit addresses 80H to 0FFH reside in the SFR memory space, but not every SFR is bit addressable, see Figure 2.3B. The addressable bits are scattered throughout the SFRs. The addressable bits occur every eighth SFR address starting at 80H and occupy the entire byte. Most of the bits that are addressable in the SFRs have been given symbolic names. These names will often be referred to in this or other documentation on the C152. Most assemblers also allow the use of the symbolic names when writing in assembly language. These names are shown in Figure 2.3C.

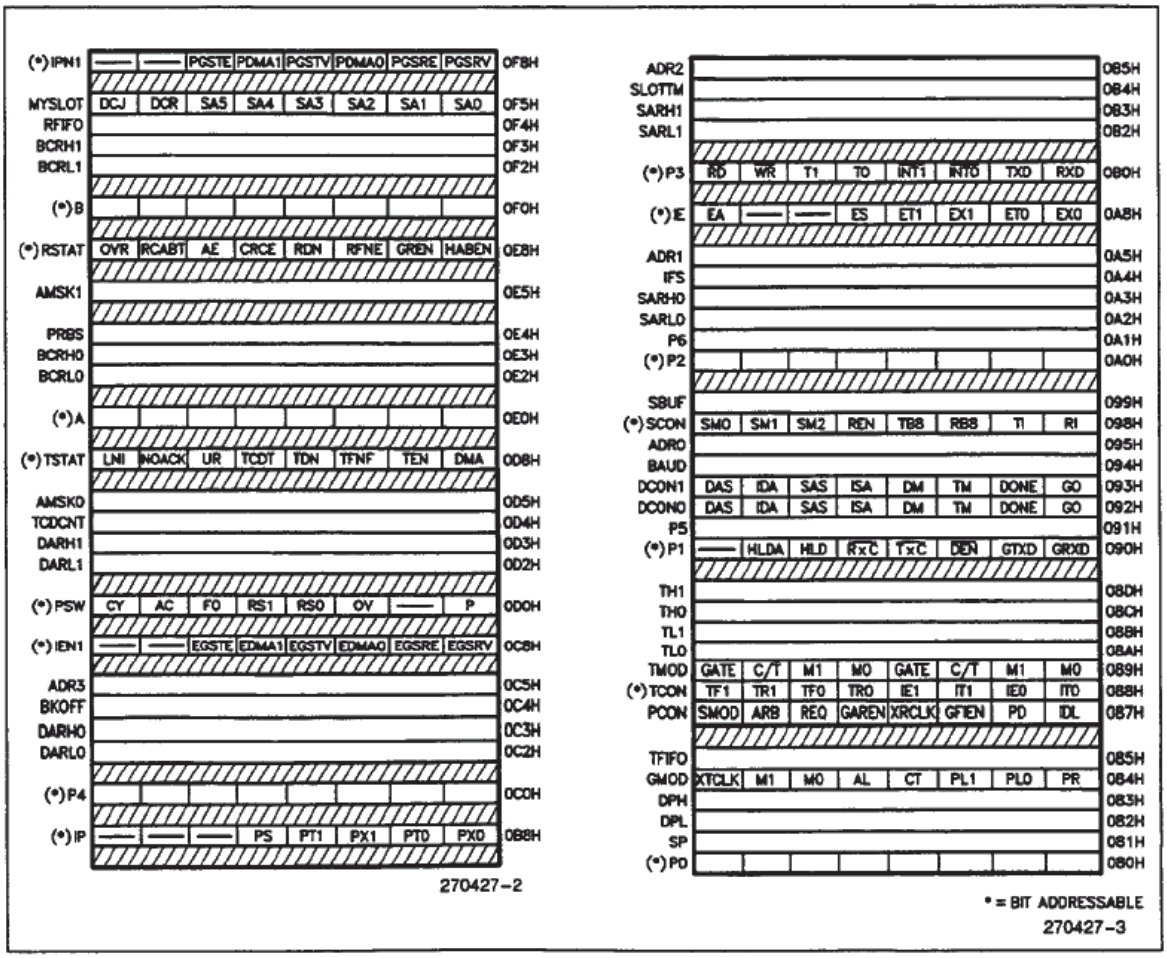



**Figure 2.2. Special Function Registers**

**Data Memory Map (bits):**

| Byte Address | BIT ADDRESSES (MSB) | | | | | | | (LSB) |
|---|---|---|---|---|---|---|---|---|
| 020H | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 021H | 0F | 0E | 0D | 0C | 0B | 0A | 09 | 08 |
| 022H | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| 023H | 1F | 1E | 1D | 1C | 1B | 1A | 19 | 18 |
| 024H | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 025H | 2F | 2E | 2D | 2C | 2B | 2A | 29 | 28 |
| 026H | 37 | 36 | 35 | 34 | 33 | 32 | 31 | 30 |
| 027H | 3F | 3E | 3D | 3C | 3B | 3A | 39 | 38 |
| 028H | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 |
| 029H | 4F | 4E | 4D | 4C | 4B | 4A | 49 | 48 |
| 02AH | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 |
| 02BH | 5F | 5E | 5D | 5C | 5B | 5A | 59 | 58 |
| 02CH | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 |
| 02DH | 6F | 6E | 6D | 6C | 6B | 6A | 69 | 68 |
| 02EH | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 |
| 02FH | 7F | 7E | 7D | 7C | 7B | 7A | 79 | 78 |

**Figure 2.3A. Bit Addresses**

| Byte Address | BIT ADDRESSES (MSB) | | | | | | | (LSB) | |
|---|---|---|---|---|---|---|---|---|---|
| 080H | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | (P0) |
| 088H | 8F | 8E | 8D | 8C | 8B | 8A | 89 | 88 | (TCON) |
| 090H | 97 | 96 | 95 | 94 | 93 | 92 | 91 | 90 | (P1) |
| 098H | 9F | 9E | 9D | 9C | 9B | 9A | 99 | 98 | (SCON) |
| 0A0H | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | (P2) |
| 0A8H | AF | – | – | AC | AB | AA | A9 | A8 | (IE) |
| 0B0H | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | (P3) |
| 0B8H | – | – | – | BC | BB | BA | B9 | B8 | (IP) |
| 0C0H | C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 | (P4) |
| 0C8H | – | – | CD | CC | CB | CA | C9 | C8 | (IEN1) |
| 0D0H | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | (PSW) |
| 0D8H | DF | DE | DD | DC | DB | DA | D9 | D8 | (TSTAT) |
| 0E0H | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 | (A) |
| 0E8H | EF | EE | ED | EC | EB | EA | E9 | E8 | (RSTAT) |
| 0F0H | F7 | F6 | F5 | F4 | F3 | F2 | F1 | F0 | (B) |
| 0F8H | – | – | FD | FC | FB | FA | F9 | F8 | (IPN1) |

**Figure 2.3B. Bit Addresses**

| Byte Address | SYMBOLIC NAME BIT MAP (MSB) | | | | | | | (LSB) | |
|---|---|---|---|---|---|---|---|---|---|
| 080H | P0.7 | P0.6 | P0.5 | P0.4 | P0.3 | P0.2 | P0.1 | P0.0 | (P0) |
| 088H | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 | (TCON) |
| 090H | P1.7 | P1.6 | P1.5 | P1.4 | P1.3 | P1.2 | P1.1 | P1.0 | (P1) |
| 098H | SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI | (SCON) |
| 0A0H | P2.7 | P2.6 | P2.5 | P2.4 | P2.3 | P2.2 | P2.1 | P2.0 | (P2) |
| 0A8H | EA | — | — | ES | ET1 | EX1 | ET0 | EX0 | (IE) |
| 0B0H | P3.7 | P3.6 | P3.5 | P3.4 | P3.3 | P3.2 | P3.1 | P3.0 | (P3) |
| 0B8H | — | — | — | PS | PT1 | PX1 | PT0 | PX0 | (IP) |
| 0C0H | P4.7 | P4.6 | P4.5 | P4.4 | P4.3 | P4.2 | P4.1 | P4.0 | (P4) |
| 0C8H | — | — | EGSTE | EDMA1 | EGSTV | EDMA0 | EGSRE | EGSRV | (IEN1) |
| 0D0H | CY | AC | F0 | RS1 | RS0 | OV | — | P | (PSW) |
| 0D8H | LNI | NOACK | UR | TCDT | TDN | TFNF | TEN | DMA | (TSTAT) |
| 0E0H | | | | | | | | | (A) |
| 0E8H | OVR | RCABT | AE | CRCE | RDN | RFNE | GREN | HABEN | (RSTAT) |
| 0F0H | | | | | | | | | (B) |
| 0F8H | — | — | PGSTE | PDMA1 | PGSTV | PDMA0 | PGSRE | PGSRV | (IPN1) |

**Figure 2.3C. Bit Addresses**

### 2.1.3 PROGRAM MEMORY

The 83C152 contains 8K of ROM program memory, and the 80C152 uses only external program memory. Figure 2.4 shows the program memory locations and where they reside. The user is allowed a maximum of 64K of program memory. In the 83C152 program memory fetches beyond 8K automatically access external program memory. When program memory is externally addressed, all of the Port 2 pins emit the address. Since all of Port 2 is affected by the address, unused address pins cannot be used as normal I/O ports even if less than 64K of memory is being accessed.

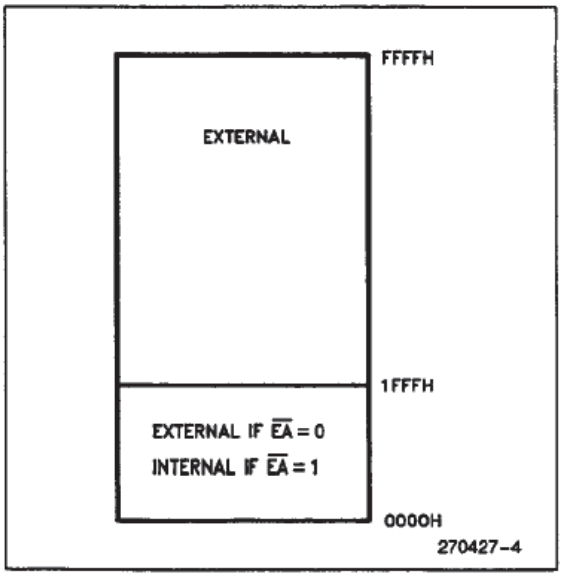



**Figure 2.4. Program Memory**

## 2.2 Interrupt Structure

The C152 retains all five interrupts of the 80C51BH. In addition, six new interrupts have been added for a total of 11 available interrupts. Two SFRs have been added to the C152 for control of the new interrupts. These added SFRs are IEN1 (C8H) for enabling the interrupts and IPN1 (F8H) for setting the priority. For an explanation on how the priority of interrupts affects their operation please refer to the MCS-51 Architecture and Hardware Chapters in the Intel Embedded Controller Handbook. A detailed description on how the interrupts function is in the MCS®-51 Architectural Overview.

| IEN1 FUNCTIONS | | | |
|---|---|---|---|
| **Symbol** | **Position** | **Vector** | **Function** |
| — | IEN1.7 | | **RESERVED** and do not exist on chip. |
| — | IEN1.6 | | **RESERVED** and do not exist on chip. |
| EGSTE | IEN1.5 | 04BH | **GSC TRANSMIT ERROR**—The interrupt service routine at 4BH is invoked if NOACK or TCDT is set when the GSC is under CPU control and EGSTE is enabled. This interrupt service routine is invoked if NOACK, TCDT, or UR is set when the GSC is under DMA control and EGSTE is enabled. |
| EDMA1 | IEN1.4 | 053H | **DMA CHANNEL REQUEST 1**—The interrupt service routine at 53H is invoked when DCON1.1 (DONE) is set and EDMA1 is enabled. |
| EGSTV | IEN1.3 | 043H | **GSC TRANSMIT VALID**—The interrupt service routine at 43H is invoked if TFNF is set when the GSC is under CPU control and EGSTV is enabled. This interrupt service routine is invoked if TDN is set when the GSC is under DMA control and EGSTV is enabled. |
| EDMA0 | IEN1.2 | 03BH | **DMA CHANNEL REQUEST 0**—The interrupt service routine at 3BH will be invoked when DCON0.1 (DONE) is set and EDMA0 is enabled. |
| EGSRE | IEN1.1 | 033H | **GSC RECEIVE ERROR**—The interrupt service routine at 33H is invoked if CRCE, OVR, RCABT, or AE is set when the GSC is under CPU or DMA control and EGSRE is enabled. |
| EGSRV | IEN1.0 | 02BH | **GSC RECEIVE VALID**—The interrupt service routine at 2BH is invoked if RFNE is set when the GSC is under CPU control and EGSRV is enabled. This interrupt service routine is invoked if RDN is set when the GSC is under DMA control and EGSRV is enabled. |

IPN1 is used the same way the current 80C51BH interrupt priority register (IP) is. By assigning a "1" to the appropriate bit, that interrupt has a higher priority than an interrupt with a "0" assigned to it in the priority register.

The new interrupt priority register (IPN1) contents are:

| Symbol | Position | Function |
|---|---|---|
| PGSTE | IPN1.5 | GSC TRANSMIT ERROR |
| PDMA1 | IPN1.4 | DMA CHANNEL REQUEST 1 |
| PGSTV | IPN1.3 | GSC TRANSMIT VALID |
| PDMA0 | IPN1.2 | DMA CHANNEL REQUEST 0 |
| PGSRE | IPN1.1 | GSC RECEIVE ERROR |
| PGSRV | IPN1.0 | GSC RECEIVE VALID |

The eleven interrupts are sampled in the following order when assigned the same priority level in the IP and IPN1 registers:

| Priority Sequence | Priority Symbolic Address | Priority Symbolic Name | Interrupt Symbolic Address | Interrupt Symbolic Name | Vector Address | |
|---|---|---|---|---|---|---|
| 1 | IP.0 | PX0 | IE.0 | EX0 | 03H | (FIRST) |
| 2 | IPN1.0 | PGSRV | IEN1.0 | EGSRV | 2BH | |
| 3 | IP.1 | PT0 | IE.1 | ET0 | 0BH | |
| 4 | IPN1.1 | PGSRE | IEN1.1 | EGSRE | 33H | |
| 5 | IPN1.2 | PDMA0 | IEN1.2 | EDMA0 | 3BH | |
| 6 | IP.2 | PX1 | IE.2 | EX1 | 13H | |
| 7 | IPN1.3 | PGSTV | IEN1.3 | EGSTV | 43H | |
| 8 | IPN1.4 | PDMA1 | IEN1.4 | EDMA1 | 53H | |
| 9 | IP.3 | PT1 | IE.3 | ET1 | 1BH | |
| 10 | IPN1.5 | PGSTE | IEN1.5 | EGSTE | 4BH | |
| 11 | IP.4 | PS | IE.4 | ES | 23H | (LAST) |

## 2.3 Reset

RESET performs the same operations in both the 80C51BH and the C152 and those conditions that exist at the end of a valid RESET are:

| Register | Contents | Register | Contents |
|---|---|---|---|
| ACC | 00H | P0–P6 | 0FFH |
| ADR0-3 | 00H | PCON | 0XXX0000B |
| AMSK0 | 00H | PRBS | 00H |
| AMSK1 | 00H | PSW | 00H |
| B | 00H | RFIFO | INDETERMINATE |
| BAUD | 00H | RSTAT | 00000000B |
| BCRH0 | INDETERMINATE | SARH0 | INDETERMINATE |
| BCRH1 | INDETERMINATE | SARH1 | INDETERMINATE |
| BCRL0 | INDETERMINATE | SARL0 | INDETERMINATE |
| CRL1 | INDETERMINATE | SARL1 | INDETERMINATE |
| BKOFF | INDETERMINATE | SBUF | INDETERMINATE |
| DARH0 | INDETERMINATE | SCON | 00H |
| DARH1 | INDETERMINATE | SLOTTM | 00H |
| DARL0 | INDETERMINATE | SP | 07H |
| DARL1 | INDETERMINATE | TCDCNT | INDETERMINATE |
| DCON0 | 00H | TCON | 00H |
| DCON1 | 00H | TFIFO | INDETERMINATE |
| DPTR | 0000H | TH0 | 00H |
| GMOD | X0000000B | TH1 | 00H |
| IE | 0XX00000B | TL0 | 00H |
| IEN1 | XX000000B | TL1 | 00H |
| IFS | 00H | TMOD | 00H |
| IP | XXX00000B | TSTAT | XX000100B |
| IPN1 | XX000000B | PC | 0000H |
| MYSLOT | 00000000B | | |

The same conditions apply for both the 80C51BH and C152 for a correct reset pulse or "power-on" reset except that $\overline{\text{Reset}}$ is active low on the C152. Please refer to the 8051/52 Hardware Description Chapter of the Intel Embedded Controller Handbook for an explanation on how to provide a proper power-on reset. Since $\overline{\text{Reset}}$ is active low on the C152, the resistor should be tied to VCC and the capacitor should be tied to VSS.

Because the clocking on part of the GSC circuitry is independent of the processor clock, data may still be transmitted and $\overline{\text{DEN}}$ active for some time after reset is applied. The transmission may continue for a maximum of four machine cycles after reset is first pulled low. Although $\overline{\text{Reset}}$ has to be held low for only three machine cycles to be recognized by the GSC hardware, all of the GSC circuitry may not be reset until four machine cycles have passed. If it is important in the user application that all transmission and $\overline{\text{DEN}}$ becomes inactive at the end of a reset, then $\overline{\text{Reset}}$ will have to be held low for a minimum of four machine cycles.

## 2.4 Ports 4, 5 and 6

Ports 4, 5 and 6 operation is identical to Ports 1-3 on the 80C51BH. The description of port operation can be found in the 8051/52 Hardware Description Chapter of the Intel Embedded Controller Handbook. Ports 5 and 6 exist only on the "JB" and "JD" version of the C152 and can either function as standard I/O ports or can be configured so that program memory fetches are performed with these two ports. To configure ports 5 and 6 as standard I/O ports, EBEN is tied to a logic low. When in this configuration, ports 5 and 6 operation is identical to that of port 4 except they are not bit addressable. To configure ports 5 and 6 to fetch program memory, EBEN is tied to a logic high. When using ports 5 and 6 to fetch the program memory, the signal $\overline{\text{EPSEN}}$ is used to enable the external memory device instead of $\overline{\text{PSEN}}$. Regardless of which ports are used to fetch program memory, all data memory fetches occur over ports 0 and 2. The 80C152JB and 80C152JD are available as ROMless devices only. ALE is still used to latch the address in all configurations. Table 2.1 summarizes the control signals and how the ports may be used.

## 2.5 Timer/Counters

The 80C51BH and C152 have the same pair of 16-bit general purpose timer/counters. The user should refer to the Intel Embedded Controller Handbook which describes the timer/counters and their use. The user should bear in mind, when reading the Intel Embedded Controller Handbook that the C152 does not have the third event timer named Timer 2, which is in the 8052.

## 2.6 Package

The 83C152 is packaged in a 48 pin DIP and a 68 lead PLCC. This differs from the 40 pin DIP and 44 pin PLCC of the 80C51BH. The larger package is required to accommodate the extra 8 bit I/O port (P4). Figures 2.5A, 2.5B and 2.5C show the packages and the pin names.

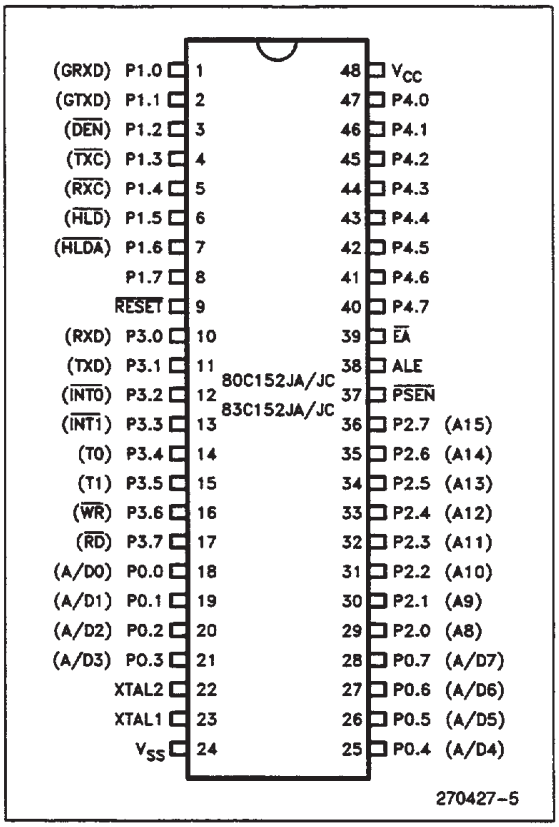



**Figure 2.5A. DIP Pin Out**

**Table 2.1 Program Memory Fetches**

| EBEN | $\overline{\text{EA}}$ | Program Fetch via | $\overline{\text{PSEN}}$ | $\overline{\text{EPSEN}}$ | Comments |
|---|---|---|---|---|---|
| 0 | 0 | P0, P2 | Active | Inactive | Addresses 0–0FFFFH |
| 0 | 1 | N/A | N/A | N/A | Invalid Combination |
| 1 | 0 | P5, P6 | Inactive | Active | Addresses 0–0FFFFH |
| 1 | 1 | P5, P6<br>P0, P2 | Inactive<br>Active | Active<br>Inactive | Addresses 0–1FFFH<br>Addresses ≥ 2000H |

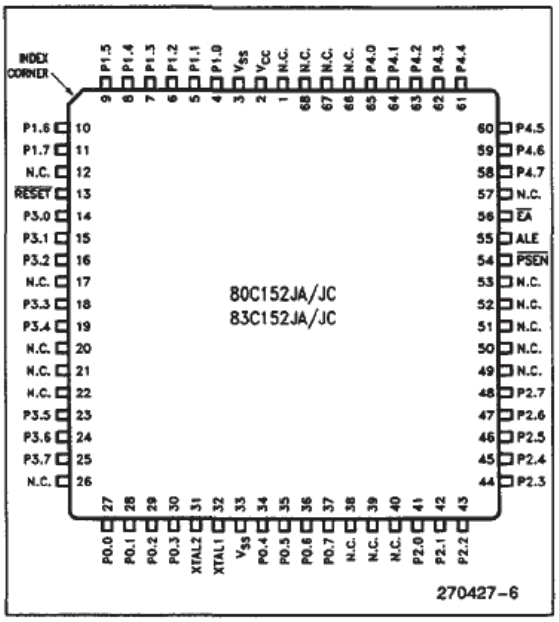



Figure 2.5B. PLCC Pin Out

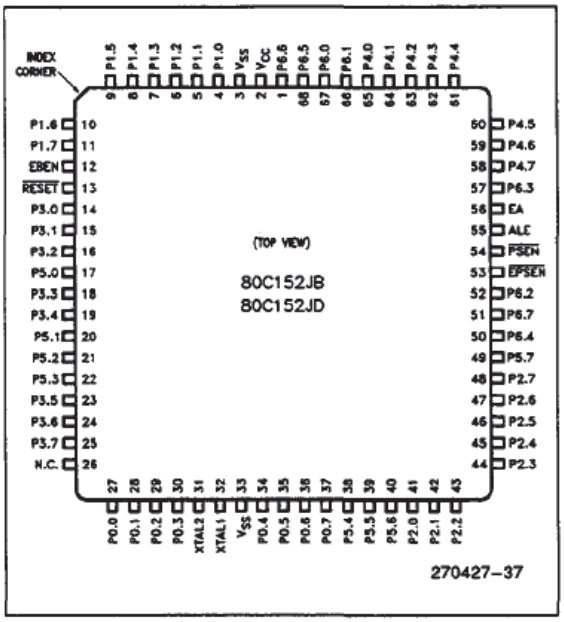



Figure 2.5C. PLCC Pin Out

## 2.7 Pin Description

The pin description for the 80C51BH also applies to the C152 and is listed below. Changes have been made to the descriptions as they apply to the C152.

### PIN DESCRIPTION

| Pin # | | Description |
|---|---|---|
| DIP | PLCC(1) | |
| 48 | 2 | $V_{CC}$—Supply voltage. |
| 24 | 3, 33(2) | **VSS**—Circuit ground. |
| 18–21, 25–28 | 27–30, 34–37 | **Port 0**—Port 0 is an 8-bit open drain bi-directional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.<br>Port 0 is also the multiplexed low-order address and data bus during accesses to external program memory if EBEN is pulled low. During accesses to external Data Memory, Port 0 always emits the low-order address byte and serves as the multiplexed data bus. In these applications it uses strong internal pullups when emitting 1s.<br>Port 0 also outputs the code bytes during program verification. External pullups are required during program verification. |

**NOTES:**
1. N.C. pins on PLCC package may be connected to internal die and should not be used in customer applications.
2. It is recommended that both Pin 3 and Pin 33 be grounded for PLCC devices.

## PIN DESCRIPTION (Continued)

<table>
<tr><th colspan="2">Pin #</th><th colspan="3" rowspan="2">Description</th></tr>
<tr><th>DIP</th><th>PLCC(1)</th></tr>
<tr><td>1–8</td><td>4–11</td><td colspan="3">Port 1—Port 1 is an 8-bit bidirectional I/O port with internal pullups. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current ($I_{IL}$, on the data sheet) because of the internal pullups.<br>Port 1 also serves the functions of various special features of the 8XC152, as listed below:
<table>
<tr><th>Pin</th><th>Name</th><th>Alternate Function</th></tr>
<tr><td>P1.0</td><td>GRXD</td><td>GSC data input pin</td></tr>
<tr><td>P1.1</td><td>GTXD</td><td>GSC data output pin</td></tr>
<tr><td>P1.2</td><td>$\overline{\text{DEN}}$</td><td>GSC enable signal for an external driver</td></tr>
<tr><td>P1.3</td><td>$\overline{\text{TXC}}$</td><td>GSC input pin for external transmit clock</td></tr>
<tr><td>P1.4</td><td>$\overline{\text{RXC}}$</td><td>GSC input pin for external receive clock</td></tr>
<tr><td>P1.5</td><td>$\overline{\text{HLD}}$</td><td>DMA hold input/output</td></tr>
<tr><td>P1.6</td><td>$\overline{\text{HLDA}}$</td><td>DMA hold acknowledge input/output</td></tr>
</table></td></tr>
<tr><td>29–36</td><td>41–48</td><td colspan="3">Port 2—Port 2 is an 8-bit bi-directional I/O port with internal pullups. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current ($I_{IL}$, on the data sheet) because of the internal pullups.<br>Port 2 emits the high-order address byte during fetches from external Program Memory if EBEN is pulled low. During accesses to external Data Memory that use 16-bit addresses (MOVX @ DPTR and DMA operations), Port 2 emits the high-order address byte. In these applications it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @ Ri), Port 2 emits the contents of the P2 Special Function Register.<br>Port 2 also receives the high-order address bits during program verification.</td></tr>
<tr><td>10–17</td><td>14–16, 18, 19, 23–25</td><td colspan="3">Port 3—Port 3 is an 8-bit bi-directional I/O port with internal pullups. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current ($I_{IL}$, on the data sheet) because of the pullups.<br>Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:
<table>
<tr><th>Pin</th><th>Name</th><th>Alternate Function</th></tr>
<tr><td>P3.0</td><td>RXD</td><td>Serial input line</td></tr>
<tr><td>P3.1</td><td>TXD</td><td>Serial output line</td></tr>
<tr><td>P3.2</td><td>$\overline{\text{INT0}}$</td><td>External interrupt 0</td></tr>
<tr><td>P3.3</td><td>$\overline{\text{INT1}}$</td><td>External interrupt 1</td></tr>
<tr><td>P3.4</td><td>T0</td><td>Timer 0 external input</td></tr>
<tr><td>P3.5</td><td>T1</td><td>Timer 1 external input</td></tr>
<tr><td>P3.6</td><td>$\overline{\text{WR}}$</td><td>External Data Memory Write strobe</td></tr>
<tr><td>P3.7</td><td>$\overline{\text{RD}}$</td><td>External Data Memory Read strobe</td></tr>
</table></td></tr>
<tr><td>47–40</td><td>65–58</td><td colspan="3">Port 4—Port 4 is an 8-bit bi-directional I/O port with internal pullups. Port 4 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 4 pins that are externally being pulled low will source current ($I_{IL}$, on the data sheet) because of the internal pullups. In addition, Port 4 also receives the low-order address bytes during program verification.</td></tr>
</table>

**NOTES:**
1. N.C. pins on PLCC package may be connected to internal die and should not be used in customer applications.
2. It is recommended that both Pin 3 and Pin 33 be grounded for PLCC devices.

