Microprocessors and Microcontrollers

Microprocessors and microcontrollers stem from the same basic idea, are made by the same people, and are sold to the same types of system designers and programmers. What is the difference between the two?

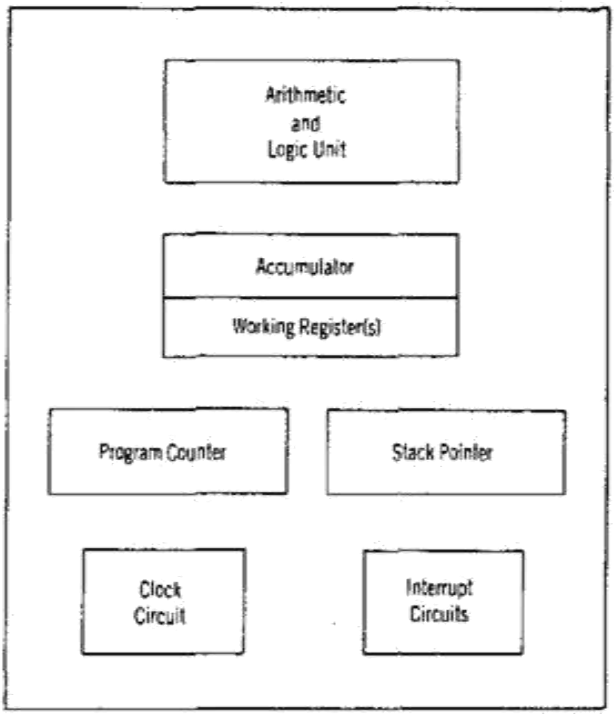
Microprocessors

A *microprocessor*, as the term has come to be known, is a general-purpose digital computer central processing unit (CPU). Although popularly known as a "computer on a chip," the microprocessor is in no sense a complete digital computer.

Figure 1.1 shows a block diagram of a microprocessor CPU, which contains an arithmetic and logic unit (ALU), a program counter (PC), a stack pointer (SP), some working registers, a clock timing circuit, and interrupt circuits.

To make a complete microcomputer, one must add memory, usually read-only program memory (ROM) and random-access data memory (RAM), memory decoders, an oscillator, and a number of input/output (I/O) devices, such as parallel and serial data ports. Additionally, special-purpose devices, such as interrupt handlers, or counters, may

FIGURE 1.1 A Block Diagram of a Microprocessor



be added to relieve the CPU from time-consuming counting or timing chores. Equipping the microcomputer with a mass storage device, commonly a floppy disk drive, and I/O peripherals, such as a keyboard and a CRT display, yields a small computer that can be applied to a range of general-purpose software applications.

The key term in describing the design of the microprocessor is *"general-purpose."* The hardware design of a microprocessor CPU is arranged so that a small, or very large, system can be configured around the CPU as the application demands. The internal CPU architecture, as well as the resultant machine level code that operates that architecture, is comprehensive but as flexible as possible.

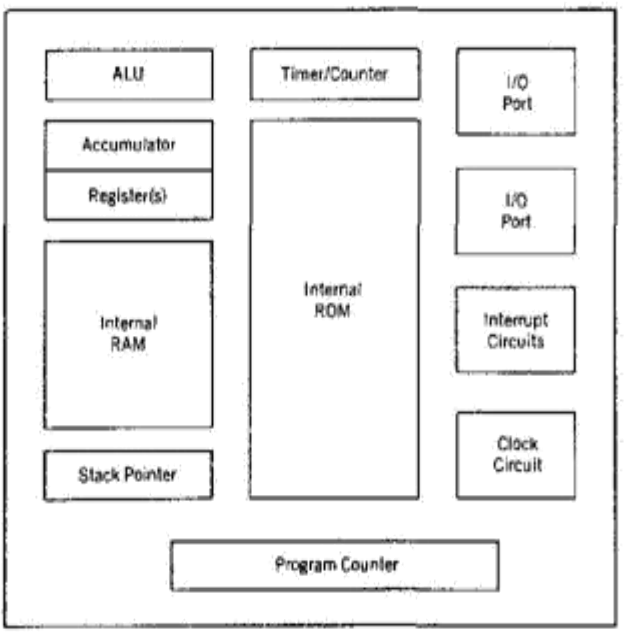
The prime use of a microprocessor is to fetch data, perform extensive calculations on that data, and store those calculations in a mass storage device or display the results for human use. The programs used by the microprocessor are stored in the mass storage device and loaded into RAM as the user directs. A few microprocessor programs are stored in ROM. The ROM-based programs are primarily small fixed programs that operate peripherals and other fixed devices that are connected to the system. The design of the microprocessor is driven by the desire to make it as expandable as possible, in the expectation of commercial success in the marketplace.