## PIN DESCRIPTION (Continued)

| Pin # | | Description |
|---|---|---|
| DIP | PLCC(1) | |
| 9 | 13 | **$\overline{RST}$**—Reset input. A logic low on this pin for three machine cycles while the oscillator is running resets the device. An internal pullup resistor permits a power-on reset to be generated using only an external capacitor to $V_{SS}$. Although the GSC recognizes the reset after three machine cycles, data may continue to be transmitted for up to 4 machine cycles after Reset is first applied. |
| 38 | 55 | **ALE**—Address Latch Enable output signal for latching the low byte of the address during accesses to external memory.<br>In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory. While in Reset, ALE remains at a constant high level. |
| 37 | 54 | **$\overline{PSEN}$**—Program Store Enable is the Read strobe to External Program Memory. When the 8XC152 is executing from external program memory, $\overline{PSEN}$ is active (low). When the device is executing code from External Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to External Data Memory. While in Reset, $\overline{PSEN}$ remains at a constant high level. |
| 39 | 56 | **$\overline{EA}$**—External Access enable. $\overline{EA}$ must be externally pulled low in order to enable the 8XC152 to fetch code from External Program Memory locations 0000H to 0FFFH.<br>$\overline{EA}$ must be connected to $V_{CC}$ for internal program execution. |
| 23 | 32 | **XTAL1**—Input to the inverting oscillator amplifier and input to the internal clock generating circuits. |
| 22 | 31 | **XTAL2**—Output from the oscillator amplifier. |
| N/A | 17, 20<br>21, 22<br>38, 39<br>40, 49 | **Port 5**—Port 5 is an 8-bit bi-directional I/O port with internal pullups. Port 5 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 5 pins that are externally being pulled low will source current ($I_{IL}$, on the data sheet) because of the internal pullups.<br>Port 5 is also the multiplexed low-order address and data bus during accesses to external program memory if EBEN is pulled high. In this application it uses strong pullups when emitting 1s. |
| N/A | 67, 66<br>52, 57<br>50, 68<br>1, 51 | **Port 6**—Port 6 is an 8-bit bi-directional I/O port with internal pullups. Port 6 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 6 pins that are externally pulled low will source current ($I_{IL}$, on the data sheet) because of the internal pullups.<br>Port 6 emits the high-order address byte during fetches from external Program Memory if EBEN is pulled high. In this application it uses strong pullups when emitting 1s. |
| N/A | 12 | **EBEN**—E-Bus Enable input that designates whether program memory fetches take place via Ports 0 and 2 or Ports 5 and 6. Table 2.1 shows how the ports are used in conjunction with EBEN. |
| | 53 | **$\overline{EPSEN}$**—E-bus Program Store Enable is the Read strobe to external program memory when EBEN is high. Table 2.1 shows when $\overline{EPSEN}$ is used relative to $\overline{PSEN}$ depending on the status of EBEN and $\overline{EA}$. |

**NOTES:**
1. N.C. pins on PLCC package may be connected to internal die and should not be used in customer applications.
2. It is recommended that both Pin 3 and Pin 33 be grounded for PLCC devices.

## 2.8 Power Down and Idle

Both of these operations function identically as in the 80C51BH. Application Note 252, "Designing with the 80C51BH" gives an excellent explanation on the use of the reduced power consumption modes. Some of the items not covered in AP-252 are the considerations that are applicable when using the GSC or DMA in conjunction with the power saving modes.

The GSC continues to operate in Idle as long as the interrupts are enabled. The interrupts need to be enabled, so that the CPU can service the FIFO's. In order to properly terminate a reception or transmission the C152 must not be in idle when the EOF is transmitted or received. After servicing the GSC, user software will need to again invoke the Idle command as the CPU does not automatically re-enter the Idle mode after servicing the interrupts.

The GSC does not operate while in Power Down so the steps required prior to entering Power Down become more complicated. The sequence when entering Power Down and the status of the I/O is of major importance in preventing damage to the C152 or other components in the system. Since the only way to exit Power Down is with a Reset, several problem areas become very significant. Some of the problems that merit careful consideration are cases where the Power Down occurs during the middle of a transmission, and the possibility that other stations are not or cannot enter this same mode. The state of the GSC I/O pins becomes critical and the GSC status will need to be saved before power down is entered. There will also need to be some method of identifying to the CPU that the following Reset is probably not a cold start and that other stations on the link may have already been initialized.

The DMA circuitry stops operation in both Idle and Power Down modes. Since operation is stopped in both modes, the process should be similar in each case. Specific steps that need to be taken include: notification to other devices that DMA operation is about to cease for a particular station or network, proper withdrawal from DMA operation, and saving the status of the DMA channels. Again, the status of the I/O pins during Power Down needs careful consideration to avoid damage to the C152 or other components.

Port 4 returns to its input state, which is high level using weak pullup devices.

## 2.9 Local Serial Channel

The Local Serial Channel (LSC) is the name given to the UART that exists on all MCS-51 devices. The LSC's function and operation is exactly the same as on the 80C51BH. For a description on the use of the LSC, refer to the 8051/52 Hardware Description Chapter in the Intel Embedded Controller Handbook, under Serial Interface.

# 3.0 GLOBAL SERIAL CHANNEL

## 3.1 Introduction

The Global Serial Channel (GSC) is a multi-protocol, high performance serial interface targeted for data rates up to 2 MBPS with on-chip clock recovery, and 2.4 MBPS using the external clock options. In applications using the serial channel, the GSC implements the Data Link Layer and Physical Link Layer as described in the ISO reference model for open systems interconnection.

The GSC is designed to meet the requirements of a wide range of serial communications applications and is optimized to implement Carrier-Sense Multi-Access with Collision Detection (CSMA/CD) and Synchronous Data Link Control (SDLC) protocols. The GSC architecture is also designed to provide flexibility in defining non-standard protocols. This provides the ability to retrofit new products into older serial technologies, as well as the development of proprietary interconnect schemes for serial backplane environments.

The versatility of the GSC is demonstrated by the wide range of choices available to the user. The various modes of operation are summarized in Table 3.1. In subsequent sections, each available choice of operation will be explained in detail.

In using Table 3.1, the parameters listed vertically (on the left hand side) represent an option that is selected (X). The parameters listed horizontally (along the top of the table) are all the parameters that could theoretically be selected (Y). The symbol at the junction of both X and Y determines the applicability of the option Y.

Note, that not all combinations are backwards compatible. For example, Manchester encoding requires half duplex, but half duplex does not require Manchester encoding.

Table 3.1

| AVAILABLE OPTIONS → / ↓ SELECTED FUNCTION | DATA ENCODING | | | FLAGS | | CRC | | | DU-PLEX | | ACKNOW-LEDGE | | | ADDRESS RECOG-NITION | | | BACKOFF | | | PRE-AMBLE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N=NOT AVAILABLE<br>M=MANDATORY<br>O=OPTIONAL<br>P=NORMALLY PREFERRED<br>X=N/A | MANCHESTER | NRZ | NRZI | 01111110 | 11/IDLE | NONE | 16BITCCIT | 32BITAUTO | HALF | FULL | NONE | HARDWARE | USERDEFINED | NONE/ALL | 8BIT | 16BIT | NORMAL | ALTERNATE | DETERMINISTIC | NONE | 8BIT |
| DATA ENCODING: | | | | | | | | | | | | | | | | | | | | | |
| MANCHESTER(CSMA/CD) | X | N | N | 1 | P | 1 | O | O | M | N | O | O | O | O | O | O | O | O | O | N | O |
| NRZI (SDLC) | N | X | N | P | 1 | 1 | O | O | O | O | O | N | P | O | O | O | N | N | N | O | O |
| NRZ (EXT CLK) | N | N | X | O | O | 1 | O | O | O | O | O | N | O | O | O | O | O | O | O | O | O |
| FLAGS:01111110 (SDLC) | N | P | O | X | 1 | 1 | O | O | O | O | O | N | P | O | O | O | N | N | N | O | O |
| 11/IDLE | P | N | O | 1 | X | 1 | O | O | O | N | O | O | O | O | O | O | O | O | O | 1 | O |
| CRC:NONE | 1 | 1 | 1 | 1 | 1 | X | N | N | 1 | N | 1 | 1 | 1 | 1 | 1 | 1 | N | N | N | 1 | 1 |
| 16-BIT CCITT | O | O | O | O | O | N | X | N | O | O | O | O | O | O | O | O | O | O | O | O | O |
| 32-BIT AUTODIN II | O | O | O | O | O | N | N | X | O | N | O | O | O | O | O | O | O | O | O | O | O |
| DUPLEX:HALF | O | O | O | O | O | 1 | O | O | X | N | O | O | O | O | O | O | O | O | O | O | O |
| FULL | N | O | O | M | N | N | M | N | N | X | O | N | P | O | O | O | N | N | N | O | O |
| ACKNOWLEDGEMENT:NONE | O | O | O | O | O | 1 | O | O | O | O | X | N | N | O | O | O | O | O | O | O | O |
| HARDWARE | O | N | N | N | O | 1 | O | O | O | N | N | X | N | O | O | O | N | O | O | N | O |
| USER DEFINED | O | P | O | O | O | 1 | O | O | O | P | N | N | X | O | O | O | O | O | O | O | O |
| ADDRESS RECOGNITION: | | | | | | | | | | | | | | | | | | | | | |
| NONE/ALL | O | O | O | O | O | 1 | O | O | O | O | O | O | O | X | N | N | O | O | O | O | O |
| 8-BIT | O | O | O | O | O | 1 | O | O | O | O | O | O | O | N | X | N | O | O | O | O | O |
| 16-BIT | O | O | O | O | O | 1 | O | O | O | O | O | O | O | N | N | X | O | O | O | O | O |
| COLLISION RESOLUTION: | | | | | | | | | | | | | | | | | | | | | |
| NORMAL | O | N | O | N | O | N | O | O | M | N | O | N | O | O | O | O | X | N | N | N | O |
| ALTERNATE | O | N | O | N | O | N | O | O | M | N | O | O | O | O | O | O | N | X | N | N | O |
| DETERMINISTIC | O | N | O | N | O | N | O | O | M | N | O | O | O | O | O | O | N | N | X | N | O |
| PREAMBLE:NONE | N | O | O | O | 1 | 1 | O | O | O | O | O | N | O | O | O | O | N | N | N | X | N |
| 8-BIT | O | O | O | O | O | 1 | O | O | O | O | O | O | O | O | O | O | O | O | O | N | X |
| 32-BIT | O | O | O | O | O | 1 | O | O | O | O | O | O | O | O | O | O | O | O | O | N | N |
| 64-BIT | O | O | O | O | O | 1 | O | O | O | O | O | O | O | O | O | O | O | O | O | N | N |
| JAM:D.C. | M | N | N | N | O | N | O | O | M | N | O | O | O | O | O | O | O | O | O | N | O |
| $\overline{CRC}$ | M | N | N | N | O | N | O | O | M | N | O | O | O | O | O | O | O | O | O | N | O |
| CLOCKING:EXTERNAL | N | M | N | O | O | N | O | O | O | O | O | N | O | O | O | O | N | N | N | O | O |
| INTERNAL | O | O | N | O | O | O | O | O | O | O | O | O | O | O | O | O | O | O | O | O | O |
| CONTROL: CPU | O | O | O | O | O | 1 | O | O | O | O | O | O | O | O | O | O | O | O | O | O | O |
| DMA | O | O | O | O | O | 1 | O | O | O | O | O | O | O | O | O | O | O | O | O | O | O |
| RAW RECEIVE: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | N | 1 | 1 | 1 | 1 | 1 | 1 | O | O | O | 1 | 1 |
| RAW TRANSMIT: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | N | 1 | 1 | 1 | 1 | 1 | 1 | N | N | N | 1 | 1 |
| CSMA/CD: | O | N | N | 1 | P | 1 | O | O | M | N | O | O | O | O | O | O | O | O | O | N | O |
| SDLC: | N | O | O | P | 1 | 1 | O | O | O | O | O | N | O | O | O | O | N | N | N | P | O |

Table 3.1 (Continued)

| AVAILABLE OPTIONS → / ↓ SELECTED FUNCTION | PREAMBLE | | JAM | | CLOCK | | CONTROL | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 32 BIT | 64 BIT | DC | CRC/ | EXTERNAL | INTERNAL | CPU | DMA | RAW RECEIVE | RAW TRANSMIT | CSMA/CD | SDLC |
| DATA ENCODING: | | | | | | | | | | | | |
| MANCHESTER | O | O | O | O | N | M | O | O | O | O | M | N |
| NRZI | O | O | N | N | N | M | O | O | O | O | N | M |
| NRZ | O | O | O | O | M | N | O | O | O | O | O | O |
| FLAGS:01111110 | O | O | N | N | O | O | O | O | O | 1 | 1 | P |
| 11/IDLE | O | O | O | O | O | O | O | O | O | 1 | P | 1 |
| CRC:NONE | 1 | 1 | N | N | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 16-BIT CCITT | O | O | O | O | O | O | O | O | 1 | 1 | O | O |
| 32-BIT AUTODIN II | O | O | O | O | O | O | O | O | 1 | 1 | O | O |
| DUPLEX:HALF | O | O | O | O | O | O | O | O | O | O | O | O |
| FULL | O | O | N | N | O | O | O | O | N | N | N | P |
| ACKNOWLEDGEMENT:NONE | O | O | O | O | O | O | O | O | O | O | O | O |
| HARDWARE | O | O | O | O | N | O | O | O | N | N | O | N |
| USER DEFINED | O | O | O | O | O | O | O | O | O | O | O | 1 |
| ADDRESS RECOGNITION: | | | | | | | | | | | | |
| NONE | O | O | O | O | O | O | O | O | O | O | O | O |
| 8-BIT | O | O | O | O | O | O | O | O | 1 | 1 | O | O |
| 16-BIT | O | O | O | O | O | O | O | O | 1 | 1 | O | O |
| COLLISION RESOLUTION: | | | | | | | | | | | | |
| NORMAL | O | O | O | O | N | O | O | O | O | N | M | N |
| ALTERNATE | O | O | O | O | N | O | O | O | O | N | M | N |
| DETERMINISTIC | O | O | O | O | N | O | O | O | O | N | M | N |
| PREAMBLE:NONE | N | N | N | N | O | O | O | O | O | O | N | P |
| 8-BIT | N | N | O | O | O | O | O | O | 1 | 1 | O | O |
| 32-BIT | X | N | O | O | O | O | O | O | 1 | 1 | O | O |
| 64-BIT | N | X | O | O | O | O | O | O | 1 | 1 | O | O |
| JAM:D.C. | O | O | X | N | N | O | O | O | O | N | M | N |
| $\overline{CRC}$ | O | O | N | X | N | O | O | O | O | N | M | N |
| CLOCKING:EXTERNAL | O | O | N | N | X | N | O | O | O | O | N | O |
| INTERNAL | O | O | O | O | N | X | O | O | O | O | O | O |
| CONTROL:CPU | O | O | O | O | O | O | X | N | O | O | O | O |
| DMA | O | O | O | O | O | O | N | X | O | O | O | O |
| RAW RECEIVE: | 1 | 1 | O | O | 1 | 1 | 1 | 1 | X | N | 1 | 1 |
| RAW TRANSMIT: | 1 | 1 | N | N | 1 | 1 | 1 | 1 | N | X | 1 | 1 |
| CSMA/CD: | O | O | O | O | N | O | O | O | O | O | X | N |
| SDLC: | O | O | N | N | O | O | O | O | O | O | N | X |

N = NOT AVAILABLE
M = MANDATORY
O = OPTIONAL
P = NORMALLY PREFERRED
X = N/A

Note 1: Programmable in Raw transmit or receive mode.

Almost all the options available from Table 3.1 can be implemented with the proper software to perform the functions that are necessary for the options selected. In Table 3.1, a judgment has been made by the authors on which options are practical and which are not. What this means is that in Table 3.1, an "N" should be interpreted as meaning that the option is either not practical when implemented with user software or that it cannot be done. An "O" is used when that function is one of several that can be implemented with the GSC without additional user software.

The GSC is targeted to operate at bit rates up to 2.4 MBps using the external clock options and up to 2 MBps using the internal baud rate generator, internal data formatting and on-chip clock recovery. The baud rate generator allows most standard rates to be achieved. These standards include the proposed IEEE802.3 LAN standard (1.0MBps) and the T1 standard (1.544MBps). The baud rate is derived from the crystal frequency. This makes crystal selection important when determining the frequency and accuracy of the baud rate.

The user needs to be aware that after reset, the GSC is in CSMA/CD mode, IFS = 256 bit times, and a bit time equals 8 oscillator periods. The GSC will remain in this mode until the interframe space expires. If the user changes to SDLC mode or the parameters used in CSMA/CD, these changes will not take effect until the interframe space expires. A requirement for the interframe space timer to begin is that the receiver be in an idle state. This makes it possible for the GSC to be in some other mode than the user intends for a significant amount of time after reset. To prevent unwanted GSC errors from occurring, the user should not enable the GSC or the GSC interrupts for 170 machine cycles ( (256 × 8)/12) after LNI bit is set.

## 3.2 CSMA/CD Operation

### 3.2.1 CSMA/CD OVERVIEW

CSMA/CD operates by sensing the transmission line for a carrier, which indicates link activity. At the end of link activity, a station must wait a period of time, called the deference period, before transmission may begin. The deference period is also known as the interframe space. The interframe space is explained in Section 3.2.3.

With this type of operation, there is always the possibility of a collision occurring after the deference period due to line delays. If a collision is detected after transmission is started, a jamming mechanism is used to ensure that all stations monitoring the line are aware of the collision. A resolution algorithm is then executed to resolve the contention. There are three different modes of collision resolution made available to the user on the C152. Re-transmission is attempted when a resolution algorithm indicates that a station's opportunity has arrived.

Normally, in CSMA/CD, re-transmission slot assignments are intended to be random. This method gives all stations an equal opportunity to utilize the serial communication link but also leaves the possibility of another collision due to two stations having the same slot assignment. There is an option on the C152 which allows all the stations to have their slot assignments previously determined by user software. This pre-assignment of slots is called the deterministic resolution mode. This method allows resolution after the first collision and ensures the access of the link to each station during the resolution. Deterministic resolution can be advantageous when the link is being heavily used and collisions are frequently occurring and in real time applications where determinism is required. Deterministic resolution may also be desirable if it is known beforehand that a certain station's communication needs to be prioritized over those of other stations if it is involved in a collision.

### 3.2.2 CSMA/CD FRAME FORMAT

The frame format in CSMA/CD consists of a preamble, Beginning of Frame flag (BOF), address field, information field, CRC, and End of Frame flag (EOF) as shown in Figure 3.1.


**Figure 3.1 Typical CSMA/CD Frame**

PREAMBLE - The preamble is a series of alternating 1s and 0s. The length of the preamble is programmable to be 0, 8, 32, or 64 bits. The purpose of the preamble is to allow all the receivers to synchronize to the same clock edges and identifies to the other stations on-line that there is activity indicating the link is being used. For these reasons zero preamble length is not compatible with standard CSMA/CD, protocols. When using CSMA/CD, the BOF is considered part of the preamble compared to SDLC, where the BOF is not part of the preamble. This means that if zero preamble length were to be used in CSMA/CD mode, no BOF would be generated. It is strongly recommended that zero preamble length never be used in CSMA/CD mode. If the preamble contains two consecutive 0s, the preamble is considered invalid. If the C152 detects an invalid preamble, the frame is ignored.

BOF - In CSMA/CD the Beginning-Of-Frame is a part of the preamble and consists of two sequential 1s. The purpose of the BOF is to identify the end of the preamble and indicate to the receiver(s) that the address will immediately follow.

ADDRESS - The address field is used to identify which messages are intended for which stations. The user must assign addresses to each destination and source. How the addresses are assigned, how they are maintained, and how each transmitter is made aware of which addresses are available is an issue that is left to the user. Some suggestions are discussed in Section 3.5.5. Generally, each address is unique to each station but there are special cases where this is not true. In these special cases, a message is intended for more than one station. These multi-targeted messages are called broadcast or multicast-group addresses. A broadcast address consisting of all 1s will always be received by all stations. A multicast-group address usually is indicated by using a 1as the first address bit. The user can choose to mask off all or selective bits of the address so that the GSC receives all messages or multicast-group messages. The address length is programmable to be 8 or 16 bits. An address consisting of all 1s will always be received by the GSC on the C152. The address bits are always passed from the GSC to the CPU. With user software, the address can be extended beyond 16 bits, but the automatic address recognition will only work on a maximum of 16 bits. User software will have to resolve any remaining address bits.

INFO - This is the information field and contains the data that one device on the link wishes to transmit to another device. It can be of any length the user wishes but needs to be in multiples of 8 bits. This is because multiples of 8 bits are used to transfer data into or out of the GSC FIFOs. The information field is delineated from the rest of the components of the frame by the preceding address field and the following CRC. The receiver determines the position of the end of the information field by passing the bytes through a temporary storage space. When the EOF is received the bytes in temporary storage are the CRC, and the last bit received previous to the CRC constitute the end of the information field.

CRC - The Cyclic Redundancy Check (CRC) is an error checking algorithm commonly used in serial communications. The C152 offers two types of CRC algorithms, a 16-bit and a 32-bit. The 16-bit algorithm is normally used in the SDLC mode and will be described in the SDLC section. In CSMA/CD applications either algorithm can be used but IEEE 802.3 uses a 32-bit CRC. The generation polynomial the C152 uses with the 32-bit CRC is:

$$G(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

The CRC generator, as shown in Figure 3.2, operates by taking each bit as it is received and XOR'ing it with bit 31 of the current CRC. This result is then placed in temporary storage. The result of XOR'ing bit 31 with the received bit is then XOR'd with bits 0, 1, 3, 4, 6, 7, 9, 10, 11, 15, 21, 22, 25 as the CRC is shifted right one position. When the CRC is shifted right, the temporary storage space holding the result of XOR'ing bit 31 and the incoming bit is shifted into position 0. The whole process is then repeated with the next incoming or outgoing bit.

The user has no access to the CRC generator or the bits which constitute the CRC while in CSMA/CD. On transmission, the CRC is automatically appended to the data being sent, and on reception, the CRC bits are not normally loaded into the receive FIFO. Instead, they are automatically stripped. The only indication the user has for the status of the CRC is a pass/fail flag. The pass/fail flag only operates during reception. A CRC is considered as passing when the the CRC generator has 11000111 00000100 11011010 01111011B as a remainder after all of the data, including the CRC checksum, from the transmitting station has been cycled through the CRC generator. The preamble, BOF and EOF are not included as part of the CRC algorithm. An interrupt is available that will interrupt the CPU if the CRC of the receiver is invalid. The user can enable the CRC to be passed to the CPU by placing the receiver in the raw receive mode.

This method of calculating the CRC is compatible with IEEE 802.3.

EOF - The End Of Frame indicates when the transmission is completed. The end flag in CSMA/CD consists of an idle condition. An idle condition is assumed when there is no transitions and the link remains high for 2 or more bit times.

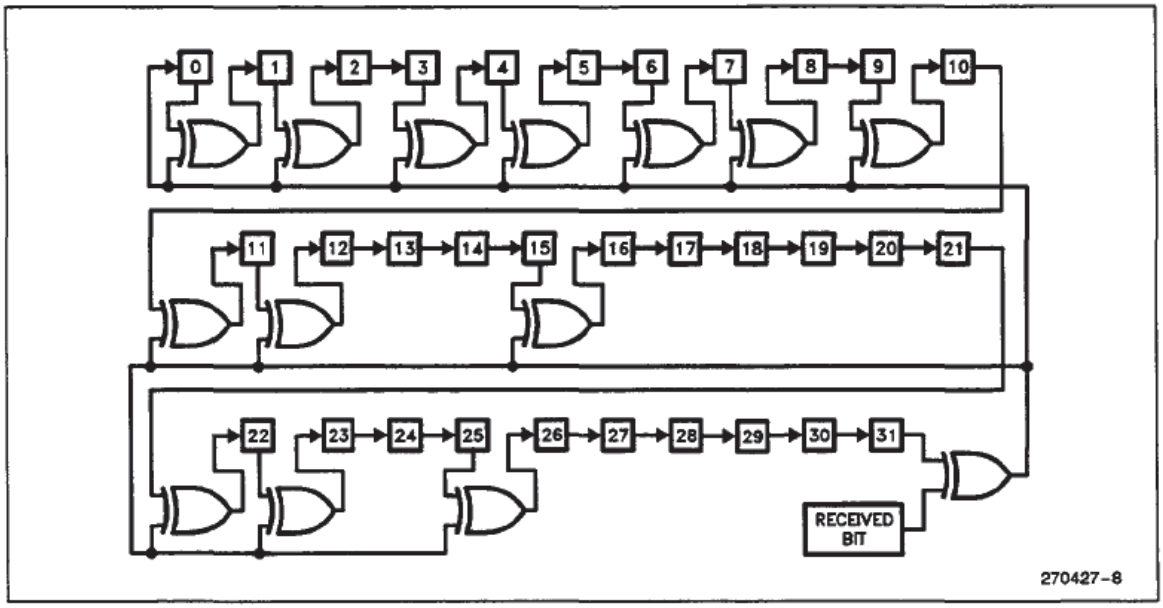



**Figure 3.2. CRC Generator**

### 3.2.3 INTERFRAME SPACE

The interframe space is the amount of time that transmission is delayed after the link is sensed as being idle and is used to separate transmitted frames. In alternate backoff mode, the interframe space may also be included in the determination of when retransmissions may actually begin. The C152 allows programmable interframe spaces of even numbers of bit times from 2 to 256. The hardware enforces the interframe space in SDLC mode as well as in CSMA/CD mode.

The period of the interframe space is determined by the contents of IFS. IFS is an SFR that is programmable from 0 to 254. The interframe space is measured in bit times. The value in IFS multiplied by the bit time equals the interframe space unless IFS equals 0. If IFS does equal 0, then the interframe space will equal 256 bit times. One of the considerations when loading the IFS is that only even numbers (LSB must be 0) can be used because only the 7 most significant bits are loaded into IFS. The LSB is controlled by the GSC and determines which half of the IFS is currently being used. In some modes, the interframe space timer is re-triggered if activity is detected during the first half of the period. The GSC determines which half of the interframe space is currently being used by examining the LSB. A one indicates the first half and zero indicates the second half of the IFS.

After reset IFS is 0, which delays the first transmission for both SDLC and CSMA/CD by 256 bit times (after reset, a bit time equals 8 oscillator clock periods).

In most applications, the period of the interframe space will be equal to or greater than the amount of time needed to turn-around the received frame. The turn-around period is the amount of time that is needed by user software to complete the handling of a received frame and be prepared to receive the next frame. An interframe space smaller than the required turn-around period could be used, but would allow some frames to be missed.

When a GSC transmitter has a new message to send, it will first sense the link. If activity is detected, transmission will be deferred to allow the frame in progress to complete. When link activity ceases, the station continues deferring for one interframe space period.

As mentioned earlier, the interframe space is used during the collision resolution period as well as during normal transmission. The backoff method selected affects how the deference period is handled during normal transmission. If normal backoff mode is selected, the interframe space timer is reset if activity occurs during approximately the first half of the interframe space. If alternate backoff or deterministic backoff is selected, the timer is not reset. In all cases when the interframe space timer expires, transmission may begin, regardless if there is activity on the link or not. Although the C152 resets the interframe space timer if activity is detected during the first one-half of the interframe space, this is not necessarily true of all CSMA/CD systems. (IEEE 802.3 recommends that the interframe space be reset if activity is detected during the first two-thirds or less of the interframe space.)

### 3.2.4 CSMA/CD DATA ENCODING

Manchester encoding/decoding is automatically selected when the user software selects CSMA/CD transmission mode (See Figure 3.3). In Manchester encoding the value of the bit is determined by the transition in the middle of the bit time, a positive transition is decoded as a 1 and a negative transition is decoded as a 0. The Address and Info bytes are transmitted LSB first. The CRC is transmitted MSB first.

If the external 1X clock feature is chosen the transmission mode is always NRZ (see Section 3.5.11). Using CSMA/CD with the external clock option is not supported because the data needs reformatting from NRZ to Manchester for the receiver to be able to detect code violations and collisions.

### 3.2.5 COLLISION DETECTION

The GSC hardware detects collisions by detecting Manchester waveform violations at its GRXD pin. Three kinds of waveform violations are detected: a missing 0-to-1 transition where one was expected, a 1-to-0 transition where none was expected, and a waveform that stays low (or high) for too short a time.

#### Jitter Tolerance

A valid Manchester waveform must have a transition at the midpoint of any bit cell, and may have a transition at the edge of any bit cell. Therefore, transitions will nominally be separated by either 1/2 bit-time or 1 bit-time.

The GSC samples the GRXD pin at the rate of 8 x the bit rate. The sequence of samples for the received bit sequence 001 would nominally be:

```
samples:11110000:11110000:00001111:
bit value:      0     :      0     :      1     :
        :<-bit cell->:<-bit cell->:<-bit cell->:
```

The sampling system allows a jitter tolerance of ±1 sample for transitions that are 1/2 bit-time apart, and ±2 samples for transitions that are 1 bit-time apart.

#### Narrow Pulses

A valid Manchester waveform must stay high or low for at least a half bit-time, nominally 4 sample-times. Jitter tolerance allows a waveform which stays high or low for 3 sample-times to also be considered valid. A sample sequence which shows a second transition only 1 or 2 sample-times after the previous transition is considered to be the result of a collision. Thus, sample sequences such as 0000110000 and 111101111 are interpreted as collisions.

The GSC hardware recognizes the collision to have occurred within 3/8 to 1/2 bit-time following the second transition.

#### Missing 0-to-1 Transition

A 0-to-1 transition is expected to occur at the center of any bit cell that begins with 0. If the previous 1-to-0 transition occurred at the bit cell edge, a jitter tolerance of ±1 sample is allowed. Sample sequences such as 1111:00001111 and 1111:000001111 are valid, where ":" indicates a bit cell edge. Sequences of the form 1111:000000XXX are interpreted as collisions.

For these kinds of sequences, the GSC recognizes the collision to have occurred within 1 to 1 1/8 bit-times after the previous 1-to-0 transition.

If the previous 1-to-0 transition occurred at the center of the previous bit cell, a jitter tolerance of ±2 samples is allowed. Thus, sample sequences such as 11110000:00001111 and 111100000:000001111 are valid. Sequences of the form 111100000:000000XXX are interpreted as collisions.

For these kinds of sequences, the GSC recognizes the collision to have occurred within 1 5/8 to 1 3/4 bit-times after the previous 1-to-0 transition.

#### Unexpected 1-to-0 Transition

If the line is at a logic 1 during the first half of a bit cell, then it is expected to make a 1-to-0 transition at the midpoint of the bit cell. If the transition is missed, it is assumed that this bit cell is the first half of an EOF flag

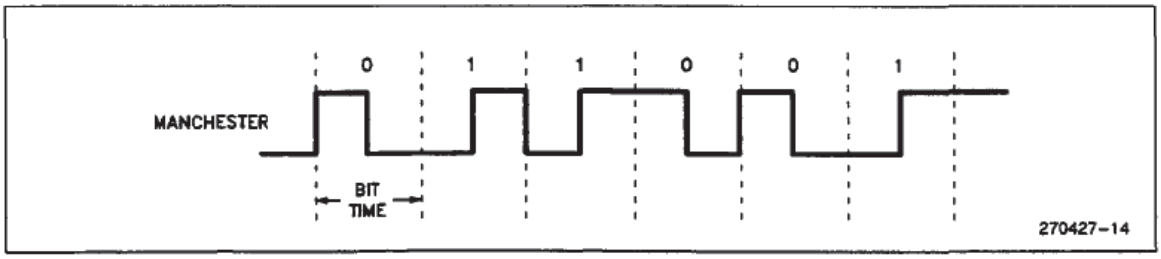



**Figure 3.3. Manchester Encoding**

(line idle for two bit-times). One bit-time later (which marks the midpoint of the next bit cell), if there is still no 1-to-0 transition, a valid EOF is assumed and the line idle bit (LNI in TSTAT) gets set.

However, if the assumed EOF flag is interrupted by a 1-to-0 transition in the bit-time following the first missing transition, a collision is assumed. In that case the GSC hardware recognizes the collision to have occurred within 1/2 to 5/8 bit-time after the unexpected transition.

### 3.2.6 RESOLUTION OF COLLISIONS

How the GSC responds to a detected collision depends on what it was doing at the time the collision was detected. What it might be doing is either transmitting or receiving a frame, or it might be inactive.

#### GSC Inactive

The collision is detected whether the GSC is active or not. If the GSC is neither transmitting nor receiving at the time the collision is detected, it takes no action unless user software has selected the Deterministic Collision Resolution (DCR) algorithm. If DCR has been selected, the GSC will participate in the resolution algorithm.

#### GSC Receiving

If the GSC is already in the process of receiving a frame at the time the collision is detected, its response depends on whether the first byte of the frame has been transferred into RFIFO yet or not. If that hasn't occurred, the GSC simply aborts the reception, but takes no other action unless DCR has been selected. If DCR has been selected, the GSC participates in the resolution algorithm.

If the reception has already progressed to the point where a byte has been transferred to RFIFO by the time the collision is detected, the receiver is disabled (GREN = 0), and the Receive Error Interrupt flag RCABT is set. If DCR has been selected, the GSC participates in the resolution algorithm.

Incoming bits take 1/2 bit time to get from the GRXD pin to the bit decoder. The bit decoder strips off the preamble/BOF bits, and the first bit after BOF is shifted into a serial strip buffer. The length of the strip buffer is equal to the number of bits in the selected CRC. It is within this buffer that address recognition takes place. If the address is recognized as one for which reception should proceed, then when the first address bit exits the strip buffer it is shifted into an 8-bit shift register. When the shift register is full, its content is transferred to RFIFO. That is the event that determines whether a collision sets RCABT or not.

#### GSC Transmitting

If the GSC is in the process of transmitting a frame at the time the collision is detected, it will in every case execute its jam/backoff procedure. Its reponse beyond that depends on whether the first byte of the frame has been transferred from TFIFO to the output shift register yet or not. That transfer takes place at the beginning of the first bit of the BOF; that is, 2 bit-times before the end of the preamble/BOF sequence.

If the transfer from TFIFO hasn't occurred yet, the GSC hardware will try again to gain access to the line after its backoff time has expired. Up to 8 automatic restarts can be attempted. If the 8th restart is interrupted by yet another collision, the transmitter is disabled (TEN = 0) and the Transmit Error Interrupt flag TCDT is set.

If the transfer from TFIFO occurs before a collision is detected, the transmitter is disabled (TEN = 0) and the TCDT flag is set.

The response of the GSC to detected collisions is summarized in Figure 3.4.

| What the GSC was doing | Response |
|---|---|
| nothing | None, unless DCR = 1.<br>If DCR = 1, begin DCR countdown. |
| Receiving a Frame, first byte not in RFIFO yet. | None, unless DCR = 1.<br>If DCR = 1, begin DCR countdown. |
| Receiving a Frame, first byte already in RFIFO. | Set RCABT, clear GREN.<br>If DCR = 1, begin DCR countdown. |
| Transmitting a Frame, first byte still in TFIFO | Execute jam/backoff.<br>Restart if collision count ≤8. |
| Transmitting a Frame, first byte already taken from TFIFO | Execute jam/backoff.<br>Set TCDT, clear TEN. |

**Figure 3-4. Response to a Detected Collision. References to DCR and the DCR Countdown Have to Do with the Deterministic Collision Resolution Algorithm.**

### Jam

The jam signal is generated by any 8XC152 that is involved in transmitting a frame at the time a collision is detected at its GRXD pin. This is to ensure that if one transmitting station detects a collision, all the other stations on the network will also detect a collision.

If a transmitting 8XC152 detects a collision during the preamble/BOF part of the frame that it is trying to transmit, it will complete the preamble/BOF and then begin the jam signal in the first bit time after BOF. If the collision is detected later in the frame, the jam signal will begin in the next bit time after the collision was detected.

The jam signal lasts for the same number of bit times as the selected CRC length—either 16- or 32-bit times.

The 8XC152 provides two types of jam signals that can be selected by user software. If the node is DC-coupled to the network, the DC jam can be selected. In this case the GTXD pin is pulled to a logic 0 for the duration of the jam. If the node is AC-coupled to the network, then AC jam must be selected. In this case the GSC takes the CRC it has calculated thus far in the transmission, inverts each bit, and transmits the inverted CRC. The selection of DC or AC jam is made by setting or clearing the DCJ bit, which resides in the SFR named MYSLOT.

When the jam signal is completed, the 8XC152 goes into an idle state. Presumeably, other stations on the network are also generating their own jam signals, after which they too go into an idle state. When the 8XC152 detects the idle state at its own GRXD pin, the backoff sequence begins.

### Backoff

There are three software selectable collision resolution algorithms in the 8XC152. The selection is made by writing values to 3 bits:

| DCR | M1 | M0 | Algorithm |
|---|---|---|---|
| 0 | 0 | 0 | Normal Random |
| 0 | 1 | 1 | Alternate Random |
| 1 | 1 | 1 | Deterministic |

M1 and M0 reside in GMOD, and DCR is in MYSLOT.

In the Normal Random algorithm, the GSC backs off for a random number of slot times and then decides whether to restart the transmission. The backoff time begins as soon as a line idle condition is detected.

The Alternate Random algorithm is the same as the Normal Random except the backoff time doesn't start until an IFS has transpired.

In the Deterministic algorithm, the GSC backs off to await its pre-determined turn.

### Random Backoff

In either of the random algorithms, the first thing that happens after a collision is detected is that a 1 gets shifted into the TCDCNT (Transmit Collision Detect Count) register, from the right.

Thus if the software cleared TCDCNT before telling the GSC to transmit, then TCDCNT keeps track of how many times the transmission had to be aborted because of collisions:

| | | |
|---|---|---|
| TCDCNT = | 00000000 | first attempt |
| | 00000001 | first collision |
| | 00000011 | second collision |
| | 00000111 | third collision |
| | 00001111 | fourth collision |
| | . . . . . | |
| | 11111111 | eighth collision |

After TCDCNT gets a 1 shifted into it, the logical AND of TCDCNT and PRBS is loaded into a countdown timer named BKOFF. PRBS is the name of an SFR which contains the output of a pseudo-random binary sequence generator. Its function is to provide a random number for use in the backoff algorithm.

Thus on the first collision BKOFF gets loaded randomly with either 00000000 or 00000001. If there is a second collision it gets loaded with the random selection of 00000000, 00000001, 00000010, or 00000011. On the third collision there will be a random selection among 8 possible numbers. On the fourth, among 16, etc. Figure 3.5 shows the logical arrangement of PRBS, TCDCNT, and BKOFF.

BKOFF starts counting down from its preload value, counting slot times. At any time, the current value in BKOFF can be read by the CPU, but CPU writes to BKOFF have no effect. While BKOFF is counting down, if its current value is not 0, transmission is disabled. The output signal "BKOFF = 0" is asserted when BKOFF reaches 0, and is used to re-enable transmission.

At that time transmission can proceed, subject of course to IFS enforcement, unless:

- shifting a 1 into TCDCNT from the right caused a 1 to shift out from the MSB of TCDCNT, or
- the collision was detected after TFIFO had been accessed by the transmit hardware.

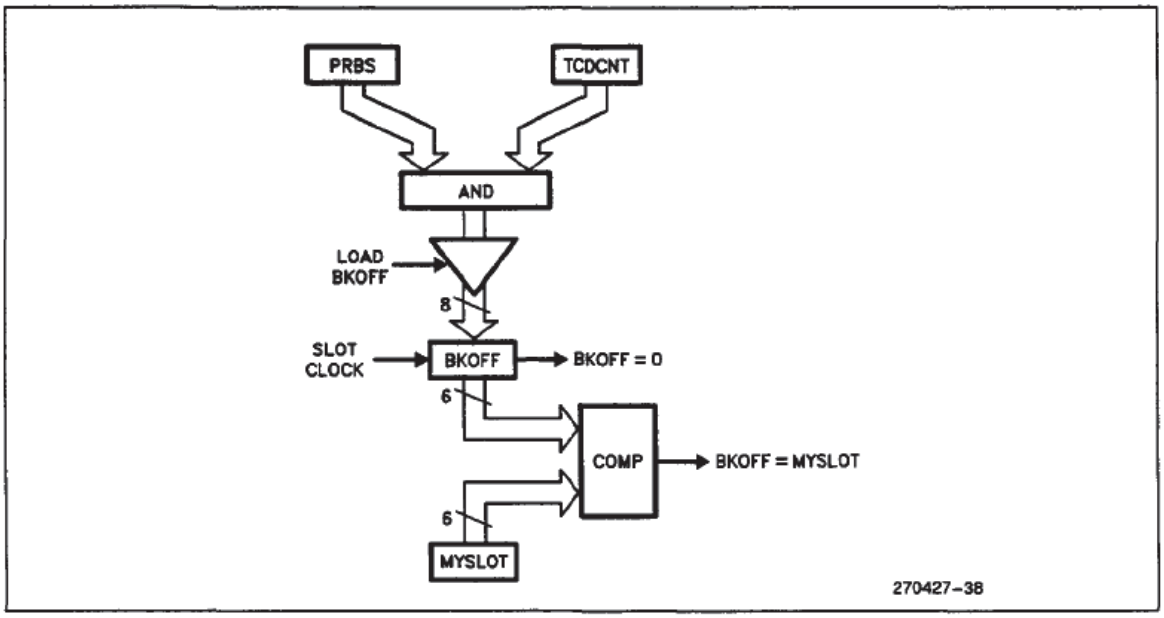



**Figure 3.5. Backoff Timer Logic**

In either of these cases, the transmitter is disabled (TEN = 0) and the Transmit Error flag TCDT is set. The automatic restart is canceled.

Where the Normal and Alternate Random backoff algorithms differ is that in Normal Random backoff the BKOFF timer starts counting down as soon as a line idle condition is detected, whereas in Alternate Random backoff the BKOFF timer doesn't start counting down till the IFS expires.

The Alternate Random mode was designed for networks in which the slot time is less than the IFS. If the randomly assigned backoff time for a given transmitter happens to be 0, then it is free to transmit as soon as the IFS ends. If the slot time is shorter than the IFS, Normal Random mode would nearly guarantee that if there's a first collision there will be a second collision. The situation is avoided in Alternate Random mode, since the BKOFF countdown doesn't start till the IFS is over.

The unit of count to the BKOFF timer is the slot time. The slot time is measured in bit-times, and is determined by a CPU write to the register SLOTTM. The slot time clock is a 1-byte downcounter which starts its countdown from the value written to SLOTTM. It is decremented each bit time when a backoff is in progress, and when it gets to 1 it generates one tick in the slot time clock. The next state after 1 is the reload value which was written to SLOTTM. If 0 is the value written to SLOTTM, the slot time clock will equal 256 bit times.

A CPU write to SLOTTM accesses the reload register. A CPU read of SLOTTM accesses the downcounter. In most protocols, the slot period must be equal to or greater than the longest round trip propagation time plus the jam time.

### Deterministic Backoff

In the Deterministic backoff mode, the GSC is assigned (in software) a slot number. The slot assignment is written to the low 6 bits of the register MYSLOT. This same register also contains, in the 2 high bit positions, the control bits DCJ and DCR.

Slot assignments therefore can run from 0 to 63. It will turn out that the higher the slot assignment, the sooner the GSC will get to restart its transmission in the event of a collision.

The highest slot assignment in the network is written by each station's software into its TCDCNT register. Normally the highest slot assignment is just the total number of stations that are going to participate in the backoff algorithm.

In deterministic backoff mode a collision will not cause a 1 to be shifted into TCDCNT. TCDCNT will still be ANDed with PRBS and the result loaded into BKOFF. In order to insure that all stations have the same value loaded into BKOFF, which determines the first slot number to occur, the PRBS should be loaded with 0FFH; the PRBS will maintain this value until either the 8XC152 is reset or the user writes some other value into PRBS. After BKOFF is loaded it begins counting down slot times as soon as the IFS ends. Slot times are defined by the user, the same way as before, by loading SLOTTM with the number of bit times per slot.

When BKOFF equals the slot assignment (as defined in MYSLOT), the signal "BKOFF = MYSLOT" in Figure 3.5 is asserted for one slot time, during which the GSC can restart its transmission.

While BKOFF is counting down, if any activity is detected at the GRXD pin, the countdown is frozen until the activity ends, a line idle condition is detected, and an IFS transpires. Then the countdown resumes from where it left off.

If a collision is detected at the GRXD pin while BKOFF is counting down, the collision resolution algorithm is restarted from the beginning.

In effect, the GSC "owns" its assigned slot number, but with one exception. Nobody owns slot number 0. Therefore if the GSC is assigned slot number 0, then when BKOFF = 0, this station and any other station that has something to say at this time will have an equal chance to take the line.

### 3.2.7 HARDWARE BASED ACKNOWLEDGE

Hardware Based Acknowledge (HBA) is a data link packet acknowledging scheme that the user software can enable with CSMA/CD protocol. It is not an option with SDLC protocol however.

In general HBA can give improved system response time and increased effective transmission rates over acknowledge schemes implemented in higher layers of the network architecture. Another benefit is the possibility of early release of the transmit buffer as soon as the acknowledge is received.

The acknowledge consists of a preamble followed by an idle condition. A receiving station with HABEN enabled will send an acknowledge only if the incoming address is unique to the receiving station and if the frame is determined to be correct with no errors. For the acknowledge to be sent, TEN must be set. For the transmitting station to recognize the acknowledge GREN must be set. A zero as the LSB of the address indicates that the address is unique and not a group or broadcast address. Errors can be caused by collisions, incorrect CRC, misalignment, or FIFO overflow. The receiver sends the acknowledge as soon as the line is sensed to be idle. The user must program the interframe space and the preamble length such that the acknowledge is completed before IFS expires. This is normally done by programming IFS larger than the preamble.

A transmitting station with HABEN enabled expects an acknowledge. It must receive one prior to the end of the interframe space, or else an error is assumed and the NOACK bit is set. Setting of the TDN bit is also delayed until the end of the interframe space. Collisions detected during the interframe space will also cause NOACK to be set.

If the user software has enabled DMA servicing of the GSC, an interrupt is generated when TDN is set. TDN will be set at the end of the interframe space if a hardware based acknowledge is required and received. If the GSC is serviced by the CPU, the user must time out the interframe space and then check TDN before disabling the transmitter or transmit error interrupts. NOACK will generate a transmit error interrupt if the transmitter and interrupts are enabled during the interframe space.

## 3.3 SDLC Operation

### 3.3.1 SDLC OVERVIEW

SDLC is a communication protocol developed by IBM and widely used in industry. It is based on a primary/secondary architecture and requires that each secondary station have a unique address. The secondary stations can only communicate to the primary station, and then, only when the primary station allows communication to take place. This eliminates the possibility of contention on the serial line caused by the secondary station's trying to transmit simultaneously.

In the C152, SDLC can be configured to work in either full or half duplex. When adhering to strict SDLC protocol, full duplex is required. Full duplex is selected whenever a 16-bit CRC is selected. At the end of a valid reset the 16-bit CRC is selected. To select half duplex with a 16-bit CRC, the receiver must be turned off by user software before transmission. The receiver is turned off by clearing the GREN bit (RSTAT.1). The receiver needs to be turned off because the address that is transmitted is the address of the secondary station's receiver. If not turned off, the receiver could mistake the outgoing message as being intended for itself. When 32-bit CRCs are used, half duplex is the only method available for transmission.

### 3.3.2 SDLC Frame Format

The format of an SDLC frame is shown in Figure 3.6. The frame consists of a Beginning of Frame flag, Address field, Control Field, Information field (optional), a CRC, and the End of Frame flag.


**Figure 3.6. Typical SDLC Frame**

BOF - The begin of frame flag for SDLC is 01111110. It is only one of two possible combinations that have six consecutive ones in SDLC. The other possibility is an abort character which consists of eight or more consecutive ones. This is because SDLC utilizes a process called bit stuffing. Bit stuffing is the insertion of a 0 as the next bit every time a sequence of five consecutive 1s is detected. The receiver automatically removes a 0 after every consecutive group of five ones. This removal of the 0 bit is referred to as bit stripping. Bit stuffing is discussed in Section 3.3.4. All the procedures required for bit stuffing and bit stripping are automatically handled by the GSC.

In standard SDLC protocol the BOF signals the start of a frame and is limited to 8 bits in length. Since there is no preamble in SDLC the BOF is considered an entire separate field and marks the beginning of the frame. The BOF also serves as the clock synchronization mechanism and the reference point for determining the position of the address and control fields.

ADDRESS - The address field is used to identify which stations the message is intended for. Each secondary station must have a unique address. The primary station must then be made aware of which addresses are assigned to each station. The address length is specified as 8-bits in standard SDLC protocols but it is expandable to 16-bits in the C152. User software can further expand the number of address bits, but the automatic address recognition feature works on a maximum of 16-bits.

In SDLC the addresses are normally unique for each station. However, there are several classes of messages that are intended for more than one station. These messages are called broadcast and group addressed frames. An address consisting of all 1s will always be automatically received by the GSC, this is defined as the broadcast address in SDLC. A group address is an address that is common to more than one station. The GSC provides address masking bits to provide the capability of receiving group addresses.

If desired, the user software can mask off all the bits of the address. This type of masking puts the GSC in a promiscuous mode so that all addresses are received.

CONTROL - The control field is used for initialization of the system, identifying the sequence of a frame, to identify if the message is complete, to tell secondary stations if a response is expected, and acknowledgement of previously sent frames. The user software is responsible for insertion of the control field as the GSC hardware has no provisions for the management of this field. The interpretation and formation of the control field must also be handled by user software. The information following the control field is typically used for information transfer, error reporting, and various other functions. These functions are accomplished by the format of the control field. There are three formats available. The types of formats are Informational, Supervisory, or Unnumbered. Figure 3.7 shows the various format types and how to identify them.

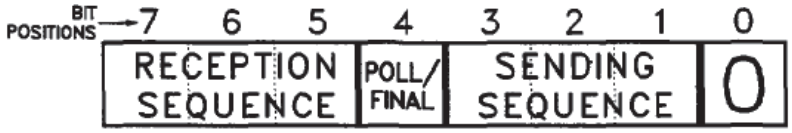
Since the user software is responsible for the implementation of the control field, what follows is a simple explanation on the control field and its functions. For a complete understanding and proper implementation of SDLC, the user should refer to the IBM document, GA27-3093-2, IBM Synchronous Data Link Control General Information. Within that document, is another list of IBM documents which go into detail on the SDLC protocol and its use.

The control field is eight bits wide and the format is determined by bits 0 and 1. If bit 0 is a zero, then the frame is an informational frame. If bit 0 is a one and bit 1 a zero, then it is a supervisory frame, and if bit 0 is a one and bit 1 a one then the frame is an unnumbered frame.

In an informational frame bits 3,2,1 contain the sequence count of the frame being sent.

Bit 4 is the P/F (Poll/Final) bit. If bit 4 equals 1 and originates from the primary, then the secondary station is expected to initiate a transmission. If bit 4 equals 1 and originates from a secondary station, then the frame is the final frame in a transmission.

Bits 7,6,5 contain the sequence count a station expects on the next transmission to it. The sequence count can vary from 000B to 111B. The count then starts over again at 000B after the value 111B is incremented. The acknowledgement is recognized by the receiving station when it decodes bits 7,6,5 of an incoming frame. The station sending the transmission is acknowledging the frames received up to the count represented in bits 7,6,5 (sequence count-1). With this method, up to seven sequential frames may be transmitted prior to an acknowledgement being received. If eight frames were allowed to pass before an acknowledgement, the sequence count would roll over and this would negate the purpose of the sequence numbers.




270427-15

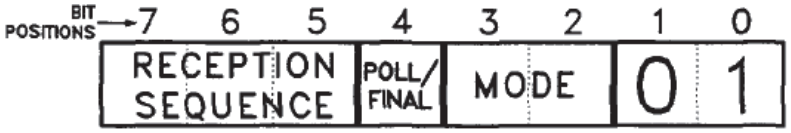
RECEPTION SEQUENCE - The sequence expected in the SENDING SEQUENCE portion of the control byte in the next received frame. This also confirms correct reception of up to seven frames prior to the sequence given.

POLL/FINAL - Identifies the frame as being a polling request from the master station or the last in a series of frames from the master or secondary.

SENDING SEQUENCE - Identifies the sequence of the frame being transmitted.

0 - If bit 0 = 0 the frame is identified as a informational format type.

## INFORMATION FORMAT




270427-16

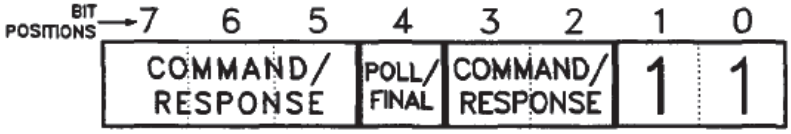
RECEPTION SEQUENCE - Expected sequence of frame for next reception.

POLL/FINAL - Identifies frame as being a polling request from the master station or the last in a series of frames from the master or secondary.

MODE - Identifies whether receiver is ready (00), not ready (10) or a frame was rejected (01). The rejected frame is identified by the reception sequence.

0,1 - If bits 1,0 = 0,1 the frame is identified as a supervisory format type.

## SUPERVISORY FORMAT




270427-17

COMMAND/RESPONSE - Identifies the type of command or response.

POLL/FINAL - Identifies frame as being a polling request from the master station or the last in a series of frames from the master or secondary.

1,1 - If bits 1,0 = 1,1 the frame is identified as an unnumbered format type.

## NONSEQUENCED FORMAT

270427-18

**Figure 3.7. SDLC Control Field**

Following the informational control field comes the information to be transferred.

In the supervisory format (bits 1,0 = 0,1) bits 3,2 determine which mode is being used.

When the mode is 00 it indicates that the receive line of the station that sent the supervisory frame is enabled and ready to accept frames.

When the mode is 01, it indicates that previously a received frame was rejected. The value in the receive count identifies which frame(s) need to be retransmitted.

When the mode is 10, the sending station is indicating that its receiver is not ready to accept frames.

Mode 11 is an illegal mode in SDLC protocol.

Bits 7,6,5 represent the value of the sequence the station expects when the next transfer occurs for that station. There is no information following the control field when the supervisory format is used.

In the unnumbered format (bits 1,0 = 1,1) bits 7, 6, 5, 3, 2 (notice bit 4 is missing) indicate commands from the primary to secondary stations or requests of secondary stations to the primary.

The standard commands are:

| BITS | 7 | 6 | 5 | 3 | 2 | Command |
|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | Unnumbered Information (UI) |
| | 0 | 0 | 0 | 0 | 1 | Set initialization mode (SIM) |
| | 0 | 1 | 0 | 0 | 0 | Disconnect (DISC) |
| | 0 | 0 | 1 | 0 | 0 | Response optional (UP) |
| | 1 | 1 | 0 | 0 | 1 | Function descriptor in information field (CFGR) |
| | 1 | 0 | 1 | 1 | 1 | Identification in information field. (XID) |
| | 1 | 1 | 1 | 0 | 0 | Test pattern in information field. (TEST) |

The standard responses are:

| BITS | 7 | 6 | 5 | 3 | 2 | Command |
|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | Unnumbered information (UI) |
| | 0 | 0 | 0 | 0 | 1 | Request for initialization (RIM) |
| | 0 | 0 | 0 | 1 | 1 | Station in disconnected mode (DM) |
| | 1 | 0 | 0 | 0 | 1 | Invalid frame received (FRMR) |
| | 0 | 1 | 1 | 0 | 0 | Unnumbered acknowledgement (UA) |
| | 1 | 1 | 1 | 1 | 1 | Signal loss of input (BCN) |
| | 1 | 1 | 0 | 0 | 1 | Function descriptor in information field (CFGR) |
| | 0 | 1 | 0 | 0 | 0 | Station wants to disconnect (RD) |
| | 1 | 0 | 1 | 1 | 1 | Identification in information field (XID) |
| | 1 | 1 | 1 | 0 | 0 | Test pattern in information field (TEST) |

In an unnumbered frame, information of variable length may follow the control field if UI is used, or information of fixed length may follow if FRMR is used.

As stated earlier, the user software is responsible for the proper management of the control field. This portion of the frame is passed to or from the GSC FIFOs as basic informational type data.

INFO - This is the information field and contains the data that one device on the link wishes to transmit to another device. It can be of any length the user wishes, but must be a multiple of 8 bits. It is possible that some frames may contain no information field. The information field is identified to the receiving stations by the preceding control field and the following CRC. The GSC determines where the last of the information field is by passing the bits through the CRC generator. When the last bit or EOF is received the bits that remain constitute the CRC.

CRC - The Cyclic Redundancy Check (CRC) is an error checking sequence commonly used in serial communications. The C152 offers two types of CRC algorithms, a 16-bit and a 32-bit. The 32-bit algorithm is normally used in CSMA/CD applications and is described in section 3.2.2. In most SDLC applications a 16-bit CRC is used and the hardware configuration that supports 16-bit CRC is shown in Figure 3.8. The generating polynomial that the CRC generator uses with the 16-bit CRC is:

$$G(X) = X^{16} + X^{12} + X^5 + 1$$

The way the CRC operates is that as a bit is received it is XOR'd with bit 15 of the current CRC and placed in temporary storage. The result of XOR'ing bit 15 with the received bit is then XOR'd with bit 4 and bit 11 as the CRC is shifted one position to the right. The bit in temporary storage is shifted into position 0.

The required CRC length for SDLC is 16 bits. The CRC is automatically stripped from the frame and not passed on to the CPU. The last 16 bits are then run though the CRC generator to insure that the correct remainder is left. The remainder that is checked for is 0011101000001111B (1D0F Hex). If there is a mismatch, an error is generated. The user software has the option of enabling this interrupt so the CPU is notified.

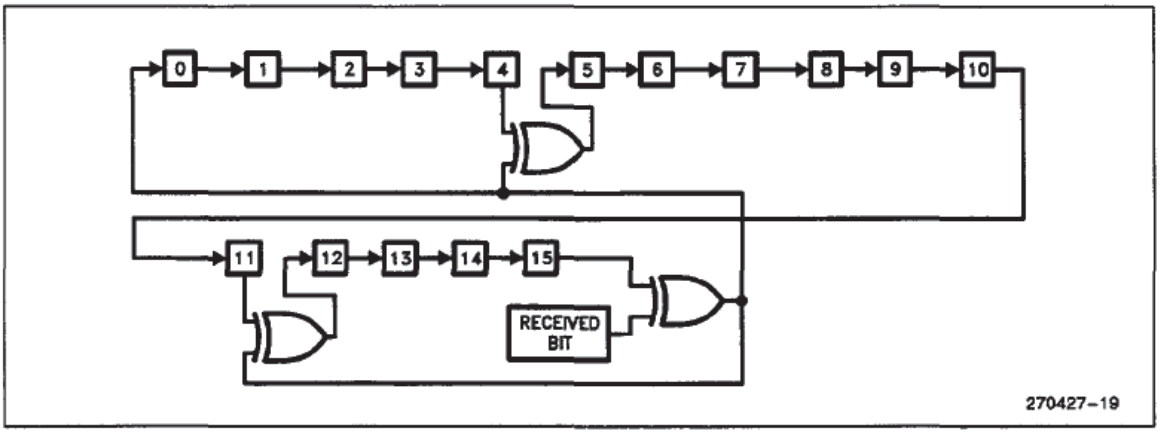



**Figure 3.8. 16-Bit CRC**

EOF - The End Of Frame (EOF) indicates when the transmission is complete. The EOF is identified by the end flag. An end flag consists of the bit pattern 01111110. The EOF can also serve as the BOF for the next frame.

### 3.3.3 DATA ENCODING

The transmission of data in SDLC mode is done via NRZI encoding as shown in Figure 3.9. NRZI encoding transmits data by changing the state of the output whenever a 0 is being transmitted. Whenever a 1 is transmitted the state of the output remains the same as the previous bit and remains valid for the entire bit time. When SDLC mode is selected it automatically enables the NRZI encoding on the transmit line and NRZI decoding on the receive line. The Address and Info bytes are transmitted LSB first. The CRC is transmitted MSB first.

### 3.3.4 BIT STUFFING/STRIPPING

In SDLC mode one of the primary rules of the protocol is that in any normal data transmission, there will never be an occurrence of more than 5 consecutive 1s. The GSC takes care of this housekeeping chore by automatically inserting a 0 after every occurrence of 5 consecutive 1s and the receiver automatically removes a zero after receiving 5 consecutive 1s. All the necessary steps required for implementing bit stuffing and stripping are incorporated into the GSC hardware. This makes the operation transparent to the user. About the only time this operation becomes apparent to the user, is if the actual data on the transmission medium is being monitored by a device that is not aware of the automatic insertion of 0s. The bit stuffing/stripping guarantees that there will be at least one transition every 6 bit times while the line is active.