Microcontrollers

Figure 1.2 shows the block diagram of a typical *microcontroller*, which is a true computer on a chip. The design incorporates all of the features found in a microprocessor CPU: ALU, PC, SP, and registers. It also has added the other features needed to make a complete computer: ROM, RAM, parallel I/O, serial I/O, counters, and a clock circuit.

Like the microprocessor, a microcontroller is a general-purpose device, but one which is meant to fetch data, perform limited calculations on that data, and control its

FIGURE 1.2 A Block Diagram of a Microcontroller

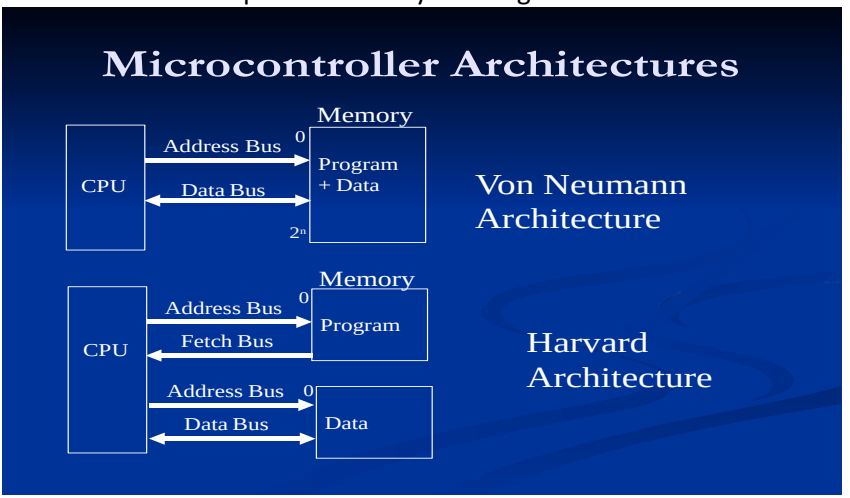


11

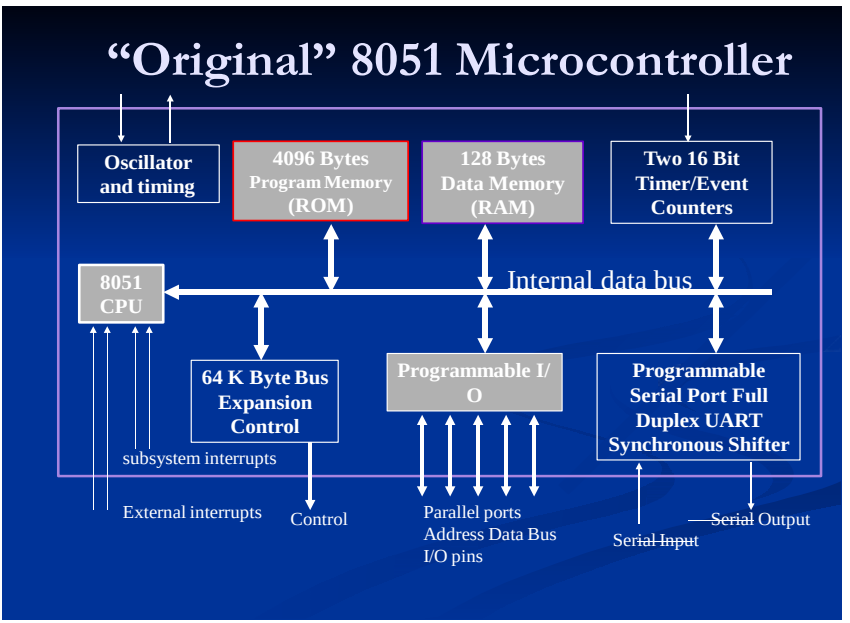
The 8051 microcontroller actually includes a whole family of microcontrollers that have numbers ranging from 8031 to 8751 and are available in N-Channel Metal Oxide Silicon (NMOS) and Complementary Metal Oxide Silicon (CMOS) construction in a variety of