### 3.3.5 SENDING ABORT CHARACTER

An abort character is one of the exceptions to the rule that disallows more than 5 consecutive 1s. The abort character consists of any occurrence of seven or more consecutive ones. The simplest way for the C152 to send an abort character is to clear the TEN bit. This causes the output to be disabled which, in turn, forces it to a constant high state. The delay necessary to insure that the link is high for seven bit times is a task that needs to be handled by user software. Other methods of sending an abort character are using the IFS register or using the Raw Transmit mode. Using IFS still entails clearing the TEN bit, but TEN can be immediately re-enabled. The next message will not begin until the IFS expires. The IFS begins timing out as soon as $\overline{DEN}$ goes high which identifies the end of transmission. This also requires that IFS contain a value equal to or greater than 8. This method may have the undesirable effect that $\overline{DEN}$ goes high and disables the external drivers. The other alternative is to switch to Raw Transmit mode. Then, writing 0FFH to TFIFO would generate a high output for 8 bit times. This method would leave $\overline{DEN}$ active during the transmission of the abort character.

When the receiver detects seven or more consecutive 1s and data has been loaded into the receive FIFO, the RCABT flag is set in RSTAT and that frame is ignored. If no data has been loaded into the receive FIFO, there are no abort flags set and that frame is just ignored. A retransmitted frame may immediately follow an abort character, provided the proper flags are used.

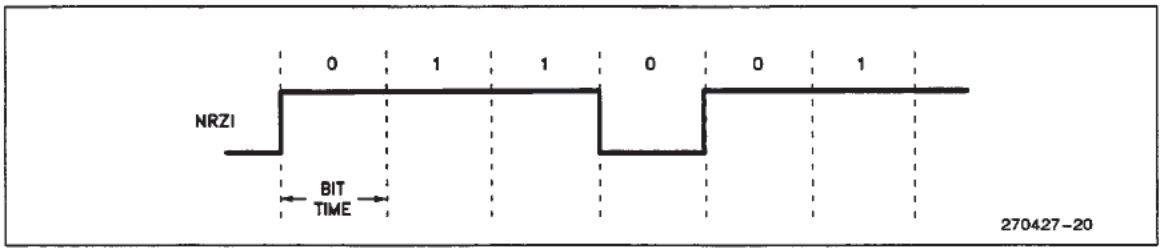



**Figure 3.9. NRZI Encoding**

### 3.3.6 LINE IDLE

If 15 or more consecutive 1s are detected by the receiver the Line Idle bit (LNI) in TSTAT is set. The seven 1s from the abort character may be included when sensing for a line idle condition. The same methods used for sending the Abort character can be used for creating the Idle condition. However, the values would need to be changed to reflect 15 bit times, instead of seven bit times.

### 3.3.7 ACKNOWLEDGEMENT

Acknowledgment in SDLC is an implied acknowledge and is contained in the control field. Part of the control frame is the sequence number of the next expected frame. This sequence number is called the Receive Count. In transmitting the Receive Count, the receiver is in fact acknowledging all the previous frames prior to the count that was transmitted. This allows for the transmission of up to seven frames before an acknowledge is required back to the transmitter. The limitation of seven frames is necessary because the Receive Count in the control field is limited to three binary digits. This means that if an eighth transmission occurred this would cause the next Receive Count to repeat the first count that still is waiting for an acknowledge. This would defeat the purpose of the acknowledgement. The processing and general maintenance of the sequence count must be done by the user software. The Hardware Based Acknowledge option that is provided in the C152 is not compatible with standard SDLC protocol.

### 3.3.8 PRIMARY/SECONDARY STATIONS

All SDLC networks are based upon a primary/secondary station relationship. There can be only one primary station in a network and all the other stations are considered secondary. All communication is between the primary and secondary station. Secondary station to secondary station direct communication is prohibited. If there is a need for secondary to secondary communication, the user software will have to make allowances for the master to act as an intermediary. Secondary stations are allowed use of the serial line only when the master permits them. This is done by the master polling the secondary stations to see if they have a need to access the serial line. This should prevent any collisions from occurring, provided each secondary station has its own unique address. This arrangement also partially determines the types of networks supported. Normal SDLC networks consist of point-to-point, multi-drop, or ring configurations and the C152 supports all of these. However, some SDLC processors support an automatic one bit delay at each node that is not supported by the C152. In a "Loop Mode" configuration, is is necessary that the transmission be delayed from the reception of the frames from the upstream station before passing the message to the downstream station. This delay is necessary so that a station can decode its own address before the message is passed on. The various networks are shown in Figure 3.10.

### 3.3.9 HDLC/SDLC COMPARISON

HDLC (High level Data Link Control) is a standard adopted by the International Standards Organization (ISO). The HDLC standard is defined in the ISO document #ISO 6159 - HDLC unbalanced classes of procedures. IBM developed the SDLC protocol as a subset of HDLC. SDLC conforms to HDLC protocol requirements, but is more restrictive. SDLC contains a more precise definition on the modes of operation.

Some of the major differences between SDLC and HDLC are:

| SDLC | HDLC |
|---|---|
| Unbalanced (primary/ secondary) | Balanced (peer to peer) |
| Modulo 8 (no extensions allowed, up to 7 outstanding frames before acknowledge is required) | Modulo 128 (up to 127 outstanding frames before acknowledge is required) |
| 8-bit addressing only | Extended addressing |
| Byte aligned data | Variable size of data |

The C152 does not support HDLC implementation requiring data alignment other than byte alignment. The user will find that many of the protocol parameters are programmable in the C152 which allows easy implementation of proprietary or standard HDLC network. User software needs to implement the control field functions.

### 3.3.10 USING A PREAMBLE IN SDLC

When transmitting a preamble in SDLC mode, the user should be aware that the pattern of 10101010 . . . is output. NRZI encoding is used in SDLC when the internal baud rate generator is the clock source and this means that a transition will occur every two bit times, when a 0 is transmitted. This compares with some other SDLC devices, most of which transmit the pattern 00000000 . . . which will cause a transition every bit time. Our past experience has shown that the C152 preamble does not cause a problem with most other devices. This is because the preamble is used only to define the relative bit time boundaries within some variation allowed by the receiving station, and the C152 preamble fulfills this function. The C152 does not have any problems with receiving a preamble consisting of all 0s. One note of caution however. If idle fill flags are used in conjunction with a preamble, the addresses 00(00)H and 55(55)H should not be assigned to any C152 as the preamble following the idle fill flags will be interpreted as an address.

## 3.4 User Defined Protocols

The explanation on the implementation of user defined protocols would go beyond the scope of this manual, but examining Table 3.1 should give the reader a consolidated list of most of the possibilities. In this manual, any deviation from the documents that cover the implementation of CSMA/CD or SDLC are considered user defined protocols. Examples of this would be the use of SDLC with the 32-bit CRC selected or CSMA/CD with hardware based acknowledge.

## 3.5 Using the GSC

### 3.5.1 LINE DISCIPLINE

Line discipline is how the management of the transfer of data over the physical medium is controlled. Two types of line discipline will be discussed in this section: full duplex and half duplex.

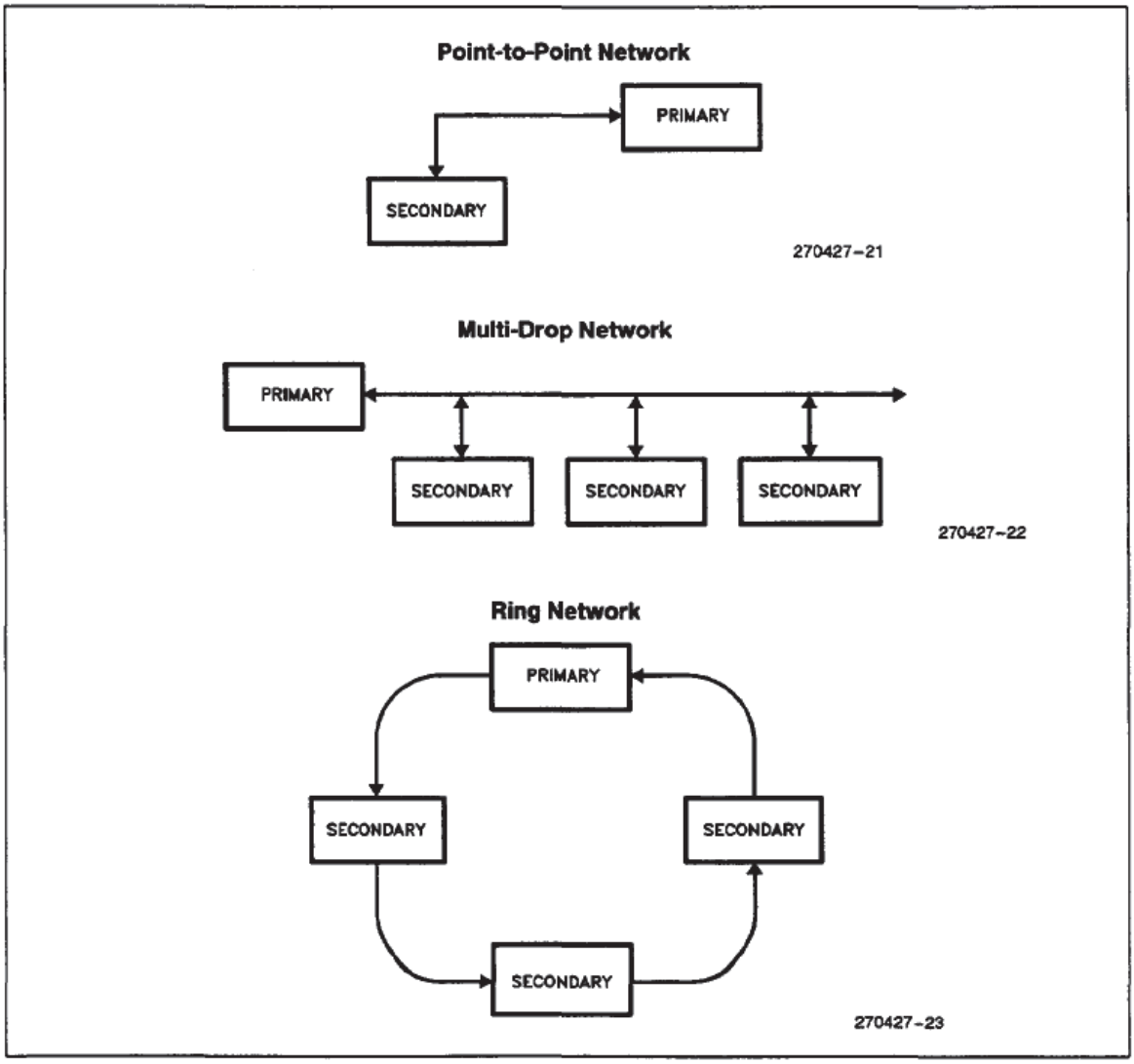



**Figure 3.10. SDLC Networks**

Full duplex is the simultaneous transmission and reception of data. Full duplex uses anywhere from two to four wires. At least one wire is needed for transmission and one wire for reception. Usually there will also be a ground reference on each signal if the distance from station to station is relatively long. Full-duplex operation in the C152 requires that both the receive and the transmit portion of the GSC are functioning at the same time. Since both the transmitter and receiver are operating, two CRC generators are also needed. The C152 handles this problem by having one 32-bit CRC generator and one 16-bit CRC generator. When supporting full-duplex operation, the 32-bit CRC generator is modified to work as a 16-bit CRC generator. Whenever the 16-bit CRC is selected, the GSC automatically enters the full duplex mode. Half duplex with a 16-bit CRC is discussed in the following paragraph.

Half duplex is the alternate transmission and reception of data over a single common wire. Only one or two wires are needed in half-duplex systems. One wire is needed for the signal and if the distance to be covered is long there will also be a wire for the ground reference. In half-duplex mode, only the receiver or transmitter can operate at one time. When the receiver or transmitter operates is determined by user software, but typically the receiver will always be enabled unless the GSC is transmitting. When using the C152 in half-duplex and the receiver is connected to the transmitter it is possible that a station will receive its' own transmission. This can occur if a broadcast address is sent, the address mask register(s) are filled with all 1s, or the address being sent matches the sending stations address through the use of the address masking registers. The receiver must be disabled by the user while transmitting if any of these conditions will occur, unless the user wants a station to receive its own transmission. The receiver is disabled by clearing GREN (and GAREN if used). Half-duplex operation in the C152 is supported with either 16-bit or 32-bit CRCs. Whenever a 32-bit CRC is selected, only half-duplex operation can be supported by the GSC. It is possible to simulate full-duplex operation with a 32-bit CRC, but this would require that the CRC be performed with software. Calculating the CRC with the CPU would greatly reduce the data rates that could be used with the GSC. Whenever a 16-bit CRC is selected, full-duplex operation is automatically chosen and the GSC must be reconfigured if half-duplex operation is preferred.

### 3.5.2 PLANNING FOR NETWORK CHANGES AND EXPANSIONS

A complete explanation on how to plan for network expansion will not be covered in this manual as there are far too many possibilities that would need to be discussed. But there are several areas that will have major impact when allowing for changes in the system. In cases where there will never be any changes allowed, expansion plans become a mute issue. However, it is strongly suggested that there always be some allowance for future modifications.

Some of the general areas that will impact the overall scheme on how to incorporate future changes to the system are:

1) Communication of the change to all the stations or the primary station.

2) Maximum distance for communication. This will affect the drivers used and the slot time.

3) More stations may be on the line at one time. This may impact the interframe space or the collision resolution used.

4) If using CSMA/CD without deterministic resolution, any increase in network size will have a negative impact on the average throughput of the network and lower the efficiency. The user will have to give careful consideration when deciding how large a system can ultimately be and still maintain adequate performance.

### 3.5.3 DMA SERVICING OF GSC CHANNELS

There are two sources that can be used to control the GSC. The first is CPU control and the second is DMA control.

CPU control is used when user software takes care of the tasks such as: loading the TFIFO, reading the RFIFO, checking the status flags, and general tracking of the transmission process. As the number of tasks grow and higher data transfer rates are used, the overhead required by the CPU becomes the dominant consumption of time. Eventually, a point is reached where the CPU is spending 100% of its time responding to the needs of the GSC. An alternative is to have the DMA channels control the GSC.

A detailed explanation on the general use of the DMA channels is covered in Section 4. In this section only those details required for the use of the DMA channels with the GSC will be covered.

The DMA channels can be configured by user software so that the GSC data transfers are serviced by the DMA controller. Since there are two DMA channels, one channel can be used to service the receiver, and one channel can be used to service the transmitter. In using the DMA channels, the CPU is relieved of much of the time required to do the basic servicing of the GSC buffers. The types of servicing that the DMA channels can provide are: loading of the transmit FIFO, removing data from the receive FIFO, notification of the CPU when the transmission or reception has ended, and response to certain error conditions. When using the

DMA channels the source or destination of the data intended for serial transmission can be internal data memory, external data memory, or any of the SFRs.

The only tasks required after initialization of the DMA and GSC registers are enabling the proper interrupts and informing the DMA controller when to start. After the DMA channels are started all that is required of the CPU is to respond to error conditions or wait until the end of transmission.

Initialization of the DMA channels requires setting up the control, source, and destination address registers. On the DMA channel servicing the receiver, the control register needs to be loaded as follows: DCONn.2 = 0, this sets the transfer mode so that response is to GSC interrupts and put the DMA control in alternate cycle mode; DCONn.3 = 1, this enables the demand mode; DCONn.4 = 0, this clears the automatic increment option for the source address; and DCONn.5 = 1, this defines the source as SFR. The DMA channel servicing the receiver also needs its source address register to contain the address of RFIFO (SARHN = XXH, SARLN = 0F4H). On the DMA channel servicing the transmitter, the control register needs to be loaded as follows: DCONn.2 = 0; DCONn.3 = 1; DCONn.6 = 0, this clears the automatic increment option for the destination address; and DCONn.7 = 1, this sets the destination as SFR. The DMA channel serving the transmitter also requires that its destination address register contains the address of TFIFO (DARHN = XXH, DARLN = 85H). Assuming that DCON0 would be serving the receiver and DCON1 the transmitter, DCON0 would be loaded with XX1010X0B and DCON1 would be loaded with 10XX10X0B. The contents of SARH0 and DARH1 do not have any impact when using internal SFRs as the source or destination.

When using the DMA channels to service the GSC, the byte count registers will also need to be initialized.

The Done flag for the DMA channel servicing the receiver should be used if fixed packet lengths only are being transmitted or to insure that memory is not overwritten by long received data packets. Overwriting of data can occur when using a smaller buffer than the packet size. In these cases the servicing of the DMA and/or GSC would be in response to the DMA Done flag when the byte count reaches zero.

In some cases the buffer size is not the limiting factor and the packet lengths will be unknown. In these cases it would be desirable to eliminate the function of the Done flag. To effectively disable the Done flag for the DMA channel servicing the receiver, the byte count should be set to some number larger than any packet that will be received, up to 64K. If not using the Done flag, then GSC servicing would be driven by the receive Done (RDN) flag and/or interrupt. RDN is set when the EOF is detected. When using the RDN flag, RFNE should also be checked to insure that all the data has been emptied out of the receive FIFO.

The byte count register is used for all transmissions and this means that all packets going out will have to be of the same length or the length of the packet to be sent will have to be known prior to the start of transmission. When using the DMA channels to service the GSC transmitter, there is no practical way to disable the Done flag. This is because the transmit done flag (TDN) is set when the transmit FIFO is empty and the last message bit has been transmitted. But, when using the DMA channel to service the transmitter, loads to the TFIFO continue to occur until the byte count reaches 0. This makes it impossible to use TDN as a flag to stop the DMA transfers to TFIFO. It is possible to examine some other registers or conditions, such as the current byte count, to determine when to stop the DMA transfers to TFIFO, but this is not recommended as a way to service the DMA and GSC when transmitting because frequent reading of the DMA registers will cause the effective DMA transfer rate to slow down.

When using the DMA channels, initialization of the GSC would be exactly the same as normal except that TSTAT.0 = 1 (DMA), this informs the GSC that the DMA channels are going to be used to service the GSC. Although only TSTAT is written to, both the receiver and transmitter use this same DMA bit.

The interrupts EGSTE (IEN1.5), GSC transmit error; EGSTV (IEN1.3), GSC transmit valid; EGSRE (IEN1.1), GSC receive error; and EGSRV (IEN1.0), GSC receive valid; need to be enabled. The DMA interrupts are normally not used when servicing the GSC with the DMA channels. To ensure that the DMA interrupts are not responded to is a function of the user software and should be checked by the software to make sure they are not enabled. Priority for these interrupts can also be set at this time. Whether to use high or low priority needs to be decided by the user. When responding to the GSC interrupts, if a buffer is being used to store the GSC information, then the DMA registers used for the buffer will probably need updating.

After this initialization, all that needs to be done when the GSC is actually going to be used is: load the byte count, set-up the source addresses for the DMA channel servicing the transmitter, set-up the destination addresses for the DMA channel servicing the receiver, and start the DMA transfer. The GSC enable bits should be set first and then the GO bits for the DMA. This initiates the data transfers.

This simplifies the maintenance of the GSC and can make the implementation of an external buffer for packetized information automatic.

An external buffer can be used as the source of data for transmission, or the destination of data from the receiver. In this arrangement, the message size is limited to the RAM size or 64K, whichever is smaller. By using an external buffer, the data can be accessed by other devices which may want access to the serial data. The amount of time required for the external data moves will also decrease. Under CPU control, a "MOVX" command would take 24 oscillator periods to complete. Under DMA control, external to internal, or internal to external, data moves take only 12 oscillator periods.

### 3.5.4 BAUD RATE

The GSC baud rate is determined by the contents of the SFR, BAUD, or the external clock. The formula used to determine the baud rate when using the internal clock is:

(fosc)/((BAUD+1)*8)

For example if a 12 MHz oscillator is used the baud rate can vary from:

12,000,000/((0+1)*8) = 1.5 MBPS

to:

12,000,000/((255+1)*8) = 5.859 KBPS

There are certain requirements that the external clock will need to meet. These requirements are specified in the data sheet. For a description of the use of the GSC with external clock please read Section 3.5.11.

### 3.5.5 INITIALIZATION

Initialization can be broken down into two major components, 1) initialization of the component so that its serial port is capable of proper communication; and 2) initialization of the system or a station so that intelligible communication can take place.

Most of the initialization of the component has already been discussed in the previous sections. Those items not covered are the parameters required for the component to effectively communicate with other components. These types of issues are common to both system and component initialization and will be covered in the following text.

Initialization of the system can be broken down into several steps. First, are the assumptions of each network station.

The first assumption is that the type of data encoding to be used is predetermined for the system and that each station will adhere to the same basic rules defining that encoding. The second assumption is that the basic protocol and line discipline is predetermined and known. This means that all stations are using CSMA/CD or SDLC or whatever, and that all stations are either full or half duplex. The third assumption is that the baud rate is preset for the whole system. Although the baud rate could probably be determined by the microprocessor just by monitoring the link, it will make it much simpler if the baud rate is known in advance.

One of the first things that will be required during system initialization is the assignment of unique addresses for each station. In a two-station only environment this is not necessary and can be ignored. However, keep in mind, that all systems should be constructed for easy future expansions. Therefore, even in only a two station system, addresses should be assigned. There are three basic ways in which addresses can be assigned. The first, and most common is preassigned addresses that are loaded into the station by the user. This could be done with a DIP-switch, through a keyboard. The second method of assigning addresses is to randomly assign an address and then check for its uniqueness throughout the system, and the third method is to make an inquiry to the system for the assignment of a unique address. Once the method of address assignment is determined, the method should become part of the specifications for the system to which all additions will have to adhere. This, then, is the final assumption.

The negotiation process may not be clear for some readers. The following two procedures are given as a guideline for dynamic address assignment.

In the first procedure, a station assumes a random address and then checks for its uniqueness throughout the system. As a station is initialized into the system it sends out a message containing its assumed address. The format of the message should be such that any station decoding the address recognizes it as a request for initialization. If that address is already used, the receiving station returns a message, with its own address stating that the address in question is already taken. The initializing station then picks another address. When the initializing station sends its inquiry for the address check, a timer is also started. If the timer expires before the inquiry is responded to, then that station assumes the address chosen is okay.

In the second procedure, an initializing station asks for an address assignment from the system. This requires that some station on the link take care of the task of maintaining a record of which addresses are used. This station will be called station-1. When the initializing station, called station-2, gets on the link, it sends out a message with a broadcast address. The format of the message should be such that all other stations on the link recognize it as a request for address assignment. Part of the message from station-2 is a random number generated by the station requesting the address. Station-2 then examines all received messages for this random number. The random number could be the address of the received message or could be within the information section of a broadcast frame. All the stations, except station-1, on the link should ignore the initialization request. Station-1, upon receiving the initialization request, assigns an address and returns it to station-2. Station-1 will be required to format the message in such a manner so that all stations on the link recognize it as a response to initialization. This means that all stations except station-2 ignore the return message.

### 3.5.6 TEST MODES

There are two test modes associated with the GSC that are made available to the user. The test modes are named Raw Receive and Raw Transmit. The test modes are selected by the proper setting of the two mode bits in GMOD (M0 = GMOD.5, M1 = GMOD.6). If M1,M0 = 0,1 then Raw Transmit is selected. If M1,M0 = 1,0 then Raw Receive is enabled. The 32-bit CRC cannot be used in any of the test modes, or else CRC errors will occur.

In Raw Transmit, the transmit output is internally connected to the Receiver input. This is intended to be used as a local loop-back test mode, so that all data written to the transmitter will be returned by the receiver. Raw Transmit can also be used to transmit user data. If Raw Transmit is used in this way the data is emitted with no preamble, flag, address, CRC, and no bit insertion. The data is still encoded with whatever format is selected, Manchester with CSMA/CD, NRZI with SDLC or as NRZ if external clocks are used. The receiver still operates as normal and in this mode most of the receive functions can be tested.

In Raw Receive, the transmitter should be externally connected to the receiver. To do this a port pin should be used to enable an external device to connect the two pins together. In Raw Receive mode the receiver acts as normal except that all bytes following the BOF are loaded into the receive FIFO, including the CRC. Also address recognition is not active but needs to be performed in software. If SDLC is selected as the protocol, zero-bit deletion is still enabled. The transmitter still operates as normal and in this mode most of the transmitter functions and an external transceiver can be tested. This is also the only way that the CRC can be read by the CPU, but the CRC error bit will not be set.

### 3.5.7 EXTERNAL DRIVER INTERFACE

A signal is provided from the C152 to enable transmitter drivers for the serial link. This is provided for systems that require more than what the GSC ports are capable of delivering. The voltage and currents that the GSC is capable of providing are the same levels as those for normal port operation. The signal used to enable the external drivers is $\overline{\text{DEN}}$. No similar signal is needed for the receiver.

$\overline{\text{DEN}}$ is active one bit time before transmission begins. In CSMA/CD $\overline{\text{DEN}}$ remains active for two bit times after the CRC is transmitted. In SDLC $\overline{\text{DEN}}$ remains active until the last bit of the EOF is transmitted.

### 3.5.8 JITTER (RECEIVE)

Data jitter is the difference between the actual transmitted waveform and the exact calculated value(s). In NRZI, data jitter would be how much the actual waveform exceeds or falls short of one calculated bit time. A bit time equals 1/baud rate. If using Manchester encoding, there can be two transitions during one bit time as shown in Figure 3.11. This causes a second parameter to be considered when trying to figure out the complete data jitter amount. This other parameter is the half-bit jitter. The half-bit jitter is comprised of the difference in time that the half-bit transition actually occurs and the calculated value. Jitter is important because if the transition occurs too soon it is considered noise, and if the transition occurs too late, then either the bit is missed or a collision is assumed.

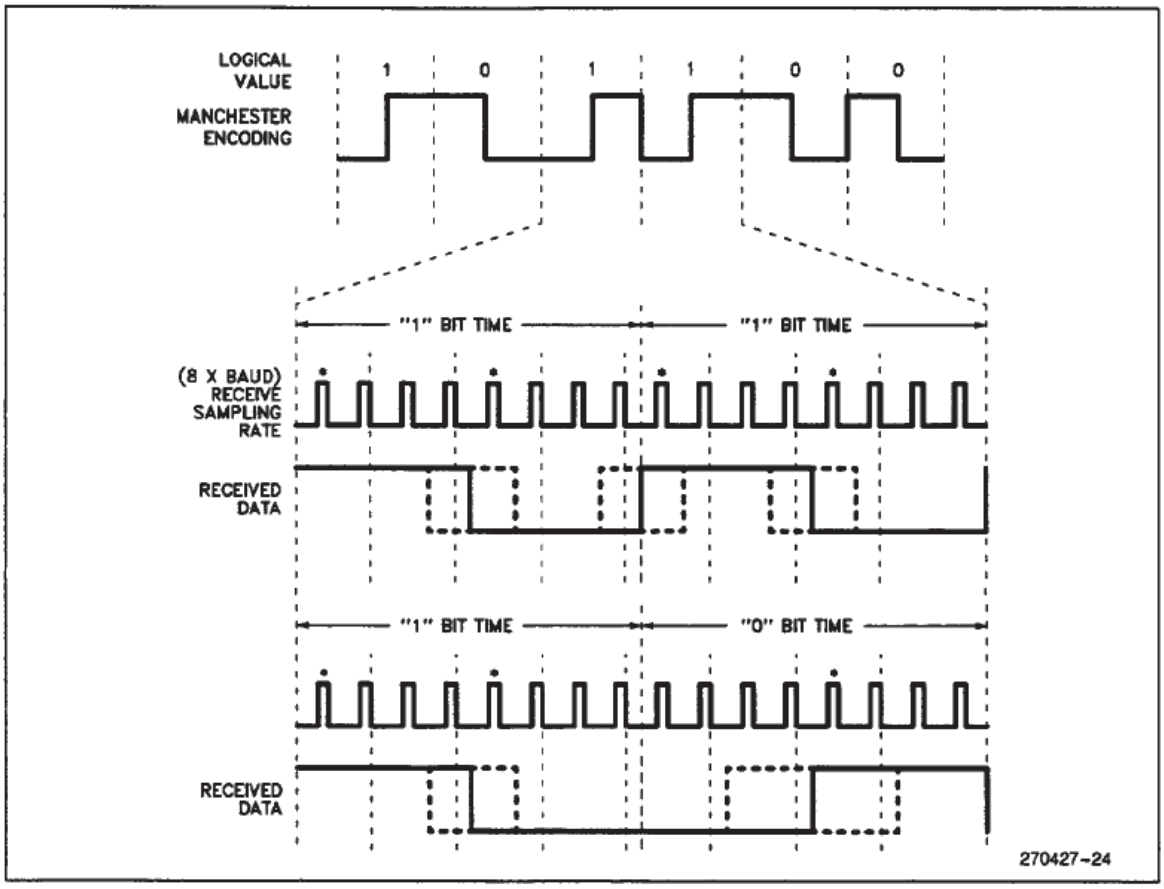



**Figure 3.11. Jitter**

### 3.5.9 Transmit Waveforms

The GSC is capable of three types of data encoding, Manchester, NRZI, and NRZ. Figure 3.12 shows examples of all three types of data encoding.

### 3.5.10 Receiver Clock Recovery

The receiver is always monitored at eight times the baud rate frequency, except when an external clock is used. When using an external clock the receiver is loaded during the clock cycle.

In CSMA/CD mode the receiver synchronizes to the transmitted data during the preamble. If a pulse is detected as being too short it is assumed to be noise or a collision. If a pulse is too long it is assumed to be a collision or an idle condition.

In SDLC the synchronization takes place during the BOF flag. In addition, pulses less than four sample periods are ignored, and assumed to be noise. This sets a lower limit on the pulse size of received zeros.

In CSMA/CD the preamble consists of alternating 1s and 0s. Consequently, the preamble looks like the waveform in Figure 3.13A and 3.13B.

### 3.5.11 External Clocking

To select external clocking, the user is given three choices. External clocking can be used with the transmitter, with the receiver, or with both. To select external clocking for the transmitter, XTCLK (GMOD.7) has to be set to a 1. To select external clocking for the receiver, XRCLK (PCON.3) has to be set to a 1. Setting both bits to 1 forces external clocking for the receiver and transmitter. The minimum frequency the GSC can be externally clocked at is 0 Hz (D.C.).

The external transmit clock is applied to pin 4 ($\overline{TXC}$), P1.3. The external receive clock is applied to pin 5 ($\overline{RXC}$), P1.4. To enable the external clock function on the port pin, that pin has to be set to a 1 in the appropriate SFR, P1.

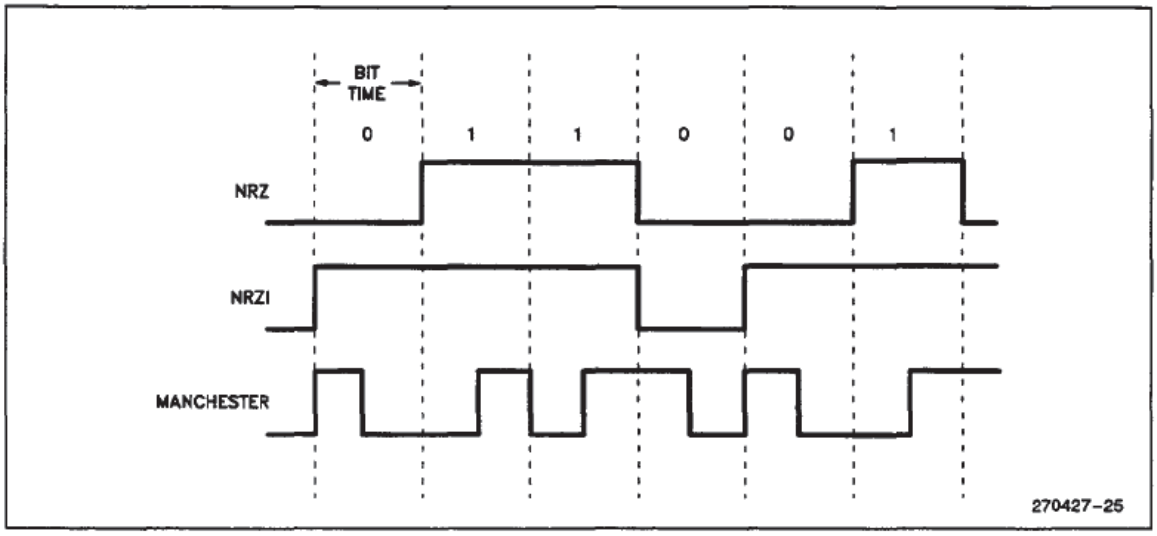



**Figure 3.12. Transmit Waveforms**

Whenever the external clock option is used, the format of the transmitted and received data is restricted to NRZ encoding and the protocol is restricted to SDLC. With external clock, the bit stuffing/stripping is still active with SDLC protocol.

### 3.5.12 Determining Receiver Errors

It is possible that several receiver error bits will be set in response to a single cause. The multiple errors that can occur are:

AE and CRCE may both be set when an alignment error occurs due to a bad CRC caused by the misaligned frame.

RCABT, AE, and CRCE may be set when an abort occurs.

OVR, AE, and CRCE may be set when a overrun occurs.

In order to determine the correct cause of the error a specific order should be followed when examining the error bits. This order is:

1) OVR
2) RCBAT
3) AE
4) CRCE

### 3.5.13 Addressing

There are four 8-bit address registers (ADR0, ADR1, ADR2, ADR3) and two 8-bit address mask registers (AMSK0, AMSK1) in the C152. These function with the GSC receiver only. The transmitted address is treated like any other data. The address is transmitted under software control by placing the address byte(s) at the proper location (usually first) in the sequence of bytes to be output in the outgoing packet.

The C152 can have up to four different 8-bit addresses or two different 16-bit addresses assigned to each station. When using 16-bit addressing, ADR0:ADR1 form one address and ADR2:ADR3 form the second address. If the receiver is enabled, it looks for a matching address after every BOF flag is detected. As the data is received, if the 8th (or 16th) bit does not match the address recognition circuitry, the rest of the frame is ignored and the search continues for another flag. If the address does match the address recognition circuitry, the address and all subsequent data is passed into the receive FIFO until the EOF flag or an error occurs. The address is not stripped and is also passed to RFIFO.

The address masking registers, AMSK0 and AMSK1, work in conjunction with ADR0 and ADR1 respectively to identify "don't care" bits. A 1 in any position in the AMSKn register makes the respective bit in the ADRn register irrelevant. These combinations can then be used for form group addresses. If the masking registers are filled with all 1s, the C152 will receive all packets, which is called the promiscuous mode. If 16-bit addressing is used, AMSK0:AMSK1 form one 16-bit address mask.

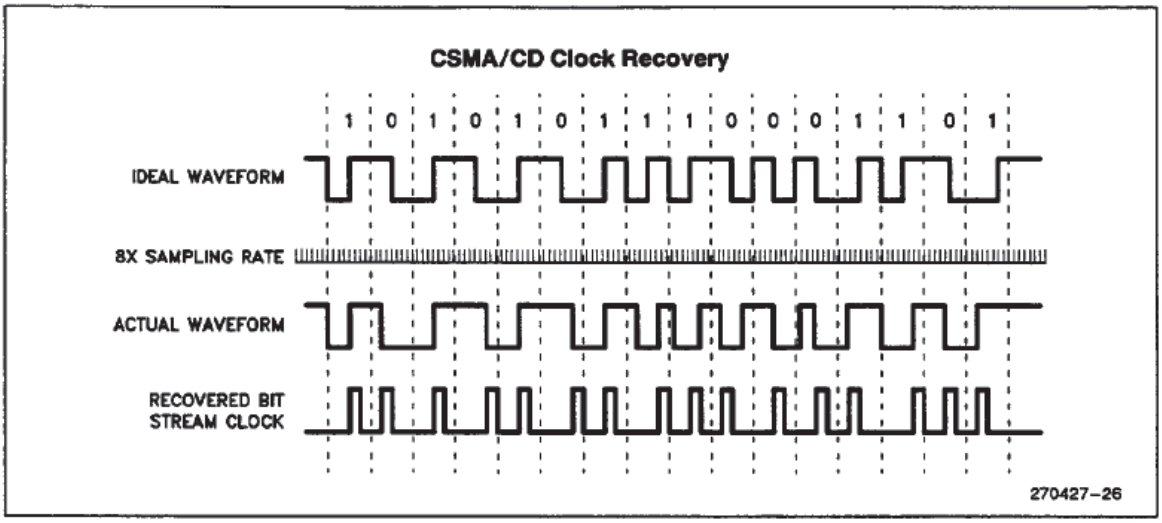



Figure 3.13A. Clock Recovery

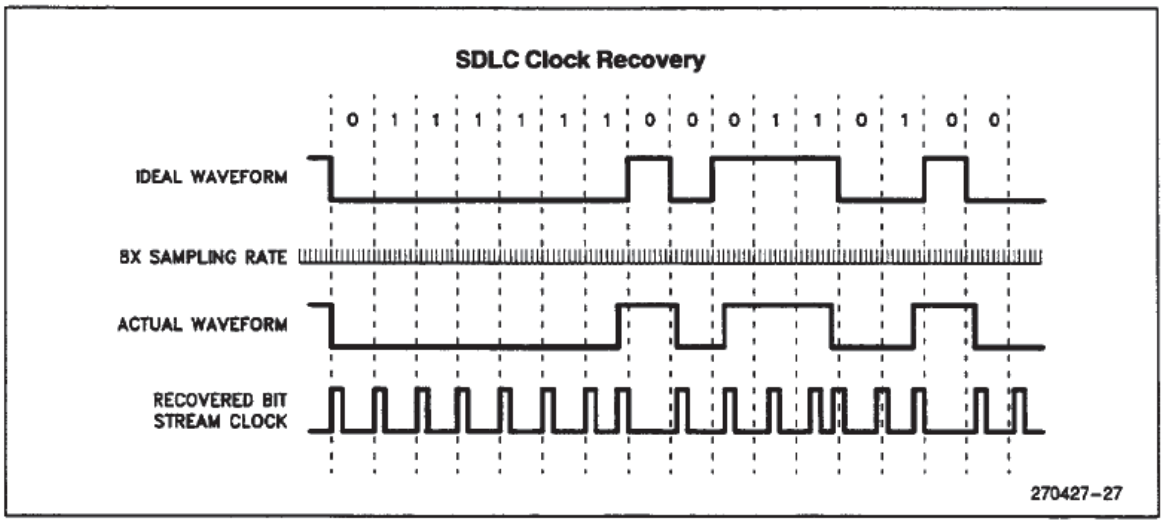



Figure 3.13B. Clock Recovery

## 3.6 GSC Operation

### 3.6.1 Determining Line Discipline

In normal operation the GSC uses full or half duplex operation. When using a 32-bit CRC (GMOD.3 = 1), operation can only be half duplex. If using a 16-bit CRC (GMOD.3 = 0), full duplex is selected by default. When using a 16-bit CRC the receiver can be turned off while transmitting (RSTAT.1 = 0), and the transmitter can be turned off during reception (TSTAT.1 = 0). This simulates half-duplex operation when using a 16-bit CRC.

Normally, HDLC uses a 16-bit CRC, so half duplex is determined by turning off the receiver or transmitter. This is so that the receiver will not detect its own address as transmission takes place. This also needs to be done when using CSMA/CD with a 16-bit CRC for the same reason.

### 3.6.2 CPU/DMA CONTROL OF THE GSC

The data for transmission or reception can be handled by either the CPU (TSTAT.0 = 0) or DMA controller (TSTAT.0 = 1). This allows the user two sets of flags to control the FIFO. Associated with these flags are interrupts, which may be enabled by the user software. Either one or both sets of flags may be used at the same time.

In CPU control mode the flags (RFNE,TFNF) are generated by the condition of the receive or transmit FIFO's. After loading a byte into the transmit FIFO, there is a one machine cycle latency until the TFNF flag is updated. Because of this latency, the status of TFNF should not be checked immediately following the instruction to load the transmit FIFO. If using the interrupts to service the transmit FIFO,the one machine cycle of latency must be considered if the TFNF flag is checked prior to leaving the subroutine.

When using the CPU for control, transmission normally is initiated by setting the TEN bit (TSTAT.1) and then writing to TFIFO. TEN must be set before loading the transmit FIFO, as setting TEN clears the transmit FIFO. TCDCNT should also be checked by user software and cleared if a collision occurred on a prior transmission.

To enable the receiver, GREN (RSTAT.1) is set. After GREN is set, the GSC begins to look for a valid BOF. After detecting a valid BOF the GSC attempts to match the received address byte(s) against the address match registers. When a match occurs the frame is loaded into the GSC. Due to the CRC strip hardware, there is a 40 or 24 bit time delay following the BOF until the first data byte is loaded into RFIFO if the 32 or 16 bit CRC is chosen. If the end of frame is detected before data is loaded into the receive FIFO, the receiver ignores that frame.

If the receiver detects a collision during reception in CSMA/CD mode and if any bytes have been loaded into the receive FIFO, the RCABT flag is set. The GSC hardware then halts reception and resets GREN. The user software needs to filter any collision fragment data which may have been received. If the collision occurred prior to the data being loaded into RFIFO the CPU is not notified and the receiver is left enabled. At the end of a reception the RDN bit is set and GREN is cleared. In HABEN mode this causes an acknowledgement to be transmitted if the frame did not have a broadcast or multi-cast address. The user software can enable the interrupt for RDN to determine when a frame is completed.

In DMA mode the interrupts are generated by the internal "transmit/receive done" (TDN,RDN) conditions. When the CPU responds to TDN or RDN, checks are performed to see if the transmit underrun error has occurred. The underrun condition is only checked when using the DMA channels.

Upon power up the CPU mode is initialized. General DMA control is covered in Section 4.0. DMA control of the GSC is covered in Section 3.5.4. If DMA is to be used for serving the GSC, it must be configured into the serial channel demand mode and the DMA bit in TSTAT has to be set.

### 3.6.3 COLLISIONS AND BACKOFF

The actions that are taken by the GSC if a collision occurs while transmitting depend on where the collision occurs. If a collision occurs in CSMA/CD mode following the preamble and BOF flag, the TCDT flag is set and the transmit hardware completes a jam. When this type of collision occurs, there will be no automatic retry at transmission. After the jam, control is returned to the CPU and user software must then initiate whatever actions are necessary for a proper recovery. The possibility that data might have been loaded into or from the GSC deserves special consideration. If these fragments of a message have been passed on to other devices, user software may have to perform some extensive error handling or notification. Before starting a new message, the transmit and receive FIFOs will need to be cleared. If DMA servicing is being used the pointers must also be reinitialized. It should be noted that a collision should never occur after the BOF flag in a well designed system, since the system slot time will likely be less than the preamble length. The occurrence of such a situation is normally due to a station on the link that is not adhering to proper CSMA/CD protocol or is not using the same timings as the rest of the network.

A collision occurring during the preamble or BOF flag is the normal type of collision that is expected. When this type of collision occurs the GSC automatically handles the retransmission attempts for as many as eight tries. If on the eighth attempt a collision occurs,

the transmitter is disabled, although the jam and back-off are performed. If enabled, the CPU is then interrupted. The user software should then determine what action to take. The possibilities range from just reporting the error and aborting transmission to reinitializing the serial channel registers and attempt retransmission.

If less than eight attempts are desired TCDCNT can be loaded with some value which will reduce the number of collisions possible before TCDCNT overflows. The value loaded should consist of all 1s as the least significant bits, e.g. 7, 0FH, 3FH. A solid block of 1s is suggested because TCDCNT is used as a mask when generating the random slot number assignment. The TCDCNT register operates by shifting the contents one bit position to the left as each collision is detected. As each shift occurs a 1 is loaded into the LSB. When TCDCNT overflows, GSC operation stops and the CPU is notified by the setting of the TCDT bit which can flag an interrupt.

The amount of time that the GSC has before it must be ready to retransmit after a collision is determined by the mode which is selected. The mode is determined M0 (GMOD.5) and M1 (GMOD.6). If M0 and M1 equal 0,0 (normal backoff) then the minimum period before retransmission will be either the interframe space or the backoff period, whichever is longer. If M0 and M1 equal 1,1 (alternate backoff) then the minimum period before retransmission will be the interframe space plus the backoff period. Both of these are shown in Figure 3.4. Alternate backoff must be enabled if using deterministic resolution. If the GSC is not ready to retransmit by the time its assigned slot becomes available, the slot time is lost and the station must wait until the collision resolution time period has passed.

Instead of waiting for the collision resolution to pass, the transmission could be aborted. The decision to abort is usually dependent on the number of stations on the link and how many collisions have already occurred. The number of collisions can be obtained by examining the register, TCDCNT. The abort is normally implemented by clearing TEN. The new transmission begins by setting TEN and loading TFIFO. The minimum amount of time available to initiate a retransmission would be one interframe space period after the line is sensed as being idle.

As the number of stations approach 256 the probability of a successful transmission decreases rapidly. If there are more than 256 stations involved in the collision there would be no resolution since at least two of the stations will always have the same backoff interval selected.

All the stations monitor the link as long as that station is active, even if not attempting to transmit. This is to ensure that each station always defers the minimum amount of time before attempting a transmission and so that addresses are recognized. However, the collision detect circuitry operates slightly differently.

In normal back-off mode, a transmitting station always monitors the link while transmitting. If a collision is detected one or more of the transmitting stations apply the jam signal and all transmitting stations enter the back-off algorithm. The receiving stations also constantly monitor for a collision but do not take part in the resolution phase. This allows a station to try to transmit in the middle of a resolution period. This in turn may or may not cause another collision. If the new station trying to transmit on the link does so during an unused slot time then there will probably not be a collision. If trying to transmit during a used slot time, then there will probably be a collision. The actions the receiver does take when detecting a collision is to just stop receiving data if data has not been loaded into RFIFO or to stop reception, clear receiver enable (REN) and set the receiver abort flag (RCABT - RSTAT.6).

If deterministic resolution is used, the transmitting stations go through pretty much the same process as in normal back-off, except that the slots are predetermined. All the receivers go through the back-off algorithm and may only transmit during their assigned slot.

### 3.6.4 SUCCESSFUL ENDING OF TRANSMISSIONS AND RECEPTIONS

In both CSMA/CD and SDLC modes, the TDN bit is set and TEN cleared at the end of a successful transmission. The end of the transmission occurs when the TFIFO is empty and the last byte has been transmitted. In CSMA/CD the user should clear the TCDCNT register after successful transmission.

At the end of a successful reception, the RDN bit is set and GREN is cleared. The end of reception occurs when the EOF flag is detected by the GSC hardware.

## 3.7 Register Descriptions

ADR0,1,2,3 (95H, 0A5H, 0B5H, 0C5H) - Address Match Registers 0,1,2,3 - Contains the address match values which determines which data will be accepted as valid. In 8 bit addressing mode, a match with any of the four registers will trigger acceptance. In 16 bit addressing mode a match with ADR1:ADR0 or ADR3:ADR2 will be accepted. Addressing mode is determined in GMOD (AL).

AMSK0,1 (0D5H, 0E5H) - Address Match Mask 0,1 - Identifies which bits in ADR0,1 are "don't care" bits. Writing a one to a bit in AMSK0,1 masks out that corresponding bit in ADDR0,1.

BAUD (94H) - GSC Baud Rate Generator - Contains the value of the programmable baud rate. The data rate will equal (frequency of the oscillator)/((BAUD + 1) × (8)). Writing to BAUD actually stores the value in a reload register. The reload register contents are copied into the BAUD register when the Baud register decrements to 00H. Reading BAUD yields the current timer value. A read during GSC operation will give a value that may not be current because the timer could decrement between the time it is read by the CPU and by the time the value is loaded into its destination.

BKOFF (0C4H) - Backoff Timer - The backoff timer is an eight bit count-down timer with a clock period equal to one slot time. The backoff time is used in the CSMA/CD collision resolution algorithm. The user software may read the timer but the value may be invalid as the timer is clocked asynchronously to the CPU. Writing to 0C4H will have no effect.

GMOD (84H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| XTCLK | M1 | M0 | AL | CT | PL1 | PL0 | PR |

**Figure 3.14. GMOD**

GMOD.0 (PR) - Protocol - If set, SDLC protocols with NRZI encoding and SDLC flags are used. If cleared, CSMA/CD link access with Manchester encoding is used. The user software is responsible for setting or clearing this flag.

GMOD.1,2 (PL0,1) - Preamble length

| PL1 | PL0 | LENGTH (BITS) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 8 |
| 1 | 0 | 32 |
| 1 | 1 | 64 |

The length includes the two bit Begin Of Frame (BOF) flag in CSMA/CD but does not include the SDLC flag. In SDLC mode, the BOF is an SDLC flag, otherwise it is two consecutive ones. Zero length is not compatible in CSMA/CD mode. The user software is responsible for setting or clearing these bits.

GMOD.3 (CT) - CRC Type - If set, 32 bit AUTODIN-II-32 is used. If cleared, 16 bit CRC-CCITT is used. The user software is responsible for setting or clearing this flag.

GMOD.4 (AL) - Address Length - If set, 16 bit addressing is used. If cleared, 8 bit addressing is used. In 8 bit mode a match with any of the 4 address registers will be accepted (ADR0, ADR1, ADR2, ADR3). "Don't Care" bits may be masked in ADR0 and ADR1 with AMSK0 and AMSK1. In 16 bit mode, addresses are matched against "ADR1:ADR0" or "ADR3:ADR2". Again, "Don't Care" bits in ADR1:ADR0 can be masked in AMSK1:AMSK0. A received address of all ones will always be recognized in any mode. The user software is responsible for setting or clearing this flag.

GMOD.5,6 (M0,M1) - Mode Select - Two test modes, an optional "alternate backoff" mode, or normal backoff can be enabled with these two bits. The user software is responsible for setting or clearing the mode bits.

| M1 | M0 | Mode |
|---|---|---|
| 0 | 0 | Normal |
| 0 | 1 | Raw Transmit |
| 1 | 0 | Raw Receive |
| 1 | 1 | Alternate Backoff |

In raw receive mode, the receiver operates as normal except that all the bytes following the BOF are loaded into the receive FIFO, including the CRC. The transmitter operates as normal.

In raw transmit mode the transmit output is internally connected to the receiver input. The internal connection is not at the actual port pin, but inside the port latch. All data transmitted is done without a preamble, flag or zero bit insertion, and without appending a CRC. The receiver operates as normal. Zero bit deletion is performed.

In alternate backoff mode the standard backoff process is modified so the the backoff is delayed until the end of the IFS. This should help to prevent collisions constantly happening because the IFS time is usually larger than the slot time.

GMOD.7 (XTCLK) - External Transmit Clock - If set an external 1X clock is used for the transmitter. If cleared the internal baud rate generator provides the transmit clock. The input clock is applied to P1.3 ($\overline{T \times C}$). The user software is responsible for setting or clearing this flag. External receive clock is enabled by setting PCON.3.

IFS (0A4H) - Interframe Spacing - Determines the number of bit times separating transmitted frames in CSMA/CD and SDLC. A bit time is equal to 1/baud rate. Only even interframe space periods can be used. The number written into this register is divided by two and loaded in the most significant seven bits. Complete interframe space is obtained by counting this seven bit number down to zero twice. A user software read of this register will give a value where the seven most significant bits gives the current count value and the least significant bit shows a one for the first count-down and a zero for the second count. The value read may not be valid as the timer is clocked in periods not necessarily associated with the CPU read of IFS. Loading this register with zero results in 256 bit times.

MYSLOT (0F5H) - Slot Address Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DCJ | DCR | SA5 | SA4 | SA3 | SA2 | SA1 | SA0 |
| SAn = SLOT ADDRESS (BITS 5 – 0) | | | | | | | |

**Figure 3.15. MYSLOT**

MYSLOT.0, 1, 2, 3, 4, 5 - Slot Address - The six address bits choose 1 of 64 slot addresses. Address 63 has the highest priority and address 1 has the lowest. A value of zero will prevent a station from transmitting during the collision resolution period by waiting until all the possible slot times have elapsed. The user software normally initializes this address in the operating software.

MYSLOT.6 (DCR) - Deterministic Collision Resolution Algorithm - When set, the alternate collision resolution algorithm is selected. Retriggering of the IFS on reappearance of the carrier is also disabled. When using this feature Alternate Backoff Mode must be selected and several other registers must be initialized. User software must initialize TCDCNT with the maximum number of slots that are most appropriate for a particular application. The PRBS register must be set to all ones. This disables the PRBS by freezing it's contents at 0FFH. The backoff timer is used to count down the number of slots based on the slot timer value setting the period of one slot. The user software is responsible for setting or clearing this flag.

MYSLOT.7 (DCJ) - D.C. Jam - When set selects D.C. type jam, when clear, selects A.C. type jam. The user software is responsible for setting or clearing this flag.

PCON (087H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SMOD | ARB | REQ | GAREN | XRCLK | GFIEN | PD | IDL |

PCON contains bits for power control, LSC control, DMA control, and GSC control. The bits used for the GSC are PCON.2, PCON.3, and PCON.4.

PCON.2 (GFIEN) - GSC Flag Idle Enable - Setting GFIEN to a 1 caused idle flags to be generated between transmitted frames in SDLC mode. SDLC idle flags consist of 01111110 flags creating the sequence 01111110011111110 . . . . . 011111110. A possible side effect of enabling GFIEN is that the maximum possible latency from writing to TFIFO until the first bit is transmitted increased from approximately 2 bit-times to around 8 bit-times. GFIEN has no effect with CSMA/CD.

PCON.3 (XRCLK) - GSC External Receive Clock Enable - Writing a 1 to XRCLK enables an external clock to be applied to pin 5 (Port 1.4). The external clock is used to determine when bits are loaded into the receiver.

PCON.4 (GAREN) - GSC Auxiliary Receiver Enable Bit - This bit needs to be set to a 1 to enable the reception of back-to-back SDLC frames. A back-to-back SDLC frame is when the EOF and BOF is shared between two sequential frames intended for the same station on the link. If GAREN contains a 0 then the receiver will be disabled upon reception of the EOF and by the time user software re-enables the receiver the first bit(s) may have already passed, in the case of back-to-back frames. Setting GAREN to a 1, prevents the receiver from being disabled by the EOF but GREN will be cleared and can be checked by user software to determine that an EOF has been received. GAREN has no effect if the GSC is in CSMA/CD mode.

PRBS (0E4H) - Pseudo-Random Binary Sequence - This register contains a pseudo-random number to be used in the CSMA/CD backoff algorithm. The number is generated by using a feedback shift register clocked by the CPU phase clocks. Writing all ones to the PRBS will freeze the value at all ones. Writing any other value to it will restart the PRBS generator. The PRBS is initialized to all zero's during RESET. A read of location 0E4H will not necessarily give the seed used in the backoff algorithm because the PRBS counters are clocked by internal CPU phase clocks. This means the contents of the PRBS may have been altered between the time when the seed was generated and before a READ has been internally executed.

RFIFO (0F4H) - Receive FIFO - RFIFO is a 3 byte buffer that is loaded each time the GSC receiver has a byte of data. Associated with RFIFO is a pointer that is automatically updated with each read of the FIFO. A read of RFIFO fetches the oldest data in the FIFO.

RSTAT (0E8H) - Receive Status Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| OR | RCABT | AE | CRCE | RDN | RFNE | GREN | HABEN |

**Figure 3.16. RSTAT**

RSTAT.0 (HABEN) - Hardware Based Acknowledge Enable - If set, enables the hardware based acknowledge feature. The user software is responsible for setting or clearing this flag.

RSTAT.1 (GREN) - Receiver Enable - When set, the receiver is enabled to accept incoming frames. The user must clear RFIFO with software before enabling the receiver. RFIFO is cleared by reading the contents of RFIFO until RFNE = 0. After each read of RFIFO, it takes one machine cycle for the status of RFNE to be updated. Setting GREN also clears RDN, CRCE, AE, and RCABT. GREN is cleared by hardware at the end of a reception or if any receive errors are detected. The user software is responsible for setting this flag and the GSC or user software can clear it. The status of GREN has no effect on whether the receiver detects a collision in CSMA/CD mode as the receiver input circuitry always monitors the receive pin.

RSTAT.2 (RFNE) - Receive FIFO Not Empty - If set, indicates that the receive FIFO contains data. The receive FIFO is a three byte buffer into which the receive data is loaded. A CPU read of the FIFO retrieves the oldest data and automatically updates the FIFO pointers. Setting GREN to a one will clear the receive FIFO. The status of this flag is controlled by the GSC. It is cleared if user empties receive FIFO.

RSTAT.3 (RDN) - Receive Done - If set, indicates the successful completion of a receiver operation. Will not be set if a CRC, alignment, abort, or FIFO overrun error occurred. The status of this flag is controlled by the GSC.

RSTAT.4 (CRCE) - CRC Error - If set, indicates that a properly aligned frame was received with a mismatched CRC. The status of this flag is controlled by the GSC.

RSTAT.5 (AE) - Alignment Error - In CSMA/CD mode, AE is set if the receiver shift register (an internal serial-to-parallel converter) is not full and the CRC is bad when an EOF is detected. In CSMA/CD the EOF is a line idle condition (see LNI) for two bit times. If the CRC is correct while in CSMA/CD mode, AE is not set and any mis-alignment is assumed to be caused by dribble bits as the line went idle. In SDLC mode, AE is set if a non-byte-aligned flag is received. CRCE may also be set. The setting of this flag is controlled by the GSC.

RSTAT.6 (RCABT) - Receiver Collision/Abort Detect - If set, indicates that a collision was detected after data had been loaded into the receive FIFO in CSMA/CD mode. In SDLC mode, RCABT indicates that 7 consecutive ones were detected prior to the end flag but after data has been loaded into the receive FIFO. AE may also be set. The setting of this flag is controlled by the GSC.

RSTAT.7 (OVR) - Overrun - If set, indicates that the receive FIFO was full and new shift register data was written into it. AE and/or CRCE may also be set. The setting of this flag is controlled by the GSC and it is cleared by user software.

SLOTTM (0BH) - Slot Time - Determines the length of the slot time used in CSMA/CD. A slot time equals (SLOTTM) × (1 / baud rate). A read of SLOTTM will give the value of the slot time timer but the value may be invalid as the timer is clocked asynchronously to the CPU. Loading SLOTTM with 0 results in 256 bit times.

TCDCNT (0D4H) - Transmit Collision Detect Count - Contains the number of collisions that have occurred if probabilistic CSMA/CD is used. The user software must clear this register before transmitting a new frame so that the GSC backoff hardware can accurately distinguish a new frame from a retransmit attempt.

In deterministic backoff mode, TCDCNT is used to hold the maximum number of slots.

TFIFO (85H) - GSC Transmit FIFO - TFIFO is a 3 byte buffer with an associated pointer that is automatically updated for each write by user software. Writing a byte to TFIFO loads the data into the next available location in the transmit FIFO. Setting TEN clears the transmit FIFO so the transmit FIFO should not be written to prior to setting TEN. If TEN is already set transmission begins as soon as data is written to TFIFO.

TSTAT (0D8) - Transmit Status Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| LNI | NOACK | UR | TCDT | TDN | TFNF | TEN | DMA |

**Figure 3.17. TSTAT**

TSTAT.0 (DMA) - DMA Select - If set, indicates that DMA channels are used to service the GSC FIFO's and GSC interrupts occur on TDN and RDN, and also enables UR to become set. If cleared, indicates that the GSC is operating in its normal mode and interrupts occur on TFNF and RFNE. For more information on DMA servicing please refer to the DMA section on DMA serial demand mode (4.2.2.3). The user software is responsible for setting or clearing this flag.

TSTAT.1 (TEN) - Transmit Enable - When set causes TDN, UR, TCDT, and NOACK flag to be reset and the TFIFO cleared. The transmitter will clear TEN after a successful transmission, a collision during the data, CRC, or end flag. The user software is responsible for setting but the GSC or user software may clear this flag. If cleared during a transmission the GSC transmit pin goes to a steady state high level. This is the method used to send an abort character in SDLC. Also $\overline{DEN}$ is forced to a high level. The end of transmission occurs whenever the TFIFO is emptied.