Two type of Architecture

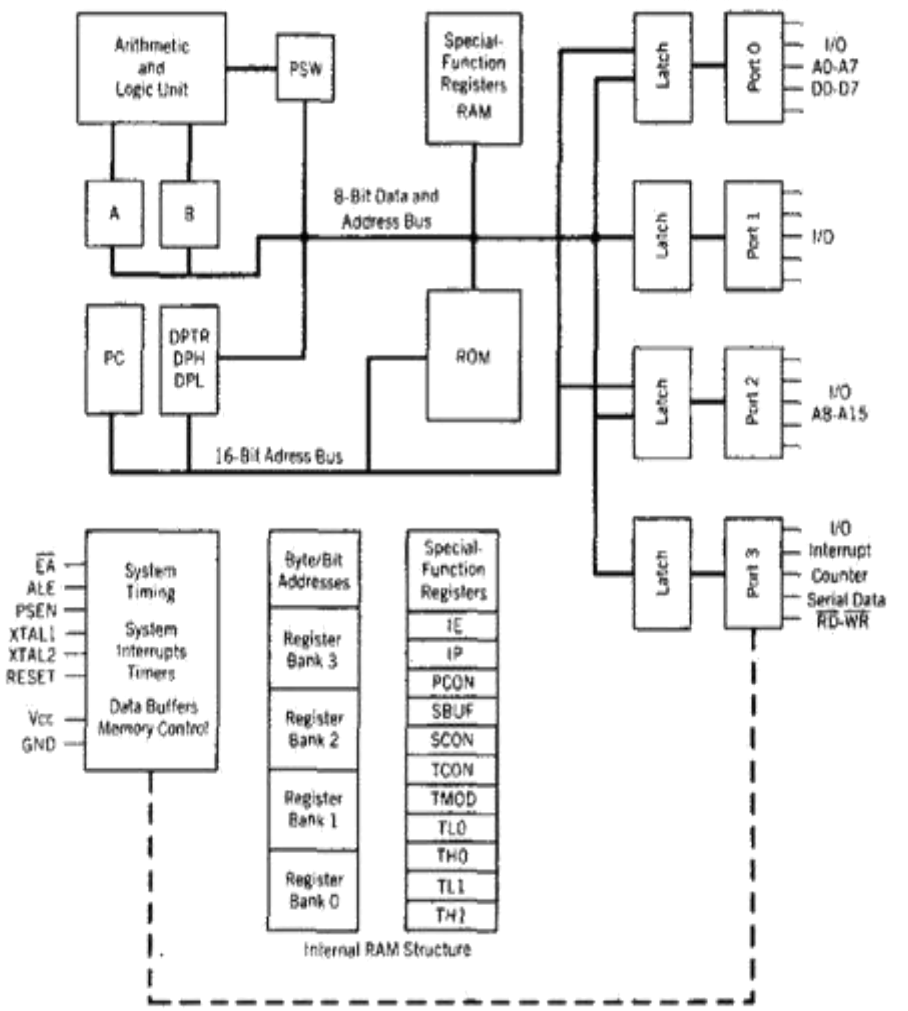
1. Von Neumann: Have same memory for Program & Data
2. Harvard: Separate memory for Program and Data



8051 Architecture



8051 Block Diagram



package types. An enhanced version of the 8051, the 8052, also exists with its own family of variations and even includes one member that can be programmed in BASIC. This galaxy of parts, the result of desires by the manufacturers to leave no market niche unfilled, would require many chapters to cover. In this chapter, we will study a "generic" 8051, housed in a 40-pin DIP, and direct the investigation of a particular type to the data books. The block diagram of the 8051 in Figure 2.1a shows all of the features unique to microcontrollers:

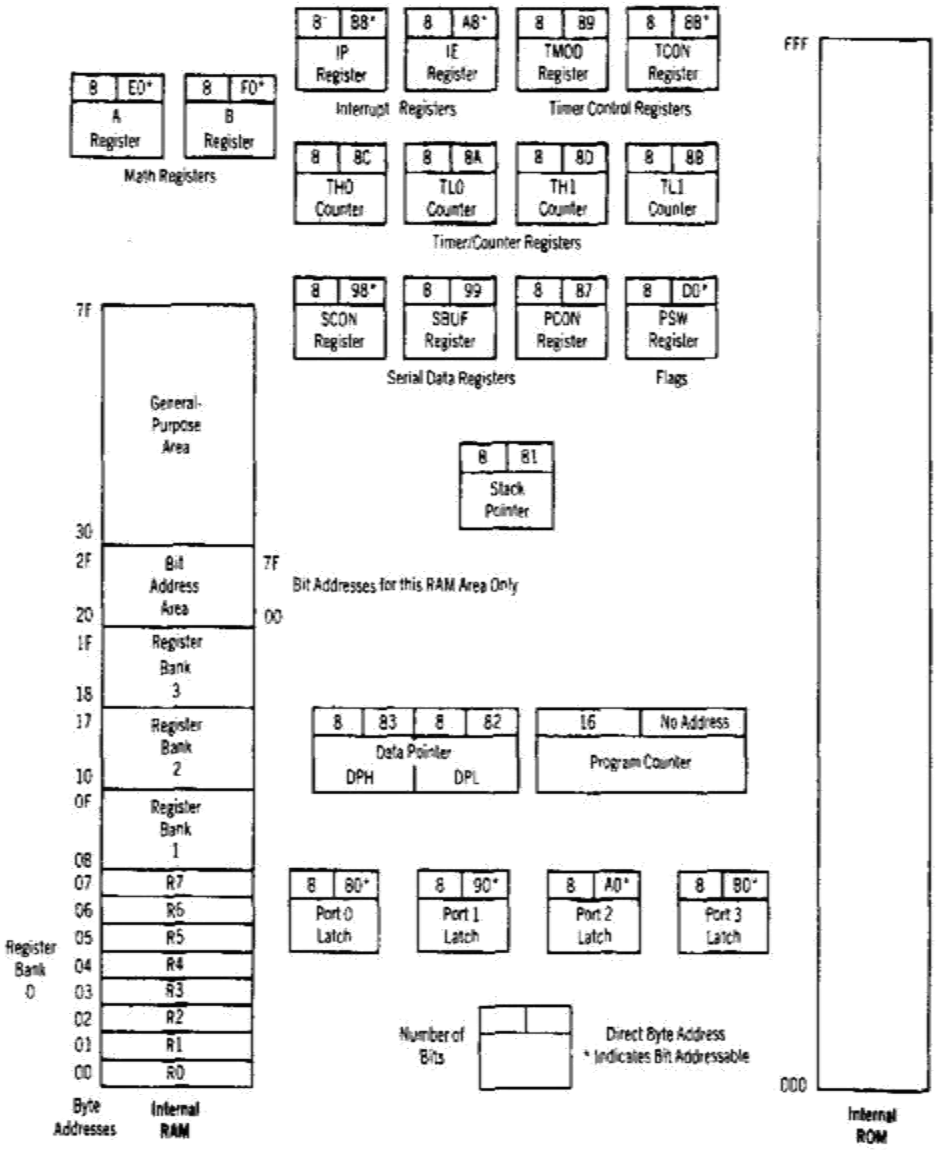
Internal ROM and RAM

I/O ports with programmable pins

Timers and counters

Serial data communication

FIGURE 2.1b 8051 Programming Model



The figure also shows the usual CPU components: program counter, ALU, working registers, and clock circuits.[1]

The 8051 architecture consists of these specific features:

- Eight-bit CPU with registers A (the accumulator) and B
- Sixteen-bit program counter (PC) and data pointer (DPTR)
- Eight-bit program status word (PSW)
- Eight-bit stack pointer (SP)
- Internal ROM or EPROM (8751) of 0 (8031) to 4K (8051)
- Internal RAM of 128 bytes:
  - Four register banks, each containing eight registers
  - Sixteen bytes, which may be addressed at the bit level
  - Eighty bytes of general-purpose data memory
- Thirty-two input/output pins arranged as four 8-bit ports: P0–P3
- Two 16-bit timer/counters: T0 and T1
- Full duplex serial data receiver/transmitter: SBUF
- Control registers: TCON, TMOD, SCON, PCON, IP, and IE
- Two external and three internal interrupt sources
- Oscillator and clock circuits

The programming model of the 8051 in Figure 2.1b shows the 8051 as a collection of 8- and 16-bit registers and 8-bit memory locations. These registers and memory locations can be made to operate using the software instructions that are incorporated as part of the design. The program instructions have to do with the control of the registers and digital data paths that are physically contained inside the 8051, as well as memory locations that are physically located outside the 8051.