TSTAT.2 (TFNF) - Transmit FIFO not full - When set, indicates that new data may be written into the transmit FIFO. The transmit FIFO is a three byte buffer that loads the transmit shift register with data. The status of this flag is controlled by the GSC.

TSTAT.3 (TDN) - Transmit Done - When set, indicates the successful completion of a frame transmission. If HABEN is set, TDN will not be set until the end of the IFS following the transmitted message, so that the acknowledge can be checked. If an acknowledge is expected and not received, TDN is not set. An acknowledge is not expected following a broadcast or multi-cast packet. The status of this flag is controlled by the GSC.

TSTAT.4 (TCDT) - Transmit Collision Detect - If set, indicates that the transmitter halted due to a collision. It is set if a collision occurs during the data or CRC or if there are more than eight collisions. The status of this flag is controlled by the GSC.

TSTAT.5 (UR) - Underrun - If set, indicates that in DMA mode the last bit was shifted out of the transmit register and that the DMA byte count did not equal zero. When an underrun occurs, the transmitter halts without sending the CRC or the end flag. The status of this flag is controlled by the GSC.

TSTAT.6 (NOACK) - No Acknowledge - If set, indicates that no acknowledge was received for the previous frame. Will be set only if HABEN is set and no acknowledge is received prior to the end of the IFS. NOACK is not set following a broadcast or a multicast packet. The status of this flag is controlled by the GSC.

TSTAT.7 (LNI) - Line Idle - If set, indicates the receive line is idle. In SDLC protocol it is set if 15 consecutive ones are received. In CSMA/CD protocol, line idle is set if GR×D remains high for approximately 1.6 bit times. LNI is cleared after a transition on GR×D. The status of this flag is controlled by the GSC.

## 3.8 Serial Backplane vs. Network Environment

The C152 GSC port is intended to fulfill the needs of both serial backplane environment and the serial communication network environment. The serial backplane is where typically, only processor to processor communications take place within a self contained box. The communication usually only encompasses those items which are necessary to accomplish the dedicated task for the box. In these types of applications there may not be a need for line drivers as the distance between the transmitter and receiver is relatively short. The network environment; however, usually requires transmission of data over large distances and requires drivers and/or repeaters to ensure the data is received on both ends.

# 4.0 DMA Operation

The C152 contains DMA (Direct Memory Accessing) logic to perform high speed data transfers between any two of

Internal Data RAM
Internal SFRs
External Data RAM

If external RAM is involved, the Port 2 and Port 0 pins are used as the address/data bus, and $\overline{RD}$ and $\overline{WR}$ signals are generated as required.

Hardware is also implemented to generate a Hold Request signal and await a Hold Acknowledge response before commencing a DMA that involves external RAM.

Alternatively, the Hold/Hold Acknowledge hardware can be programmed to accept a Hold Request signal from an external device and generate a Hold Acknowledge signal in response, to indicate to the requesting device that the C152 will not commence a DMA to or from external RAM while the Hold Request is active.

## 4.1 DMA with the 80C152

The C152 contains two identical general purpose 8-bit DMA channels with 16-bit addressability: DMA0 and DMA1. DMA transfers can be executed by either channel independent of the other, but only by one channel at a time. During the time that a DMA transfer is being executed, program execution is suspended. A DMA transfer takes one machine cycle (12 oscillator

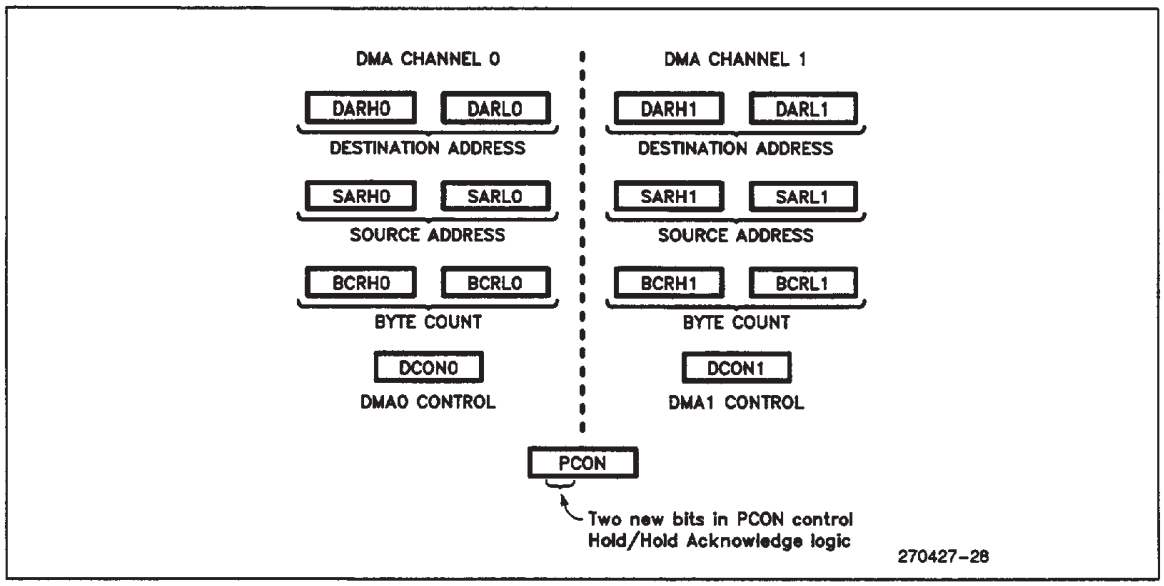



**Figure 4.1. DMA Registers**

periods) per byte transferred, except when the destination and source are both in External Data RAM. In that case the transfer takes two machine cycles per byte. The term DMA Cycle will be used to mean the transfer of a single data byte, whether it takes 1 or 2 machine cycles.

Associated with each channel are seven SFRs, shown in Figure 4.1. SARLn and SARHn holds the low and high bytes of the source address. Taken together they form a 16-bit Source Address Register. DARLn and DARHn hold the low and high bytes of the destination address, and together form the Destination Address Register. BCRLn and BCRHn hold the low and high bytes of the number of bytes to be transferred, and together form the Byte Count Register. DCONn contains control and flag bits.

Two bits in DCONn are used to specify the physical destination of the data transfer. These bits are DAS (Destination Address Space) and IDA (Increment Destination Address). If DAS = 0, the destination is in data memory external to the C152. If DAS = 1, the destination is internal to the C152. If DAS = 1 and IDA = 0, the internal destination is a Special Function Register (SFR). If DAS = 1 and IDA = 1, the internal destination is in the 256-byte data RAM.

In any case, if IDA = 1, the destination address is automatically incremented after each byte transfer. If IDA = 0, it is not.

Two other bits in DCONn specify the physical source of the data to be transferred. These are SAS (Source Address Space) and ISA (Increment Source Address). If SAS = 0, the source is in data memory external to the C152. If SAS = 1, the source is internal. If SAS = 1 and ISA = 0, the internal source is an SFR. If SAS = 1 and ISA = 1, the internal source is in the 256-byte data RAM.

In any case, if ISA = 1, the source address is automatically incremented after each byte transfer. If ISA = 0, it is not.

The functions of these four control bits are summarized below:

| DAS | IDA | Destination | Auto-Increment |
|---|---|---|---|
| 0 | 0 | External RAM | no |
| 0 | 1 | External RAM | yes |
| 1 | 0 | SFR | no |
| 1 | 1 | Internal RAM | yes |
| **SAS** | **ISA** | **Source** | **Auto-Increment** |
| 0 | 0 | External RAM | no |
| 0 | 1 | External RAM | yes |
| 1 | 0 | SFR | no |
| 1 | 1 | Internal RAM | yes |

There are four modes in which the DMA channel can operate. These are selected by the bits DM and TM (Demand Mode and Transfer Mode) in DCONn:

| DM | TM | Operating Mode |
|---|---|---|
| 0 | 0 | Alternate Cycles Mode |
| 0 | 1 | Burst Mode |
| 1 | 0 | Serial Port Demand Mode |
| 1 | 1 | External Demand Mode |

The operating modes are described below.

### 4.1.1 ALTERNATE CYCLE MODE

In Alternate Cycles Mode the DMA is initiated by setting the GO bit in DCONn. Following the instruction that set the GO bit, one more instruction is executed, and then the first data byte is transferred from the source address to the destination address. Then another instruction is executed, and then another byte of data is transferred, and so on in this manner.

Each time a data byte is transferred, BCRn (Byte Count Register for DMA Channel n) is decremented. When it reaches 0000H, on-chip hardware clears the GO bit and sets the DONE bit, and the DMA ceases. The DONE bit flags an interrupt.

### 4.1.2 BURST MODE

Burst Mode differs from Alternate Cycles mode only in that once the data transfer has begun, program execution is entirely suspended until BCRn reaches 0000H, indicating that all data bytes that were to be transferred have been transferred. The interrupt control hardware remains active during the DMA, so interrupt flags may get set, but since program execution is suspended, the interrupts will not be serviced while the DMA is in progress.

### 4.1.3 SERIAL PORT DEMAND MODE

In this mode the DMA can be used to service the Local Serial Channel (LSC) or the Global Serial Channel (GSC).

In Serial Port Demand Mode the DMA is initiated by any of the following conditions, if the GO bit is set:

| | | |
|---|---|---|
| Source Address = SBUF | .AND. | RI = 1 |
| Destination Address = SBUF | .AND. | TI = 1 |
| Source Address = RFIFO | .AND. | RFNE = 1 |
| Destination Address = TFIFO | .AND. | TFNF = 1 |

Each time one of the above conditions is met, one DMA Cycle is executed; that is, one data byte is transferred from the source address to the destination address. On-chip hardware then clears the flag (RI, TI, RFNE, or TFNF) that initiated the DMA, and decrements BCRn. Note that since the flag that initiated the DMA is cleared, it will not generate an interrupt unless DMA servicing is held off or the byte count equals 0. DMA servicing may be held off when alternate cycle is being used or by the status of the HOLD/HLDA logic. In these situations the interrupt for the LSC may occur before the DMA can clear the RI or TI flag. This is because the LSC is serviced according to the status of RI and TI, whether or not the DMA channels are being used for the transferring of data. The GSC does not use RFNE or TFNF flags when using the DMA channels so these do not need to be disabled. When using the DMA channels to service the LSC it is recommended that the interrupts (RI and TI) be disabled. If the decremented BCRn is 0000H, on-chip hardware then clears the GO bit and sets the DONE bit. The DONE bit flags an interrupt.

### 4.1.4 EXTERNAL DEMAND MODE

In External Demand Mode the DMA is initiated by one of the External Interrupt pins, provided the GO bit is set. $\overline{INT0}$ initiates a Channel 0 DMA, and $\overline{INT1}$ initiates a Channel 1 DMA.

If the external interrupt is configured to be transition-activated, then each 1-to-0 transition at the interrupt pin sets the corresponding external interrupt flag, and generates one DMA Cycle. Then, BCRn is decremented. No more DMA Cycles take place until another 1-to-0 transition is seen at the external interrupt pin. If the decremented BCRn = 0000H, on-chip hardware clears the GO bit and sets the DONE bit. If the external interrupt is enabled, it will be serviced.

If the external interrupt is configured to be level-activated, then DMA Cycles commence when the interrupt pin is pulled low, and continue for as long as the pin is held low and BCRn is not 0000H. If BCRn reaches 0 while the interrupt pin is still low, the GO bit is cleared, the DONE bit is set, and the DMA ceases. If the external interrupt is enabled, it will be serviced.

If the interrupt pin is pulled up before BCRn reaches 0000H, then the DMA ceases, but the GO bit is still 1 and the DONE bit is still 0. An external interrupt is not generated in this case, since in level-activated mode, pulling the pin to a logical 1 clears the interrupt flag. If the interrupt pin is then pulled low again, DMA transfers will continue from where they were previously stopped.

The timing for the DMA Cycle in the transition-activated mode, or for the first DMA Cycle in the level-activated mode is as follows: If the 1-to-0 transition is

detected before the final machine cycle of the instruction in progress, then the DMA commences as soon as the instruction in progress is completed. Otherwise, one more instruction will be executed before the DMA starts. No instruction is executed during any DMA Cycle.

## 4.2 Timing Diagrams

Timing diagrams for single-byte DMA transfers are shown in Figures 4.2 through 4.5 for four kinds of DMA Cycles: internal memory to internal memory, internal memory to external memory, external memory to internal memory, and external memory to external memory. In each case we assume the C152 is executing out of external program memory. If the C152 is executing out of internal program memory, then $\overline{PSEN}$ is inactive, and the Port 0 and Port 2 pins emit P0 and P2 SFR data. If External Data Memory is involved, the Port 0 and Port 2 pins are used as the address/data bus, and $\overline{RD}$ and/or $\overline{WR}$ signals are generated as needed, in the same manner as in the execution of a MOVX @DPTR instruction.

## 4.3 Hold/Hold Acknowledge

Two operating modes of Hold/Hold Acknowledge logic are available, and either or neither may be invoked by software. In one mode, the C152 generates a Hold Request signal and awaits a Hold Acknowledge response before commencing a DMA that involves external RAM. This is called the Requester Mode.

In the other mode, the C152 accepts a Hold Request signal from an external device and generates a Hold Acknowledge signal in response, to indicate to the requesting device that the C152 will not commence a DMA to or from external RAM while the Hold Request is active. This is called the Arbiter mode.

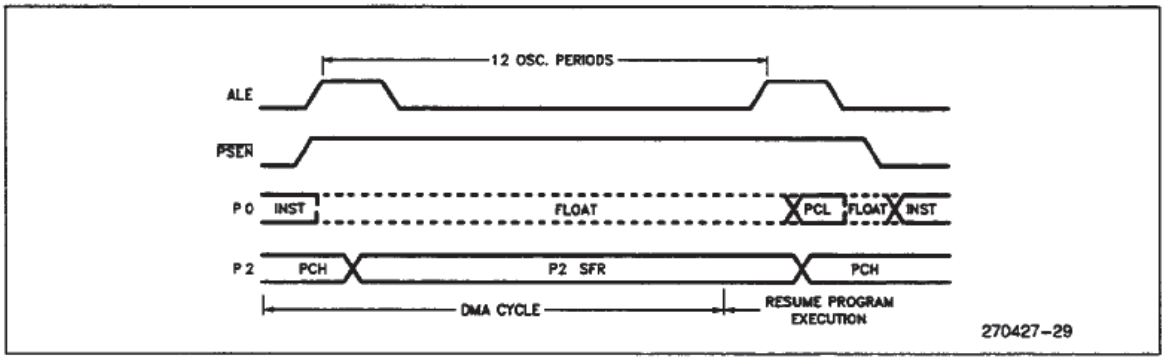



**Figure 4.2. DMA Transfer from Internal Memory to Internal Memory**

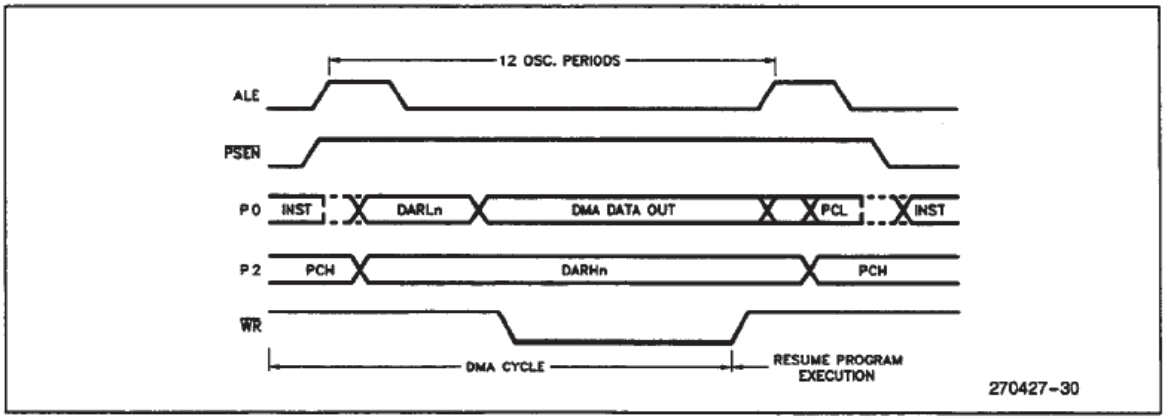



**Figure 4.3. DMA Transfer from Internal Memory to External Memory**

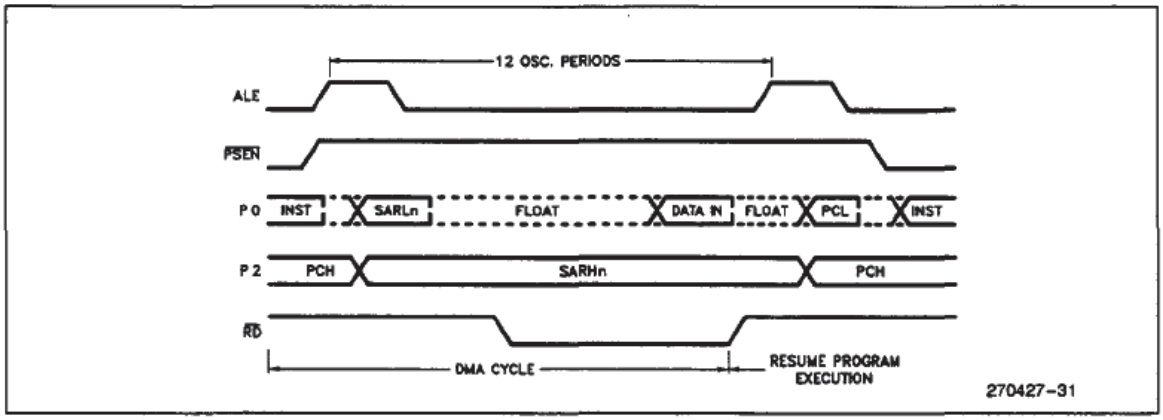



**Figure 4.4. DMA Transfer from External Memory to Internal Memory**

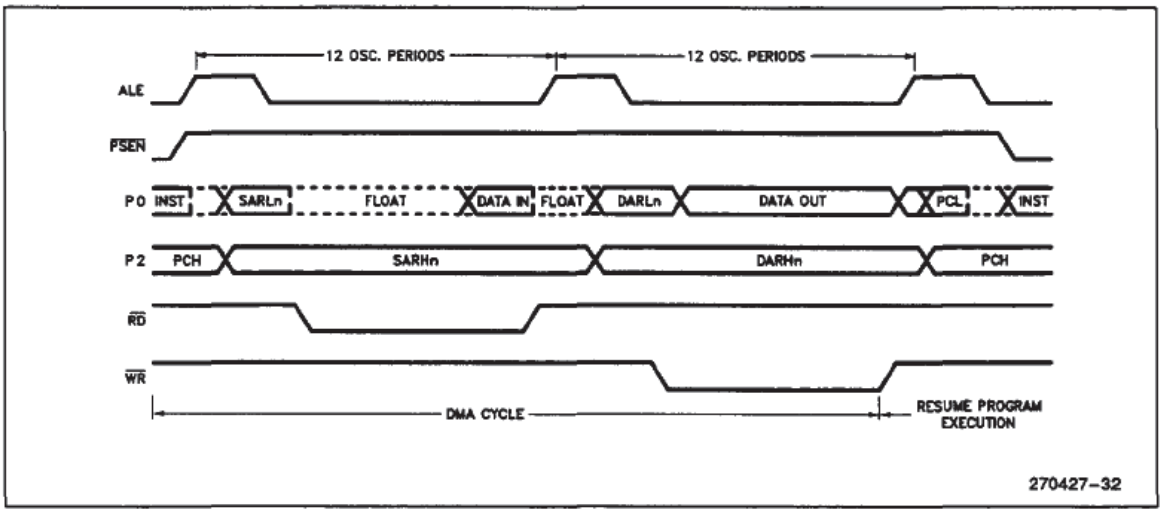



**Figure 4.5. DMA Transfer from External Memory to External Memory**

### 4.3.1 REQUESTER MODE

The Requester Mode is selected by setting the control bit REQ, which resides in PCON. In that mode, when the C152 wants to do a DMA to External Data Memory, it first generates a Hold Request signal, $\overline{\text{HLD}}$, and waits for a Hold Acknowledge signal, $\overline{\text{HLDA}}$, before commencing the DMA operation. Note that program execution continues while $\overline{\text{HLDA}}$ is awaited. The DMA is not begun until a logical 0 is detected at the $\overline{\text{HLDA}}$ pin. Then, once the DMA has begun, it goes to completion regardless of the logic level at $\overline{\text{HLDA}}$.

The protocol is activated only for DMAs (not for program fetches or MOVX operations), and only for DMAs to or from External Data Memory. If the data destination and source are both internal to the C152, the $\overline{\text{HLD}}$/$\overline{\text{HLDA}}$ protocol is not used.

The $\overline{\text{HLD}}$ output is an alternate function of port pin P1.5, and the $\overline{\text{HLDA}}$ input is an alternate function of port pin P1.6.

### 4.3.2 ARBITER MODE

For DMAs that are to be driven by some device other than the C152, a different version of the Hold/Hold Acknowledge protocol is available. In this version, the device which is to drive the DMA sends a Hold Request signal, $\overline{\text{HLD}}$, to the C152. If the C152 is currently performing a DMA to or from External Data Memory, it will complete this DMA before responding to the Hold Request. When the C152 responds to the Hold Request, it does so by activating a Hold Acknowledge signal, $\overline{\text{HLDA}}$. This indicates that the C152 will not commence a new DMA to or from External Data Memory while $\overline{\text{HLD}}$ remains active.

Note that in the Arbiter Mode the C152 does not suspend program execution at all, even if it is executing from external program memory. It does not surrender use of its own bus.

The Hold Request input, $\overline{\text{HLD}}$, is at P1.5. The Hold Acknowledge output, $\overline{\text{HLDA}}$, is at P1.6. This

version of the Hold/Hold Acknowledge feature is selected by setting the control bit ARB in PCON.

The functions of the ARB and REQ bits in PCON, then, are

| ARB | REQ | Hold/Hold Acknowledge Logic |
|---|---|---|
| 0 | 0 | Disabled |
| 0 | 1 | C152 generates $\overline{HLD}$, detects $\overline{HLDA}$ |
| 1 | 0 | C152 detects $\overline{HLD}$, generates $\overline{HLDA}$ |
| 1 | 1 | Invalid |

### 4.3.3 USING THE HOLD/HOLD ACKNOWLEDGE

The $\overline{HOLD}$/$\overline{HOLDA}$ logic only affects DMA operation with external RAM and doesn't affect other operations with external RAM, such as MOVX instruction.

Figure 4.6 shows a system in which two 83C152s are sharing a global RAM. In this system, both CPUs are executing from internal ROM. Neither CPU uses the bus except to access the shared RAM, and such accesses are done only through DMA operations, not by MOVX instructions.

One CPU is programmed to be the Arbiter and the other, to be the Requester. The ALE Switch selects which CPU's ALE signal will be directed to the address latch. The Arbiter's ALE is selected if $\overline{HLDA}$ is high, and the Requester's ALE is selected if $\overline{HLDA}$ is low.

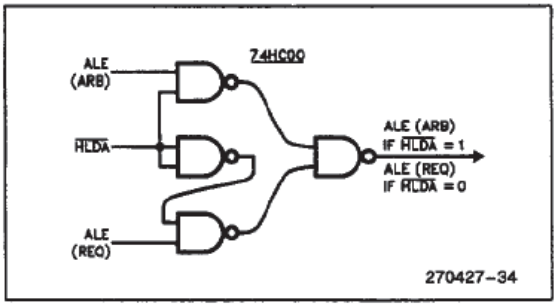



**Figure 4.7. ALE Switch Select**

The ALE Switch logic can be implemented by a single 74HC00, as shown in Figure 4.7.

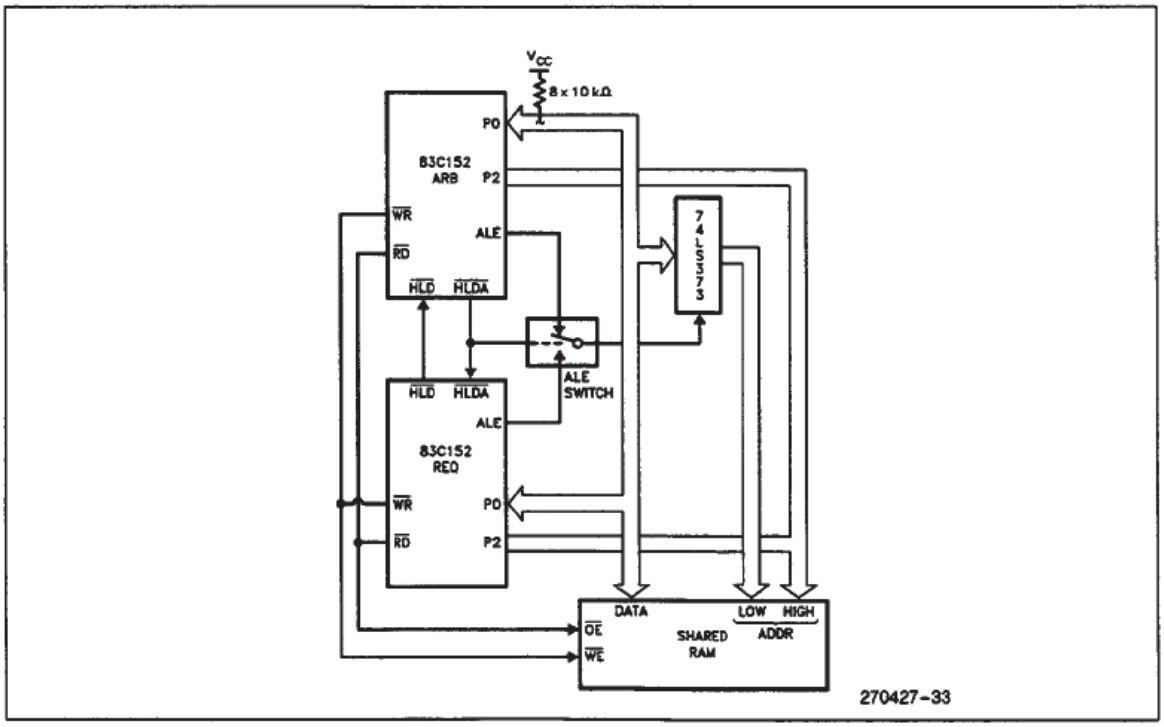



**Figure 4.6. Two 83C152s Sharing External RAM**

### 4.3.4 INTERNAL LOGIC OF THE ARBITER

The internal logic of the arbiter is shown in Figure 4.8. In operation an input low at $\overline{HLD}$ sets Q2 if the arbiter's internal signal DMXRQ is low. DMXRQ is the arbiter's "DMA to XRAM Request". Setting Q2 activates $\overline{HLDA}$ through Q3. Q2 being set also disables any DMAs to XRAM that the arbiter might decide to do during the requester's DMA.

Figure 4.9 shows the minimum response time, 4 to 7 CPU oscillator periods, between a transition at the $\overline{HLD}$ input and the response at $\overline{HLDA}$.

When the arbiter wants to DMA the XRAM, it first activates DMXRQ. This signal prevents Q2 from being set if it is not already set. An output low from Q2 enables the arbiter to carry out its DMA to XRAM, and maintains an output high at $\overline{HLDA}$. When the arbiter completes its DMA, the signal DMXRQ goes to 0, which enables Q2 to accept signals from the $\overline{HLD}$ input again.