The model is complicated by the number of special-purpose registers that must be present to make a microcomputer a microcontroller. A cursory inspection of the model is recommended for the first-time viewer; return to the model as needed while progressing through the remainder of the text.

Most of the registers have a specific function; those that do occupy an individual block with a symbolic name, such as A or TH0 or PC. Others, which are generally indistinguishable from each other, are grouped in a larger block, such as internal ROM or RAM memory.

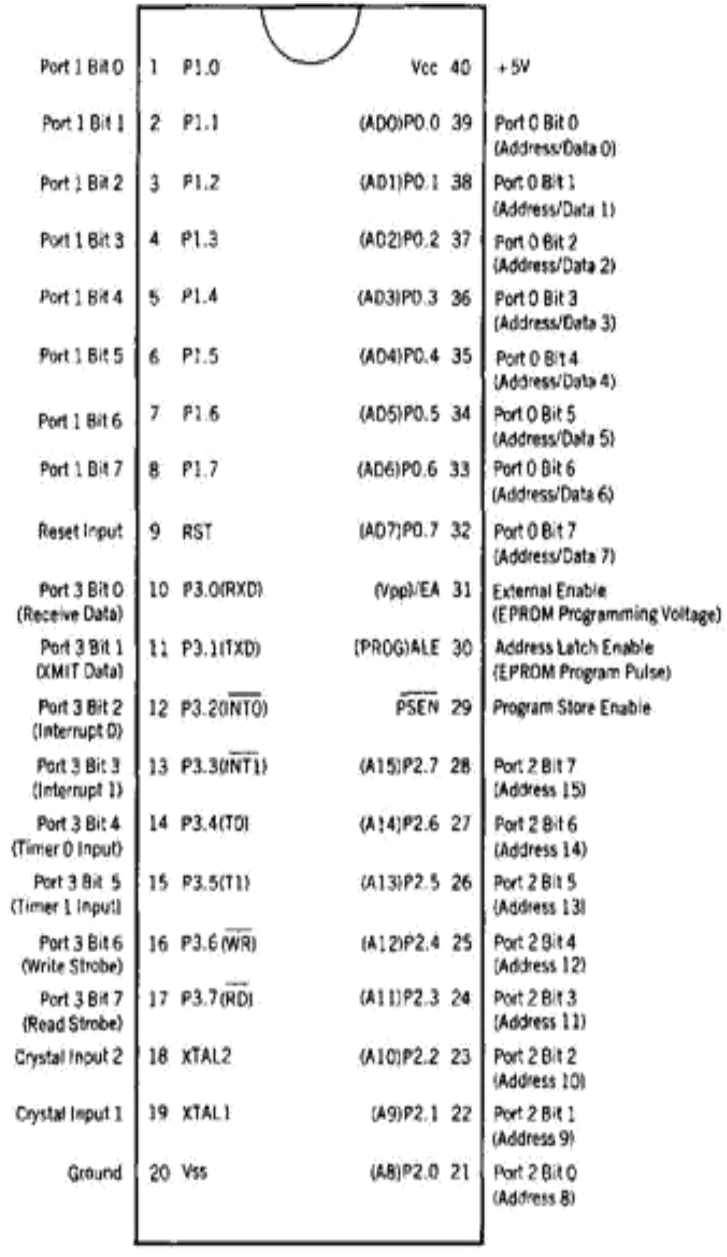
Each register, with the exception of the program counter, has an internal 1-byte address assigned to it. Some registers (marked with an asterisk * in Figure 2.1b) are both byte and bit addressable. That is, the entire byte of data at such register addresses may be read or altered, or individual bits may be read or altered. Software instructions are generally able to specify a register by its address, its symbolic name, or both.

A pinout of the 8051 packaged in a 40-pin DIP is shown in Figure 2.2 with the full and abbreviated names of the signals for each pin. It is important to note that many of the

---

[1] Knowledge of the details of circuit operation that cannot be affected by any instruction or external data, while intellectually stimulating, tends to confuse the student new to the 8051. For this reason, this text will concentrate on the essential features of the 8051; the more advanced student may wish to refer to manufacturers' data books for additional information.

FIGURE 2.2 8051 DIP Pin Assignments



Note: Alternate functions are shown below the port name (in parentheses). Pin numbers and pin names are shown inside the DIP package.

pins are used for more than one function (the alternate functions are shown in parentheses in Figure 2.2). Not all of the possible 8051 features may be used *at the same time*.