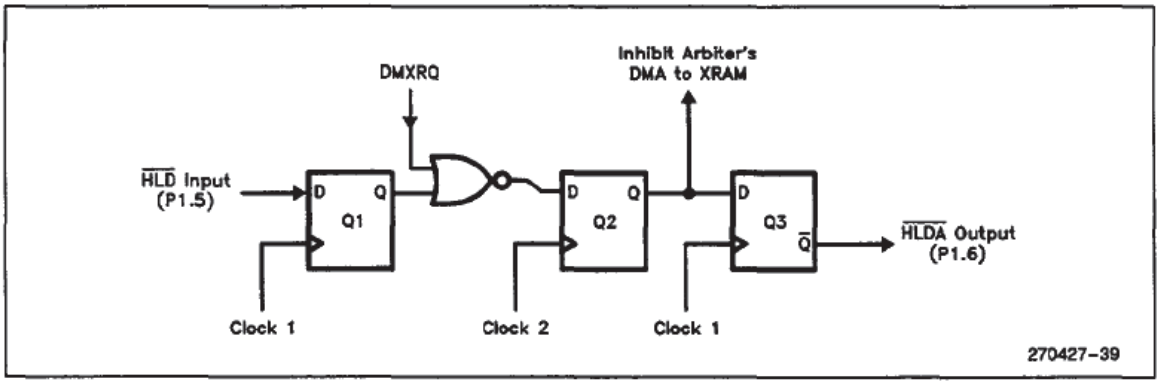



**Figure 4.8. Internal Logic of the Arbiter**

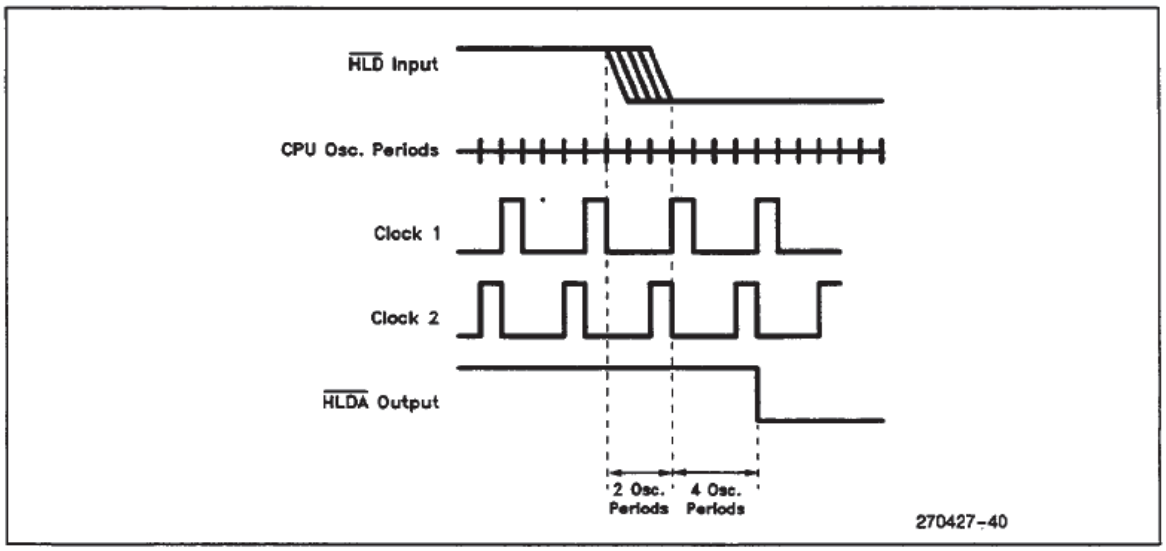



**Figure 4.9. Minimum HLD/HLDA Response Time**

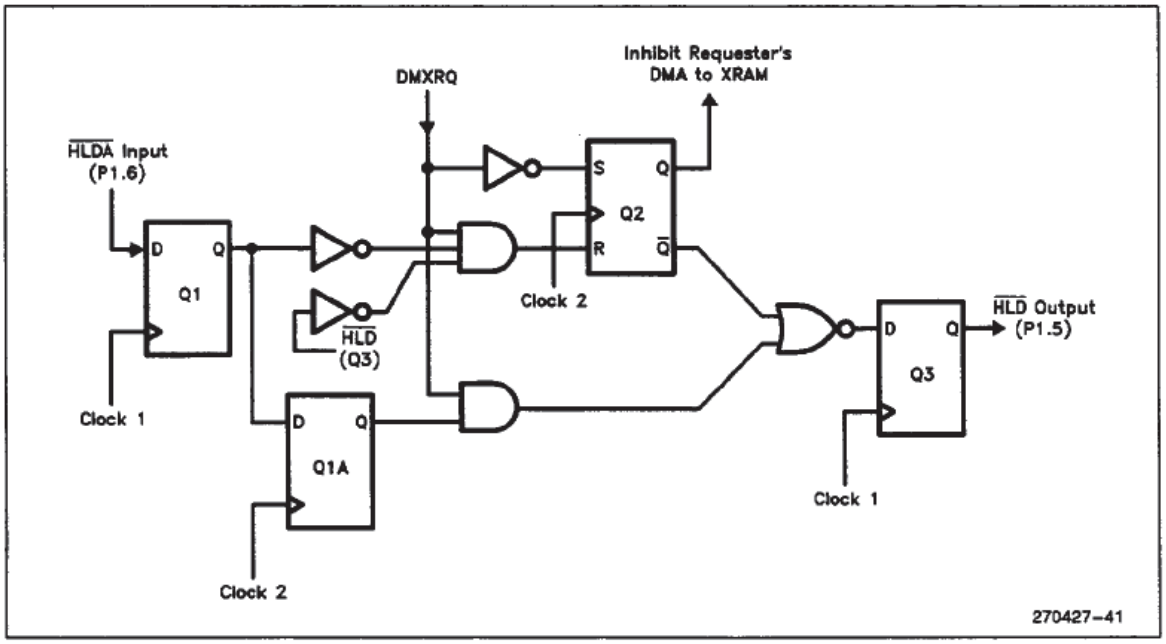



**Figure 4.10. Internal Logic of the Requester (Clock 1 and Clock 2 are Shown in Figure 4.9)**

### 4.3.5 Internal Logic of the Requester

The internal logic of the requester is shown in Figure 4.10. Initially, the requester's internal signal DMXRQ (DMA to XRAM Request) is at 0, so Q2 is set and the $\overline{HLD}$ output is high. As long as Q2 stays set, the requester is inhibited from starting any DMA to XRAM.

When the requester wants to DMA the XRAM, it first activates DMXRQ. This signal enables Q2 to be cleared (but doesn't clear it), and, if $\overline{HLDA}$ is high, also activates the $\overline{HLD}$ output.

A 1-to-0 transition from $\overline{HLDA}$ can now clear Q2, which will enable the requester to commence its DMA to XRAM. Q2 being low also maintains an output low at $\overline{HLD}$. When the DMA is completed, DMXRQ goes to 0, which sets Q2 and de-activates $\overline{HLD}$.

Only DMXRQ going to 0 can set Q2. That means once Q2 gets cleared, enabling the requester's DMA to proceed, the arbiter has no way to stop the requester's DMA in progress. At this point, de-activating $\overline{HLDA}$ will have no effect on the requester's use of the bus. Only the requester itself can stop the DMA in progress, and when it does, it de-activates both DMXRQ and $\overline{HLD}$.

If the DMA is in alternate cycles mode, then each time a DMA cycle is completed DMXRQ goes to 0, thus de-activating $\overline{HLD}$. Once $\overline{HLD}$ has been de-activated, it can't be re-asserted till after $\overline{HLDA}$ has been seen to go high (through flip-flop Q1A). Thus every time the DMA is suspended to allow an instruction cycle to proceed, the requester gives up the bus and must renew the request and receive another acknowledge before another DMA cycle to XRAM can proceed. Obviously in this case, the "alternate cycles" mode may consist of single DMA cycles separated by any number of instruction cycles, depending on how long it takes the requester to regain the bus.

A channel 1 DMA in progress will always be overridden by a DMA request of any kind from channel 0. If a channel 1 DMA to XRAM is in progress and is overridden by a channel 0 DMA which does not require the bus, DMXRQ will go to 0 during the channel 0 DMA, thus de-activating $\overline{HLD}$. Again, the requester must renew its request for the bus, and must receive a new 1-to-0 transition in $\overline{HLDA}$ before channel 1 can continue its DMA to XRAM.

## 4.4 DMA Arbitration

The DMA Arbitration described in this section is not arbitration between two devices wanting to access a shared RAM, but on-chip arbitration between the two DMA channels on the 8XC152.

The 8XC152 provides two DMA channels, either of which may be called into operation at any time in response to real time conditions in the application circuit. Since a DMA cycle always uses the 8XC152's internal bus, and there's only one internal bus, only one DMA channel can be serviced during a single DMA cycle. Executing program instructions also requires the internal bus, so program execution will also be suspended in order for a DMA to take place.

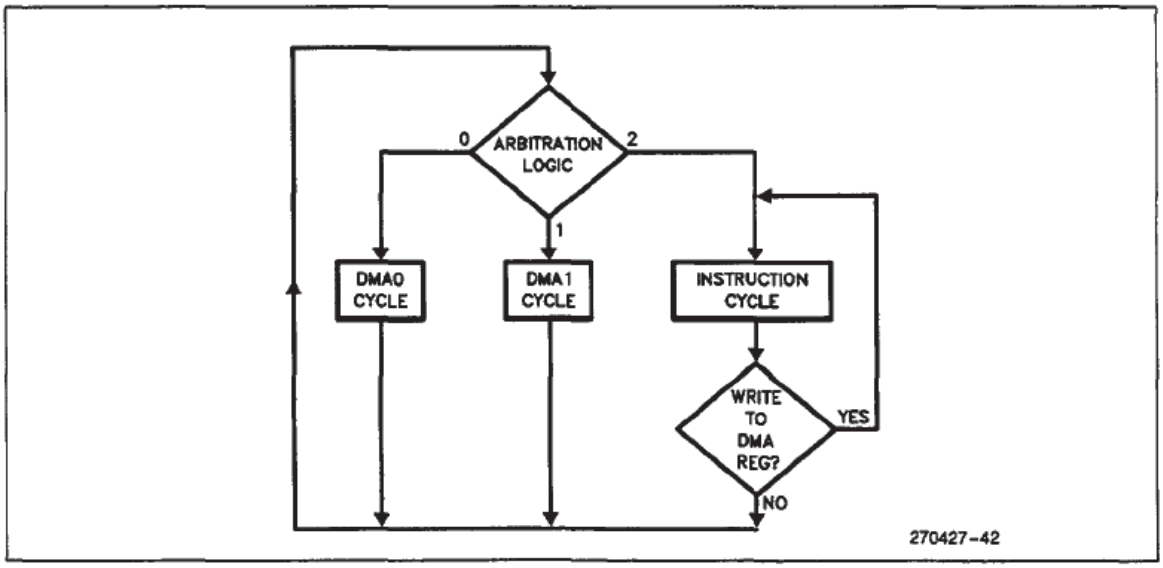



**Figure 4.11. Internal Bus Usage**

Figure 4.11 shows the three tasks to which the internal bus of the 8XC152 can be dedicated. In this figure, Instruction Cycle means the complete execution of a single instruction, whether it takes 1, 2 or 4 machine cycles. DMA Cycle means the transfer of a single data byte from source to destination, whether it takes 1 or 2 machine cycles. Each time a DMA Cycle or an Instruction Cycle is executed, on-chip arbitration logic determines which type of cycle is to be executed next.

Note that when an instruction is executed, if the instruction wrote to a DMA register (defined in Figure 4.1 but excluding PCON), then another instruction is executed without further arbitration. Therefore, a single write or a series of writes to DMA registers will prevent a DMA from taking place, and will continue to prevent a DMA from taking place until at least one instruction is executed which does not write to any DMA register.

The logic that determines whether the next cycle will be a DMA0 cycle, a DMA1 cycle, or an Instruction Cycle is shown in Figure 4.12 as a pseudo-HLL function. The statements in Figure 4.12 are executed sequentially unless an "if" condition is satisfied, in which case the corresponding "return" is executed and the remainder of the function is not. The return value of 0, 1, or 2 is passed to the arbitration logic block in Figure 4.11 to determine which exit path from the block is used.

The return value is based on the condition of the GO bit for each channel, and on the value returned by another function, named mode__logic ( ). The algorithm for mode__logic ( ) is the same for both channels. The function is shown in Figure 4.13 as a pseudo-HLL function, mode__logic (n), where n = 0 when the function is invoked for DMA channel 0, and n = 1 when it's invoked for DMA channel 1. The value returned by this function is either 0 or 1, and will be passed on to the DMA arbitration logic in Figure 4.12.

Note that the arbitration logic as shown in Figure 4.12 always gives precedence to channel 0 over channel 1. If GO0 is set and mode__logic (0) returns a 1, then a DMA0 cycle is called without further reference to the situation in channel 1. That is not to say a DMA1 Cycle will be interrupted once it has begun. Once a cycle has begun, be it an Instruction Cycle or a DMA Cycle, it will be completed without interruption.

The statements in mode__logic (n), Figure 4.13, are executed sequentially until an "if" condition, based on the DMA mode programmed into DCONn, is satisfied. For example, if the channel is configured to Burst mode, then the first if-condition is satisfied, so the "return 1" expression is executed and the remainder of the function is not.

```
arbitration_logic:

    if (GO0 = 1 .AND. mode_logic(0) = 1) return 0;

    if (GO1 = 1 .AND. mode_logic(1) = 1) return 1;

    else return 2;

    end arbitration_logic;
```

**Figure 4.12. DMA Arbitration Logic**

```
mode_logic(n):
    if (DCONn indicates burst_mode) return 1;
    if (DCONn indicates extern_demand_mode)
        {
        if (demand_flag = 1) return 1;
        else return 0;
        }
    if (DCONn indicates SP_demand_mode)
        {
        if (SARn = SBUF .AND. RI = 1) return 1;
        if (DARn = SBUF .AND. TI = 1) return 1;
        if (SARn = RFIFO .AND. RFNE = 1) return 1;
        if (DARn = TFIFO .AND. TFNF = 1 .AND.
         previous_cycle = instruction_cycle) return 1;
        else return 0;
        }
    if (DCONn indicates alt_cycles_mode)
        {
        if (DCONm indicates .NOT. alt_cycles_mode
            .OR. GOm = 0)
            {
            if (previous_cycle = instruction_cycle)
                return 1;
            else return 0;
            }
        if (previous_cycle = instruction_cycle
            .AND. previous_dma_cycle = .NOT. DMAn)
            return 1;
        }
    return 0;
    end mode_logic(n);
```

**Figure 4.13. DMA Mode Logic**

If the channel is configured to External Demand mode, then the first if-condition is not satisfied but the second one is. In that case the block of statements following that if-condition and delimited by {...} is executed: if the demand flag (IE0 for channel 0 and IE1 for channel 1) is set, the "return 1" expression is executed and the remainder of the function is not. If the demand flag is not set, the "return 0" expression is executed and the remainder of the function is not.

If the channel is configured to Serial Port Demand mode, the source and destination addresses, SARn and DARn, have to be checked to see which Serial Port buffer is being addressed, and whether its demand flag is set.

SARn refers to the 16-bit source address for "this channel." Note that the condition

SARn = SBUF

cannot be true unless the SAS and ISA bits in DCONn are configured to select SFR space. If SARn is numerically equal to the address of SBUF (99H), and SAS and ISA are configured to select internal RAM rather than SFR space, then SARn refers to location 99H in the "upper 128" of internal RAM, not to SBUF.

If the test for SARn = SBUF *is* true, and if the flag RI is set, mode_logic (n) returns as 1 and the remainder of the function is not executed. Otherwise, execution proceeds to the next if-condition, testing DARn against SBUF and T1 against 1.

The same considerations regarding SAS and ISA in the SARn test are now applied to DAS and IDA in the DARn test. If SFR space isn't selected, no Serial Port buffer is being addressed.

Note that if DMA channel n is configured to Alternate Cycles mode, the logic must examine the other DCON register, DCONm, to determine if the other channel is also configured to Alternate Cycles mode and whether its GO bit is set. In Figure 4.13, the symbol DCONn refers to the DCON register for "this channel," and DCONm refers to "the other channel."

A careful examination of the logic in Figure 4.13 will reveal some idiosyncracies that the user should be aware of. First, the logic allows sequential DMA cycles to be generated to service RFIFO, but not to service TFIFO. This idiosyncracy is due to internal timing conflicts, and results in each individual DMA cycle to TFIFO having to be immediately preceded by an Instruction cycle. The logic disallows that there be two DMAs to TFIFO in a row.

If the user is unaware of this idiosyncracy, it can cause problems in situations where one DMA channel is servicing TFIFO and the other is configured to a completely different mode of operation.

For example, consider the situation where channel 0 is configured to service TFIFO and channel 1 is configured to Alternate Cycles mode. Then DMAs to TFIFO will always override the alternate cycles of channel 1. If TFIFO needs more than 1 byte it will receive them in precendence over channel 1, but each DMA to TFIFO must be preceded by an Instruction cycle. The sequence of cycles might be:

DMA1 cycle
Instruction cycle
DMA1 cycle, during which TFNF gets set
Instruction cycle
DMA0 cycle
Instruction cycle
DMA0 cycle, as a result of which TFNF gets cleared
Instruction cycle
DMA1 cycle
Instruction cycle
DMA1 cycle
Instruction cycle
. . .

The requirement that a DMA to TFIFO be preceded by an Instruction cycle can result in the normal precedence of channel 0 over channel 1 being thwarted. Consider for example the situation where channel 0 is configured to service TFIFO, and is in the process of doing so, and channel 1 decides it wants to do a Burst mode DMA. The sequence of events might be:

Instruction cycle (sets GO bit in DCON1)
Instruction cycle (during which TFNF gets set)
DMA0 cycle
DMA1 cycle
DMA1 cycle
DMA1 cycle
. . .
DMA1 cycle (completes channel 1 burst)
Instruction cycle
DMA0 cycle
Instruction cycle
. . .

This sequence begins with two Instruction cycles. The first one accesses a DMA register (DCON1), and therefore is followed by another Instruction cycle, which presumably does not access a DMA register. After the second Instruction cycle both channels are ready to generate DMA cycles, and channel 0 of course takes precedence. After the DMA0 cycle, channel 0 must wait for an Instruction cycle before it can access TFIFO again. Channel 1, being in Burst mode, doesn't have that restriction, and is therefore granted a DMA1 cycle. After the first DMA1 cycle, channel 0 is still waiting for an Instruction cycle and channel 1 still does not have that restriction. There follows another DMA1 cycle.

The result is that in this particular case channel 0 has to wait until channel 1 completes its Burst mode DMA, and then has to wait for an Instruction cycle to be generated, before it can continue its own DMA to TFIFO. The delay in servicing TFIFO can cause an Underflow condition in the GSC transmission.

The delay will not occur if channel 1 is configured to Alternate Cycles mode, since channel 0 would then see the Instruction cycles it needs to complete its logic requirements for asserting its request.

### 4.4.1 DMA Arbitration with Hold/Hold Ack

The Hold/Hold Acknowledge feature is invoked by setting either the ARB or REQ bit in PCON. Their effect is to add the requirements of the Hold/Hold Ack protocol to mode_logic ( ). This amounts to replacing every expression "return 1" in Figure 4.13 with the expression "return hld_hlda_logic ( )", where hld_hlda_logic ( ) is a function which returns 1 if the Hold/Hold Ack protocol is satisfied, and returns 0 otherwise. A suitable definition for hld_hlda_logic ( ) is shown in Figure 4.14.

## 4.5 Summary of DMA Control Bits

| DCONn | DAS | IDA | SAS | ISA | DM | TM | DONE | GO |
|---|---|---|---|---|---|---|---|---|

DAS specifies the Destination Address Space. If DAS = 0, the destination is in External Data Memory. If DAS = 1 and IDA = 0, the destination is a Special Function Register (SFR). If DAS = 1 and IDA = 1, the destination is in Internal Data RAM.

IDA (Increment Destination Address) If IDA = 1, the destination address is automatically incremented after each byte transfer. If IDA = 0, it is not.

SAS specifies the Source Address Space. If SAS = 0, the source is in External Data Memory. If SAS = 1 and ISA = 0, the source is an SFR. If SAS = 1 and ISA = 1, the source is Internal Data RAM.

ISA (Increment Source Address) If ISA = 1, the source address is automatically incremented after each byte transfer. If ISA = 0, it is not.

DM (Demand Mode) If DM = 1, the DMA Channel operates in Demand Mode. In Demand Mode the DMA is initiated either by an external signal or by a Serial Port flag, depending on the value of the TM bit. If DM = 0, the DMA is requested by setting the GO bit in software.

TM (Transfer Mode) If DM = 1 then TM selects whether a DMA is initiated by an external signal (TM = 1) or by a Serial Port flag (TM = 0). If DM = 0 then TM selects whether the data transfers are to be in bursts (TM = 1) or in alternate cycles (TM = 0).

DONE indicates the completion of a DMA operation and flags an interrupt. It is set to 1 by on-chip hardware when BCRn = 0, and is cleared to 0 by on-chip hardware when the interrupt is vectored to. It can also be set or cleared by software.

```
hold_holda( ):
    if (ARB = 0 .AND. REQ = 0) return 1;
    if SARn = XRAM .OR. DARn = XRAM)
        {
        if (ARB = 1 .AND. HLDA = 1) return 1;
        if (REQ = 1 .AND. HLDA = 0) return 1;
        else return 0;
        }
    return 1;
    end hold_holda( );
```

**Figure 4.14. Hold/Hold Acknowledge Logic as a Pseudo-HLL Function**

GO is the enable bit for the DMA Channel itself. The DMA Channel is inactive if GO = 0.

| PCON | SMOD | ARB | REQ | GAREN | XRCLK | GFIEN | PDN | IDL |
|---|---|---|---|---|---|---|---|---|

ARB enables the DMA logic to detect $\overline{HLD}$ and generate $\overline{HLDA}$. After it has activated $\overline{HLDA}$, the C152 will not begin a new DMA to or from External Data Memory as long as $\overline{HLD}$ is seen to be active. This logic is disabled when ARB = 0, and enabled when ARB = 1.

REQ enables the DMA logic to generate $\overline{HLD}$ and detect $\overline{HLDA}$ before performing a DMA to or from External Data Memory. After it has activated $\overline{HLD}$, the C152 will not begin the DMA until $\overline{HLDA}$ is seen to be active. This logic is disabled when REQ = 0, and enabled when REQ = 1.

## 5.0 INTERRUPT STRUCTURE

The 8XC152 retains all five interrupts of the 80C51BH. Six new interrupts are added in the 8XC152, to support its GSC and the DMA features. They are as listed below, and the flags that generate them are shown in Figure 5.1.

GSCRV — GSC Receive Valid
GSCRE — GSC Receive Error
GSCTV — GSC Transmit Valid
GSCTE — GSC Transmit Error
DMA0 — DMA Channel 0 Done
DMA1 — DMA Channel 1 Done

As shown in Figure 5.1, the Receive Valid interrupt can be signaled either by the RFNE flag (Receive FIFO Not Empty), or by the RDN flag (Receive Done). Which one of these flags causes the interrupt depends on the setting of the DMA bit in the SFR named TSTAT.

DMA = 0 means the DMA hardware is not configured to service the GSC, so the CPU will service it in software in response to the Receive FIFO not being empty. In that case, RFNE generates the Receive Valid interrupt.

DMA = 1 means the DMA hardware is configured to service the GSC, in which case the CPU need not be interrupted till the receive is complete. In that case, RDN generates the Receive Valid interrupt.

Similarly the Transmit Valid interrupt can be signaled either by the TFNF flag (Transmit FIFO Not Full), or by the TDN flag (Transmit Done), depending on whether the DMA bit is 0 or 1.

Note that setting the DMA bit does not itself configure the DMA channels to service the GSC. That job must be done by software writes to the DMA registers. The DMA bit only selects whether the GSCRV and GSCTV interrupts are flagged by a FIFO needing service or by an "operation done" signal.

The Receive and Transmit Error interrupt flags are generated by the logical OR of a number of error conditions, which are described in Section 3.6.5.

Each interrupt is assigned a fixed location in Program Memory, and the interrupt causes the CPU to jump to that location. All the interrupt flags are sampled at S5P2 of every machine cycle, and then the samples are sequentially polled during the next machine cycle. If more than one interrupt of the same priority is active, the one that is highest in the polling sequence is serviced first. The interrupts and their fixed locations in Program Memory are listed below in the order of their polling sequence.

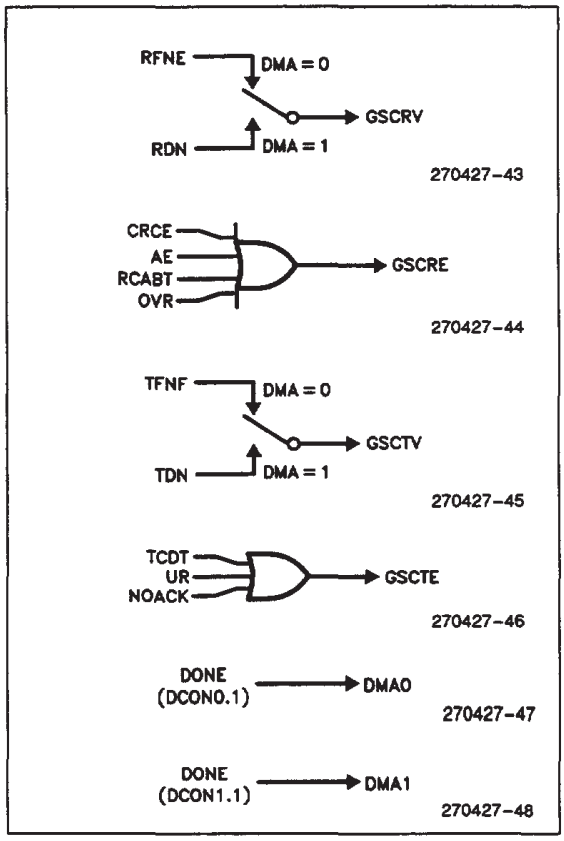



**Figure 5.1. Six New Interrupts in the 8XC152**

| Interrupt | Location | Name |
|---|---|---|
| IE0 | 0003H | External Interrupt 0 |
| GSCRV | 002BH | GSC Receive Valid |
| TF0 | 000BH | Timer 0 Overflow |
| GSCRE | 0033H | GSC Receive Error |
| DMA0 | 003BH | DMA Channel 0 Done |
| IE1 | 0013H | External Interrupt 1 |
| GSCTV | 0043H | GSC Transmit Valid |
| DMA1 | 0053H | DMA Channel 1 Done |
| TF1 | 001BH | Timer 1 Overflow |
| GSCTE | 004BH | GSC Transmit Error |
| TI+RI | 0023H | UART Transmit/Receive |

Note that the locations of the basic 8051 interrupts are the same as in the rest of the MCS-51 Family. And relative to each other they retain their same positions in the polling sequence.

The locations of the new interrupts all follow the locations of the basic 8051 interrupts in Program Memory, but they are interleaved with them in the polling sequence.

To support the new interrupts a second Interrupt Enable register and a second Interrupt Priority register are implemented in bit-addressable SFR space. The two Interrupt Enable registers in the 8XC152 are as follows:

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| IE: | EA | — | — | ES | ET1 | EX1 | ET0 | EX0 |

Address of IE in SFR space = 0A8H (bit-addressable)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| IEN1: | — | — | EGSTE | EDMA1 | EGSTV | EDMA0 | EGSRE | EGSRV |

Address pF IE1 in SFR space = 0C8H (bit-addressable)

The bits in IE are unchanged from the standard 8051 IE register. The bits in IEN1 are as follows:

EGSTE = 1 Enable GSC Transmit Error Interrupt
= 0 Disable
EDMA1 = 1 Enable DMA Channel 1 Done Interrupt
= 0 Disable
EGSTV = 1 Enable GSC Transmit Valid Interrupt
= 0 Disable
EDMA0 = 1 Enable DMA Channel 0 Done Interrupt
= 0 Disable
EGSRE = 1 Enable GSC Receive Error Interrupt
= 0 Disable
EGSRV = 1 Enable GSC Receive Valid Interrupt
= 0 Disable

The two Interrupt Priority registers in the 8XC152 are as follows:

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| IP: | — | — | — | PS | PT1 | PX1 | PT0 | PX0 |

Address of IP in SFR space = 0B8H (bit-addressable)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| IPN1: | — | — | PGSTE | PDMA1 | PGSTV | PDMA0 | PGSRE | PGSRV |

Address of IPN1 in SFR space = 0F8H (bit-addressable)

The bits in IP are unchanged from the standard 8051 IP register. The bits in IPN1 are as follows:

PGSTE = 1 GSC Transmit Error Interrupt Priority to High
= 0 Priority to Low
PDMA1 = 1 DMA Channel 1 Done Interrupt Priority to High
= 0 Priority to Low
PGSTV = 1 GSC Transmit Valid Interrupt Priority to High
= 0 Priority to Low
PDMA0 = 1 DMA Channel 0 Done Interrupt Priority to High
= 0 Priority to Low
PGSRE = 1 GSC Receive Error Interrupt Priority to High
= 0 Priority to Low
PGSRV = 1 GSC Receive Valid Interrupt Priority to High
= 0 Priority to Low

Note that these registers all have unimplemented bits ("—"). If these bits are read, they will return unpredictable values. If they are written to, the value written goes nowhere.

It is recommended that user software should never write 1s to unimplemented bits in MCS-51 devices. Future versions of the device may have new bits installed in these locations. If so, their reset value will be 0. Old software that writes 1s to newly implemented bits may unexpectedly invoke new features.

The MCS-51 interrupt structure provides hardware support for only two priority levels, High and Low. With as many interrupt sources as the 8XC152 has, it may be helpful to know how to augment the priority structure in software. Any number of priority levels can be implemented in software by saving and redefining the interrupt enable registers within the interrupt service routines. The technique is described in the "MCS-51" Architectural Overview" chapter in this handbook.

## 5.1 GSC Transmitter Error Conditions

The GSC Transmitter section reports three kinds of error conditions:

TCDT — Transmitter Collision Detector

UR — Underrun in Transmit FIFO

NOACK — No Acknowledge

These bits reside in the TSTAT register. User software can read them, but only the GSC hardware can write to them. The GSC hardware will set them in response to the various error conditions that they represent. When user software sets the TEN bit, the GSC hardware will at that time clear these flags. This is the only way these flags can be cleared.

The logical OR of these three bits flags the GSC Transmit Error interrupt (GSCTE) and clears the TEN bit, as shown in Figure 5.2. Thus any detected error condition aborts the transmission. No CRC bits are transmitted. In SDLC mode, no EOF flag is generated. In CSMA/CD mode, an EOF is generated by default, since the GTXD pin is pulled to a logic 1 and held there.

The TCDT bit can get set only if the GSC is configured to CSMA/CD mode. In that case, the GSC hardware sets TCDT when a collision is detected during a transmission, and the collision was detected after TFIFO has been accessed. Also, the GSC hardware sets TCDT when a detected collision causes the TCDCNT register to overflow.

The UR bit can get set only if the DMA bit in TSTAT is set. The DMA bit being set informs the GSC hardware that TFIFO is being serviced by DMA. In that case, if the GSC goes to fetch another byte from TFIFO and finds it empty, and the byte count register of the DMA channel servicing TFIFO is not zero, it sets the UR bit.

If the DMA hardware is not being used to service TFIFO, the UR bit cannot get set. If the DMA bit is 0, then when the GSC finds TFIFO empty, it assumes that the transmission of data is complete and the transmission of CRC bits can begin.

The NOACK bit is functional only in CSMA/CD mode, and only when the HABEN bit in RSTAT is set. The HABEN bit turns on the Hardware Based Acknowledge feature, as described in Section 3.2.6. If this feature is not invoked, the NOACK bit will stay at 0.

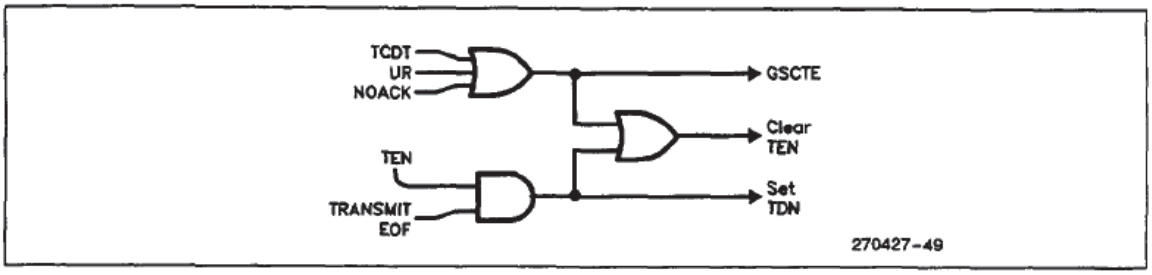



**Figure 5.2. Transmit Error Flags (Logic for Clearing TEN, Setting TDN)**

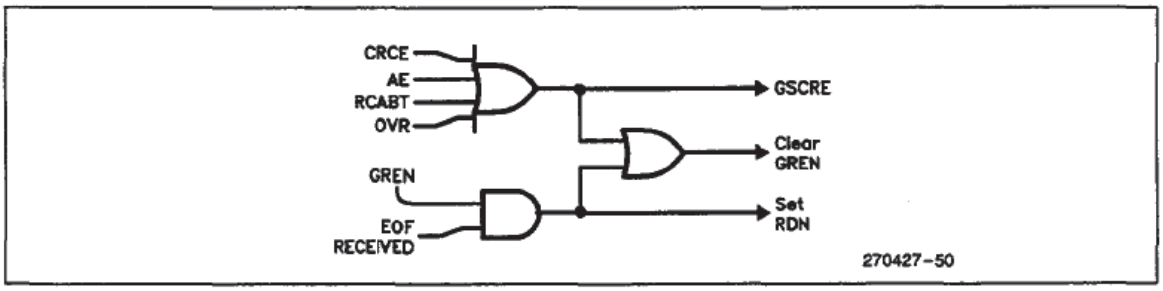



**Figure 5.3. Receive Error Flag (Logic for Clearing GREN, setting RDN)**

If the NOACK bit gets set, it means the GSC has completed a transmission, and was expecting to receive a hardware based acknowledge from the receiver of the message, but did not receive the acknowledge, or at least did not receive it cleanly. There are three ways the NOACK bit can get set:

1. The acknowledge signal (an unattached preamble) was not received before the IFS was completed.
2. A collision was detected during the IFS.
3. The line was active during the last bit-time of the IFS.

The first condition is an obvious reason for setting the NOACK bit, since that's what the hardware based acknowledge is for. The other two ways the NOACK bit can get set are to guard against the possibility that the transmitting station might mistake an unrelated transmission or transmission fragment for an acknowledge signal.

## 5.2 GSC Receiver Error Conditions

The GSC Receiver section reports four kinds of error conditions:

CRCE — CRC Error

AE — Alignment Error

RCABT — Receive Abort

OVR — Overrun in Receive FIFO

These bits reside in the RSTAT register. User software can read them, but only the GSC hardware can write to them. The GSC hardware will set them in response to the various error conditions that they represent. When user software sets the GREN bit, the GSC hardware will at that time clear these flags. This is the only way these flags can be cleared.

The logical OR of these four bits flags the GSC Receive Error interrupt (GSCRE) and clears the GREN bit, as shown in Figure 5.3. Note in this figure that any error condition will prevent RDN from being set.

A CRC Error means the CRC generator did not come to its correct value after calculating the CRC of the message plus received CRC. An Alignment Error means the number of bits received between the BOF and EOF was not a multiple of 8.

In SDLC mode, the CRCE bit gets set at the end of any frame in which there is a CRC Error, and the AE bit gets set at the end of any frame in which there is an Alignment Error.

In CSMA/CD mode, if there is no CRC Error, neither CRCE nor AE will get set. If there is a CRC Error and no Alignment Error, the CRCE bit will get set, but not the AE bit. If there is both a CRC Error and an Alignment Error, the AE bit will get set, but not the CRCE bit. Thus in CSMA/CD mode, the CRCE and AE bits are mutually exclusive.

The Receive Abort flag, RCABT, gets set if an incoming frame was interrupted after received data had already passed to the Receive FIFO. In SDLC mode, this can happen if a line idle condition is detected before an EOF flag is. In CSMA/CD mode, it can happen if there is a collision. In either case, the CPU will have to re-initialize whatever pointers and counters it might have been using.

The Overrun Error flag, OVR, gets set if the GSC Receiver is ready to push a newly received byte onto the Receive FIFO, but the FIFO is full.

Up to 7 "dribble bits" can be received after the EOF without causing an error condition.

## 6.0 GLOSSARY

ADR0,1,2,3 (95H, 0A5H, 0B5H, 0C5H) - Address Match Registers 0,1,2,3 - The contents of these SFRs are compared against the address bits from the serial data on the GSC. If the address matches the SFR, then the C152 accepts that frame. If in 8 bit addressing mode, a match with any of the four registers will trigger acceptance. In 16 bit addressing mode, a match with ADR1:ADR0 or ADR3:ADR2 will be accepted. Address length is determined by GMOD (AL).

AE - Alignment Error, see RSTAT.
AL - Address Length, see GMOD.

AMSK0,1 (0D5H, 0E5H) - Address Match Mask 0,1 - Identifies which bits in ADR0,1 are "don't care" bits. Setting a bit to 1 in AMSK0,1 identifies the corresponding bit in ADDR0,1 as not to be examined when comparing addresses.

BAUD - (94H) Contains the programmable value for the baud rate generator for the GSC. The baud rate will equal (fosc)/((BAUD+1) × 8).

BCRL0,1 (0E2H, 0F2H) - Byte Count Register Low 0,1 - Contains the lower byte of the byte count. Used during DMA transfers to identify to the DMA channels when the transfer is complete.

BCRH0,1 (0E3H, 0F3H) - Byte Count Register High 0,1 - Contains the upper byte of the byte count.

BKOFF (0C4H) - Backoff Timer - The backoff timer is an eight bit count-down timer with a clock period equal to one slot time. The backoff time is used in the CSMA/CD collision resolution algorithm.

BOF - Beginning of Frame flag - A term commonly used when dealing with packetized data. Signifies the beginning of a frame.

CRC - Cyclic Redundancy Check - An error checking routine that mathematically manipulates a value dependent on the incoming data. The purpose is to identify when a frame has been received in error.

CRCE - CRC Error, see RSTAT.

CSMA/CD - Stands for Carrier Sense, Multiple Access, with Collision Detection.

CT - CRC Type, see GMOD.

DARL0/1 (0C2H, 0D2H) - Destination Address Register Low 0/1 - Contains the lower byte of the destinations' address when performing DMA transfers.

DARH0/1 (0C3H, 0D3H) - Destination Address Register Low 0/1 - Contains the upper byte of the destinations' address when performing DMA transfers.

DAS - Destination Address Space, see DCON.

DCJ - D.C. Jam, see MYSLOT.

DCON0/1 (092H,093H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DAS | IDA | SAS | ISA | DM | TM | DONE | GO |

The DCON registers control the operation of the DMA channels by determining the source of data to be transferred, the destination of the data to be transfer, and the various modes of operation.

DCON.0 (GO) - Enables DMA Transfer - When set it enables a DMA channel. If block mode is set then DMA transfer starts as soon as possible under CPU control. If demand mode is set then DMA transfer starts when a demand is asserted and recognized.

DCON.1 (DONE) - DMA Transfer is Complete - When set the DMA transfer is complete. It is set when BCR equals 0 and is automatically reset when the DMA vectors to its interrupt routine. If DMA interrupt is disabled and the user software executes a jump on the DONE bit, then the user software must also reset the done bit. If DONE is not set, then the DMA transfer is not complete.

DCON.2 (TM) - Transfer Mode - When set, DMA burst transfers are used if the DMA channel is configured in block mode or external interrupts are used to initiate a transfer if in Demand Mode. When TM is cleared, Alternate Cycle Transfers are used if DMA is in the Block Mode, or Local Serial channel/GSC interrupts are used to initiate a transfer if in Demand Mode.

DCON.3 (DM) - DMA Channel Mode - When set, Demand Mode is used and when cleared, Block Mode is used.

DCON.4 (ISA) - Increment Source Address - When set, the source address registers are automatically incremented during each transfer. When cleared, the source address registers are not incremented.

DCON.5 (SAS) - Source Address Space - When set, the source of data for the DMA transfers is internal data memory if autoincrement is also set. If autoincrement is not set but SAS is, then the source for data will be one of the Special Function Registers. When SAS is cleared, the source for data is external data memory.

DCON.6 (IDA) - Increment Destination Address Space - When set, destination address registers are incremented once after each byte is transferred. When cleared, the destination address registers are not automatically incremented.

DCON.7 (DAS) - Destination Address Space - When set, destination of data to be transferred is internal data memory if autoincrement mode is also set. If autoincrement is not set the destinationwill be one of the Special Function Registers. When DAS is cleared then the destination is external data memory.

DCR - Deterministic Resolution, see MYSLOT.

$\overline{\text{DEN}}$ - An alternate function of one of the port 1 pins (P1.2). Its purpose is to enable external drivers when the GSC is transmitting data. This function is always active when using the GSC and if P1.2 is programmed to a 1.

DM - DMA Mode, see DCON0.

DMA - Direct Memory Access mode, see TSTAT.

DONE - DMA done bit, see DCON0.

DPH - Data Pointer High, an SFR that contains the high order byte of a general purpose pointer called the data pointer (DPTR).

DPL - Data Pointer Low, an SFR that contains the low order byte of the data pointer.

EDMA0 - Enable DMA Channel 0 interrupt, see IEN1.

EDMA1 - Enable DMA Channel 1 interrupt, see IEN1.

EGSRE - Enable GSC Receive Error interrupt, see IEN1.

EGSRV - Enable GSC Receive Valid interrupt, see IEN1.

EGSTE - Enable GSC Transmit Error interrupt, see IEN1.

EGSTV - Enable GSC Transmit Valid interrupt, see IEN1.

EOF - A general term used in serial communications. EOF stands for End Of Frame and signifies when the last bits of data are transmitted when using packetized data.

ES - Enable LSC Service interrupt, see IE.

ET0 - Enable Timer 0 interrupt, see IE.

ET1 - Enable Timer 1 interrupt, see IE.

EX0 - Enable External interrupt 0, see IE.

EX1 - Enable External interrupt 1, see IE.

GMOD (84H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| XTCLK | M1 | M0 | AL | CT | PL1 | PL0 | PR |

The bits in this SFR, perform most of the configuration on the type of data transfers to be used with the GSC. Determines the mode, address length, preamble length, protocol select, and enables the external clocking of the transmit data.

GMOD.0 (PR) - Protocol - If set, SDLC protocols with NRZI encoding, zero bit insertion, and SDLC flags are used. If cleared, CSMA/CD link access with Manchester encoding is used.

GMOD.1,2 (PL0,1) - Preamble length

| PL1 | PL0 | LENGTH (BITS) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 8 |
| 1 | 0 | 32 |
| 1 | 1 | 64 |

The length includes the two bit Begin Of frame (BOF) flag in CSMA/CD but does not include the SDLC flag. In SDLC mode, the BOF is an SDLC flag, otherwise it is two consecutive ones. Zero length is not compatible in CSMA/CD mode.

GMOD.3 (CT) - CRC Type - If set, 32-bit AUTODIN-II-32 is used. If cleared, 16-bit CRC-CCITT is used.

GMOD.4 (AL) - Address Length - If set, 16-bit addressing is used. If cleared, 8-bit addressing is used. In 8-bit mode, a match with any of the 4 address registers will allow that frame to be accepted (ADR0, ADR1, ADR2, ADR3). "Don't Care" bits may be masked in ADR0 and ADR1 with AMSK0 and AMSK1. In 16-bit mode, addresses are matched against "ADR1:ADR0" or "ADR3:ADR2". Again, "Don't Care" bits in ADR1:ADR0 can be masked in AMSK1:AMSK0. A received address of all ones will always be recognized in any mode.

GMOD.5, 6 (M0,M1) - Mode Select - Two test modes, an optional "alternate backoff" mode, or normal backoff can be enabled with these two bits.

| M1 | M0 | Mode |
|---|---|---|
| 0 | 0 | Normal |
| 0 | 1 | Raw Transmit |
| 1 | 0 | Raw Receive |
| 1 | 1 | Alternate Backoff |

GMOD.7 (XTCLK) - External Transmit Clock - If set an external 1X clock is used for the transmitter. If cleared the internal baud rate generator provides the

transmit clock. The input clock is applied to P1.3 ($\overline{TxC}$). The user software is responsible for setting or clearing this flag. External receive clock is enabled by setting PCON.3.

GO - DMA Go bit, see DCON0.

GRxD - GSC Receive Data input, an alternate function of one of the port 1 pins (P1.0). This pin is used as the receive input for the GSC. P1.0 must be programmed to a 1 for this function to operate.

GSC - Global Serial Channel - A high-level, multi-protocol, serial communication controller added to the 80C51BH core to accomplish high-speed transfers of packetized serial data.

GTxD - GSC Transmit Data output, an alternate function of one of the port 1 pins (P1.1). This pin is used as the transmit output for the GSC. P1.1 must be programmed to a 1 for this function to operate.

HBAEN - Hardware Based Acknowledge Enable, see RSTAT.

HLDA - Hold Acknowledge, an alternate function of one of the port 1 pins (P1.6). This pin is used to perform the "HOLD ACKNOWLEDGE" function for DMA transfers. HLDA can be an input or an output, depending on the configuration of the DMA channels. P1.6 must be programmed to a 1 for this function to operate.

HOLD - Hold, an alternate function of one of the port 1 pins (P1.5). This pin is used to perform the "HOLD" function for DMA transfers. HOLD can be an input or an output, depending on the configuration of the DMA channels. P1.5 must be programmed to a 1 for this function to operate.

IDA - Increment Destination Address, see DCON0.

IE (0A8H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EA | | | ES | ET1 | EX1 | ET0 | EX0 |

Interrupt Enable SFR, used to individually enable the Timer and Local Serial Channel interrupts. Also contains the global enable bit which must be set to a 1 to enable any interrupt to be automatically recognized by the CPU.

IE.0 (EX0) - Enables the external interrupt $\overline{INT0}$ on P3.2.

IE.1 (ET0) - Enables the Timer 0 interrupt.

IE.2 (EX1) - Enables the external interrupt $\overline{INT1}$ on P3.3.

IE.3 (ET1) - Enables the Timer 1 interrupt.

IE.4 (ES) - Enables the Local Serial Channel interrupt.

IE.7 (EA) - The global interrupt enable bit. This bit must be set to a 1 for any other interrupt to be enabled.

IEN1 - (0C8H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | EGSTE | EDMA1 | EGSTV | EDMA0 | EGSRE | EGSRV |

Interrupt enable register for DMA and GSC interrupts. A 1 in any bit position enables that interrupt.

IEN1.0 (EGSRV) - Enables the GSC valid receive interrupt.

IEN1.1 (EGSRE) - Enables the GSC receive error interrupt.

IEN1.2 (EDMA0) - Enables the DMA done interrupt for Channel 0.

IEN1.3 (EGSTV) - Enables the GSC valid transmit interrupt.

IEN1.4 (EDMA1) - Enables the DMA done interrupt for Channel 1.

IEN1.5 (EGSTE) - Enables the GSC transmit error interrupt

IFS - (0A4H) Interframe Space, determines the number of bit times separating transmitted frames in CSMA/CD and SDLC.

IP (0B8H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | PS | PT1 | PX1 | PT0 | PX0 |

Allows the user software two levels of prioritization to be assigned to each of the interrupts in IE. A 1 assigns the corresponding interrupt in IE a higher interrupt than an interrupt with a corresponding 0.

IP.0 (PX0) - Assigns the priority of external interrupt, $\overline{INT0}$.

IP.1 (PT0) - Assigns the priority of Timer 0 interrupt, T0.

IP.2 (PX1) - Assigns the priority of external interrupt, $\overline{INT1}$.

IP.3 (PT1) - Assigns the priority of Timer 1 interrupt, T1.

IP.4 (PS) - Assigns the priority of the LSC interrupt, SBUF.

IPN1 - (0F8H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | PGSTE | PDMA1 | PGSTV | PDMA0 | PGSRE | PGSRV |

Allows the user software two levels of prioritization to be assigned to each of the interrupts in IEN1. A 1 assigns the corresponding interrupt in IEN1 a higher interrupt than an interrupt with a corresponding 0.

IPN1.0 (PGSRV) - Assigns the priority of GSC receive valid interrupt.

IPN1.1 (PGSRE) - Assigns the priority of GSC error receive interrupt.

IPN1.2 (PDMA0) - Assigns the priority of DMA done interrupt for Channel 0.

IPN1.3 (PGSTV) - Assigns the priority of GSC transmit valid interrupt.

IPN1.4 (PDMA1) - Assigns the priority of DMA done interrupt for Channel 1.

IPN1.5 (PGSTE) - Assigns the priority of GSC transmit error interrupt.

ISA - Increment Source Address, see DCON0.

LNI - Line Idle, see TSTAT.

LSC - Local Serial Channel - The asynchronous serial port found on all MCS-51 devices. Uses start/stop bits and can transfer only 1 byte at a time.

M0 - One of two GSC mode bits, see TMOD.

M1 - One of two GSC mode bits, see TMOD

MYSLOT - (0F5H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DCJ | DCR | SA5 | SA4 | SA3 | SA2 | SA1 | SA0 |

Determines which type of Jam is used, which backoff algorithm is used, and the DCR slot address for the GSC.

MYSLOT.0,1,2,3,4,5 (SA0,1,2,3,4,5) - These bits determine which slot address is assigned to the C152 when using deterministic backoff during CSMA/CD operations on the GSC. Maximum slots available is 63. An address of 00H prevents that station from participating in the backoff process.

MYSLOT.6 (DCR) - Determines which collision resolution algorithm is used. If set to a 1, then the deterministic backoff is used. If cleared, then a random slot assignment is used.

MYSLOT.7 (DCJ) - Determines the type of Jam used during CSMA/CD operation when a collision occurs. If set to a 1 then a low D.C. level is used as the jam signal. If cleared, then $\overline{CRC}$ is used as the jam signal. The jam is applied for a length of time equal to the CRC length.

NOACK - No Acknowledgment error bit, see TSTAT.

NRZI - Non-Return to Zero inverted, a type of data encoding where a 0 is represented by a change in the level of the serial link. A 1 is represented by no change.

OVR - Overrun error bit, see RSTAT.

PR - Protocol select bit, see GMOD. PCON (87H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SMOD | ARB | REQ | GAREN | XRCLK | GFIEN | PD | IDL |

PCON.0 (IDL) - Idle bit, used to place the C152 into the idle power saving mode.

PCON.1 (PD) - Power Down bit, used to place the C152 into the power down power saving mode.

PCON.2 (GFIEN) - GSC Flag Idle Enable bit, when set, enables idle flags (01111110) to be generated between transmitted frames in SDLC mode.

PCON.3 (XRCLK) - External Receive Clock bit, used to enable an external clock to be used for only the receiver portion of the GSC.

PCON.4 (GAREN) - GSC Auxiliary Receive Enable bit, used to enable the GSC to receive back-to-back SDLC frames. This bit has no effect in CSMA/CD mode.

PCON.5 (REQ) - Requester mode bit, set to a 1 when C152 is to be operated as the requester station during DMA transfers.

PCON.6 (ARB) - Arbiter mode bit, set to a 1 when C152 is to be operated as the arbiter during DMA transfers.

PCON.7 (SMOD) - LSC mode bit, used to double the baud rate on the LSC.

PDMA0 - Priority bit for DMA Channel 0 interrupt, see IPN1.

PDMA1 - Priority bit for DMA Channel 1 interrupt, see IPN1.

PGSRE - Priority bit for GSC Receive Error interrupt, see IPN1.

PGSRV - Priority bit for GSC Receive Valid interrupt, see IPN1.

PGSTE - Priority bit for GSC Transmit Error interrupt, see IPN1.

PGSTV - Priority bit for GSC Transmit Valid interrupt, see IPN1.

PL0 - One of two bits that determines the Preamble Length, see GMOD.

PL1 - One of two bits that determines the Preamble Length, see GMOD.

PRBS - (0E4H) Pseudo-Random Binary Sequence, generates the pseudo-random number to be used in CSMA/CD backoff algorithms.

PS - Priority bit for the LSC service interrupt, see IP.

PT0 - Priority bit for Timer 0 interrupt, see IP.

PT1 - Priority bit for Timer 1 interrupt, see IP.

PX0 - Priority bit for External interrupt 0, see IP.

PX1 - Priority bit for External interrupt 1, see IP.

RCABT - GSC Receiver Abort error bit, see RSTAT.

RDN - GSC Receiver Done bit, see RSTAT.

GREN - GSC Receiver Enable bit, see RSTAT.

RFNE - GSC Receive FIFO Not Empty bit, see RSTAT.

RI - LSC Receive Interrupt bit, see SCON.

RFIFO - (F4H) RFIFO is a 3-byte FIFO that contains the receive data from the GSC.

RSTAT (0E8H) - Receive Status Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| OVR | RCABT | AE | CRCE | RDN | RFNE | GREN | HABEN |

RSTAT.0 (HBAEN) - Hardware Based Acknowledge Enable - If set, enables the hardware based acknowledge feature.

RSTAT.1 (GREN) - Receiver Enable - When set, the receiver is enabled to accept incoming frames. The user must clear RFIFO with software before enabling the receiver. RFIFO is cleared by reading the contents of RFIFO until RFNE = 0. After each read of RFIFO, it takes one machine cycle for the status of RFNE to be updated. Setting GREN also clears RDN, CRCE, AE, and RCABT. GREN is cleared by hardware at the end of a reception or if any receive errors are detected. The status of GREN has no effect on whether the receiver detects a collision in CSMA/CD mode as the receiver input circuitry always monitors the receive pin.

RSTAT.2 (RFNE) - Receive FIFO Not Empty - If set, indicates that the receive FIFO contains data. The receive FIFO is a three byte buffer into which the receive data is loaded. A CPU read of the FIFO retrieves the oldest data and automatically updates the FIFO pointers. Setting GREN to a one will clear the receive FIFO. The status of this flag is controlled by the GSC. This bit is cleared if user software empties receive FIFO.

RSTAT.3 (RDN) - Receive Done - If set, indicates the successful completion of a receiver operation. Will not be set if a CRC, alignment, abort, or FIFO overrun error occurred.

RSTAT.4 (CRCE) - CRC Error - If set, indicates that a properly aligned frame was received with a mismatched CRC.

RSTAT.5 (AE) - Alignment Error - In CSMA/CD mode, AE is set if the receiver shift register (an internal serial-to-parallel converter) is not full and the CRC is bad when an EOF is detected. In CSMA/CD the EOF is a line idle condition (see LNI) for two bit times. If the CRC is correct while in CSMA/CD mode, AE is not set and any mis-alignment is assumed to be caused by dribble bits as the line went idle. In SDLC mode, AE is set if a non-byte-aligned flag is received. CRCE may also be set. The setting of this flag is controlled by the GSC.

RSTAT.6 (RCABT) - Receiver Collision/Abort Detect - If set, indicates that a collision was detected after data had been loaded into the receive FIFO in CSMA/CD mode. In SDLC mode, RCABT indicates that 7 consecutive ones were detected prior to the end flag but after data has been loaded into the receive FIFO. AE may also be set if RCABT is set.

RSTAT.7 (OVR) - Overrun - If set, indicates that the receive FIFO was full and new shift register data was written into it. It is cleared by user software. AE and/or CRCE may also be set if OVR is set.

SARH0 (0A3H) - Source Address Register High 0, contains the high byte of the source address for DMA Channel 0.

SARH1 (0B3H) - Source Address Register High 1, contains the high byte of the source address for DMA Channel 1.

SARL0 (0A2H) - Source Address Register Low 0, contains the low byte of the source address for DMA Channel 0.

SARL1 (0B2H) - Source Address Register Low 1, contains the low byte of the source address for DMA Channel 1.

SAS - Source Address Space bit, see DCON0.

SBUF (099H) - Serial Buffer, both the receive and transmit SFR location for the LSC.

SCON (098H)

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

SCON.0 (RI) - Receive Interrupt flag.

SCON.1 (TI) - Transmit Interrupt flag.

SCON.2 (RB8) - Receive Bit 8, contains the ninth bit that was received in Modes 2 and 3 or the stop bit in Mode 1 if SM20. Not used in Mode 0.

SCON.3 (TB8) - Transmit Bit 8, the ninth bit to be transmitted in Modes 2 and 3.

SCON.4 (REN) - Receiver Enable, enables reception for the LSC.

SCON.5 (SM2) - Enables the multiprocessor communication feature in Modes 2 and 3 for the LSC.

SCON.6 (SM1) - LSC mode specifier.

SCON.7 (SM2) - LSC mode specifier.

SDLC - Stands for Synchronous Data Link Communication and is a protocol developed by IBM.

SLOTTM - (0B4H) Determines the length of the slot time in CSMA/CD.

SP (081H) - Stack Pointer, an eight bit pointer register used during a PUSH, POP, CALL, RET, or RETI.

TCDCNT - (0D4H) Contains the number of collisions in the current frame if using probabilistic CSMA/CD and contains the maximum number of slots in the deterministic mode.

TCDT - Transmit Collision Detect, see TSTAT.

TCON (088H)

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

TCON.0 (IT0) - Interrupt 0 mode control bit.

TCON.1 (IE0) - External interrupt 0 edge flag.

TCON.2 (IT1) - Interrupt 1 mode control bit.

TCON.3 (IE1) - External interrupt 1 edge flag.

TCON.4 (TR0) - Timer 0 run control bit.

CON.5 (TF0) - Timer 0 overflow flag.

TCON.6 (TR1) - Timer 1 run control bit.

TCON.7 (TF1) - Timer 1 overflow flag.

TDN - Transmit Done flag, see TSTAT.

TEN - Transmit Enable bit, see TSTAT.

TFNF - Transmit FIFO Not Full flag, see TSTAT.

TFIFO - (85H) TFIFO is a 3-byte FIFO that contains the transmission data for the GSC.

TH0 (08CH) - Timer 0 High byte, contains the high byte for timer/counter 0.

TH1 (08DH) - Timer 1 High byte, contains the high byte for timer/counter 1.

TI - Transmit Interrupt, see SCON.

TL0 (08AH) - Timer 0 Low byte, contains the low byte for timer/counter 0.

TL1 (08BH) - Timer 1 Low byte, contains the low byte for timer/counter 1.

TM - Transfer Mode, see, DCON0.

TMOD (089H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| GATE | C/$\overline{T}$ | M1 | M0 | GATE | C/$\overline{T}$ | M1 | M0 |

TMOD.0 (M0) - Mode selector bit for Timer 0.

TMOD.1 (M1) - Mode selector bit for Timer 0.

TMOD.2 (C/$\overline{T}$) - Timer/Counter selector bit for Timer 0.

TMOD.3 (GATE) - Gating Mode bit for Timer 0.

TMOD.4 (M0) - Mode selector bit for Timer 1.

TMOD.5 (M1) - Mode selector bit for Timer 1.

TMOD.6 (C/$\overline{T}$) - Timer/Counter selector bit for Timer 1.

TMOD.7 (GATE) - Gating Mode bit for Timer 1.

TSTAT (0D8) - Transmit Status Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| LNI | NOACK | UR | TCDT | TDN | TFNF | TEN | DMA |

TSTAT.0 (DMA) - DMA Select - If set, indicates that DMA channels are used to service the GSC FIFO's and GSC interrupts occur on TDN and RDN, and also enables UR to become set. If cleared, indicates that the GSC is operating in it normal mode and interrupts occur on TFNE and RFNE.For more information on DMA servicing please refer to the DMA section on DMA serial demand mode (4.2.2.3).

TSTAT.1 (TEN) - Transmit Enable - When set causes TDN, UR, TCDT, and NOACK flags to be reset and the TFIFO cleared. The transmitter will clear TEN after a successful transmission, a collision during the data, CRC, or end flag. If cleared during a transmission the GSC transmit pin goes to a steady state high level. This is the method used to send an abort character in SDLC. Also $\overline{DEN}$ is forced to a high level. The end of transmission occurs whenever the TFIFO is emptied.

TSTAT.2 (TFNF) - Transmit FIFO not full - When set, indicates that new data may be written into the transmit FIFO. The transmit FIFO is a three byte buffer that loads the transmit shift register with data.

TSTAT.3 (TDN) - Transmit Done - When set, indicates the successful completion of a frame transmission. If HBAEN is set, TDN will not be set until the end of the IFS following the transmitted message, so that the acknowledge can be checked. If an acknowledge is expected and not received, TDN is not set. An acknowledge is not expected following a broadcast or multi-cast packet.

TSTAT.4 (TCDT) - Transmit Collision Detect - If set, indicates that the transmitter halted due to a collision. It is set if a collision occurs during the data or CRC or if there are more than eight collisions.

TSTAT.5 (UR) - Underrun - If set, indicates that in DMA mode the last bit was shifted out of the transmit register and that the DMA byte count did not equal zero. When an underrun occurs, the transmitter halts without sending the CRC or the end flag.

TSTAT.6 (NOACK) - No Acknowledge - If set, indicates that no acknowledge was received for the previous frame. Will be set only if HBAEN is set and no acknowledge is received prior to the end of the IFS. NOACK is not set following a broadcast or a multicast packet.

TSTAT.7 (LNI) - Line Idle - If set, indicates the receive line is idle. In SDLC protocol it is set if 15 consecutive ones are received. In CSMA/CD protocol, line idle is set if GR×D remains high for approximately 1.6 bit times. LNI is cleared after a transition on GR×D.

TxC - External Clock input for GSC transmitter.

UR - Underrun flag, see TSTAT.

XRCLK - External GSC Receive Clock Enable bit, see PCON.

XTCLK - External GSC Transmit Clock Enable bit, see GMOD.