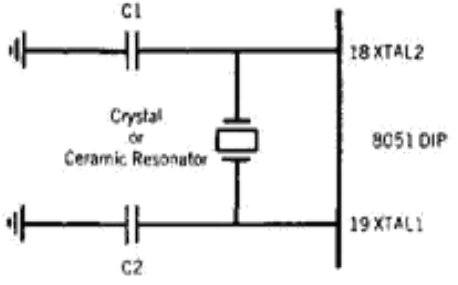
Programming instructions or physical pin connections determine the use of any multifunction pins. For example, port 3 bit 0 (abbreviated P3.0) may be used as a general-purpose I/O pin, or as an input (RXD) to SBUF, the serial data receiver register. The system designer decides which of these two functions is to be used and designs the hardware and software affecting that pin accordingly.

The 8051 Oscillator and Clock

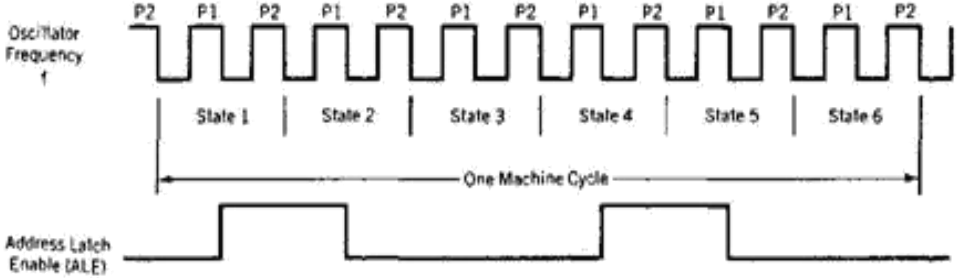
The heart of the 8051 is the circuitry that generates the clock pulses by which all internal operations are synchronized. Pins XTAL1 and XTAL2 are provided for connecting a resonant network to form an oscillator. Typically, a quartz crystal and capacitors are employed, as shown in Figure 2.3. The crystal frequency is the basic internal clock frequency of the microcontroller. The manufacturers make available 8051 designs that can run at specified maximum and *minimum* frequencies, typically 1 megahertz to 16 megahertz. Minimum frequencies imply that some internal memories are dynamic and must always operate above a minimum frequency, or data will be lost.

Communication needs often dictate the frequency of the oscillator due to the requirement that internal counters must divide the basic clock rate to yield standard communication bit per second (baud) rates. If the basic clock frequency is not divisible without a remainder, then the resulting communication frequency is not standard.

FIGURE 2.3 Oscillator Circuit and Timing



Crystal or Ceramic Resonator Oscillator Circuit



8051 Timing

Ceramic resonators may be used as a low-cost alternative to crystal resonators. However, decreases in frequency stability and accuracy make the ceramic resonator a poor choice if high-speed serial data communication with other systems, or critical timing, is to be done.

The oscillator formed by the crystal, capacitors, and an on-chip inverter generates a pulse train at the frequency of the crystal, as shown in Figure 2.3.

The clock frequency, *f*, establishes the smallest interval of time within the microcontroller, called the pulse, *P*, time. The smallest interval of time to accomplish any simple instruction, or part of a complex instruction, however, is the machine cycle. The machine cycle is itself made up of six states. A state is the basic time interval for discrete operations of the microcontroller such as fetching an opcode byte, decoding an opcode, executing an opcode, or writing a data byte. Two oscillator pulses define each state.

Program instructions may require one, two, or four machine cycles to be executed, depending on the type of instruction. Instructions are fetched and executed by the microcontroller automatically, beginning with the instruction located at ROM memory address 0000h at the time the microcontroller is first reset.

To calculate the time any particular instruction will take to be executed, find the number of cycles, *C*, from the list in Appendix A. The time to execute that instruction is then found by multiplying *C* by 12 and dividing the product by the crystal frequency:

$$\text{Tinst} = \frac{C \times 12\text{d}}{\text{crystal frequency}}$$

For example, if the crystal frequency is 16 megahertz, then the time to execute an ADD A, R1 one-cycle instruction is .75 microseconds. A 12 megahertz crystal yields the convenient time of one microsecond per cycle. An 11.0592 megahertz crystal, while seemingly an odd value, yields a cycle frequency of 921.6 kilohertz, which can be divided evenly by the standard communication baud rates of 19200, 9600, 4800, 2400, 1200, and 300 hertz.

Program Counter and Data Pointer

The 8051 contains two 16-bit registers: the program counter (PC) and the data pointer (DPTR). Each is used to hold the address of a byte in memory.

Program instruction bytes are fetched from locations in memory that are addressed by the PC. Program ROM may be on the chip at addresses 0000h to 0FFFh, external to the chip for addresses that exceed 0FFFh, or totally external for all addresses from 0000h to FFFFh. The PC is automatically incremented after every instruction byte is fetched and may also be altered by certain instructions. The PC is the only register that does not have an internal address.

The DPTR register is made up of two 8-bit registers, named DPH and DPL, that are used to furnish memory addresses for internal and external code access and external data access. The DPTR is under the control of program instructions and can be specified by its 16-bit name, DPTR, or by each individual byte name, DPH and DPL. DPTR does not have a single internal address; DPH and DPL are each assigned an address.

A and B CPU Registers

The 8051 contains 34 general-purpose, or working, registers. Two of these, registers A and B, comprise the mathematical core of the 8051 central processing unit (CPU). The other 32 are arranged as part of internal RAM in four banks, B0–B3, of eight registers each, named R0 to R7.

The A (accumulator) register is the most versatile of the two CPU registers and is used for many operations, including addition, subtraction, integer multiplication and division, and Boolean bit manipulations. The A register is also used for all data transfers between the 8051 and any external memory. The B register is used with the A register for multiplication and division operations and has no other function other than as a location where data may be stored.

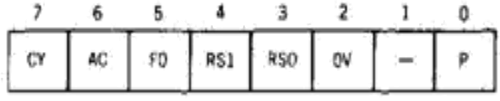
Flags and the Program Status Word (PSW)

Flags are 1-bit registers provided to store the results of certain program instructions. Other instructions can test the condition of the flags and make decisions based upon the flag states. In order that the flags may be conveniently addressed, they are grouped inside the program status word (PSW) and the power control (PCON) registers.

The 8051 has four math flags that respond automatically to the outcomes of math operations and three general-purpose user flags that can be set to 1 or cleared to 0 by the programmer as desired. The math flags include carry (C), auxiliary carry (AC), overflow (OV), and parity (P). User flags are named F0, GF0, and GF1; they are general-purpose flags that may be used by the programmer to record some event in the program. Note that all of the flags can be set and cleared by the programmer at will. The math flags, however, are also affected by math operations.

The program status word is shown in Figure 2.4. The PSW contains the math flags, user program flag F0, and the register select bits that identify which of the four general-purpose register banks is currently in use by the program. The remaining two user flags, GF0 and GF1, are stored in PCON, which is shown in Figure 2.13.

FIGURE 2.4 PSW Program Status Word Register



THE PROGRAM STATUS WORD (PSW) SPECIAL FUNCTION REGISTER

Bit	Symbol	Function
7	CY	Carry flag; used in arithmetic, JUMP, ROTATE, and BOOLEAN instructions
6	AC	Auxiliary carry flag; used for BCD arithmetic
5	F0	User flag 0
4	RS1	Register bank select bit 1
3	RS0	Register bank select bit 0

RS1	RS0	
0	0	Select register bank 0
0	1	Select register bank 1
1	0	Select register bank 2
1	1	Select register bank 3

2	OV	Overflow flag; used in arithmetic instructions
1	—	Reserved for future use
0	P	Parity flag; shows parity of register A: 1 = Odd Parity

Bit addressable as PSW.0 to PSW.7

Detailed descriptions of the math flag operations will be discussed in chapters that cover the opcodes that affect the flags. The user flags can be set or cleared using data move instructions covered in Chapter 3.

Internal Memory

A functioning computer must have memory for program code bytes, commonly in ROM, and RAM memory for variable data that can be altered as the program runs. The 8051 has internal RAM and ROM memory for these functions. Additional memory can be added externally using suitable circuits.

Unlike microcontrollers with Von Neumann architectures, which can use a *single* memory address for either program code or data, *but not for both*, the 8051 has a Harvard architecture, which uses the *same address*, in *different* memories, for code and data. Internal circuitry accesses the correct memory based upon the nature of the operation in progress.

Internal RAM

The 128-byte internal RAM, which is shown generally in Figure 2.1 and in detail in Figure 2.5, is organized into three distinct areas:

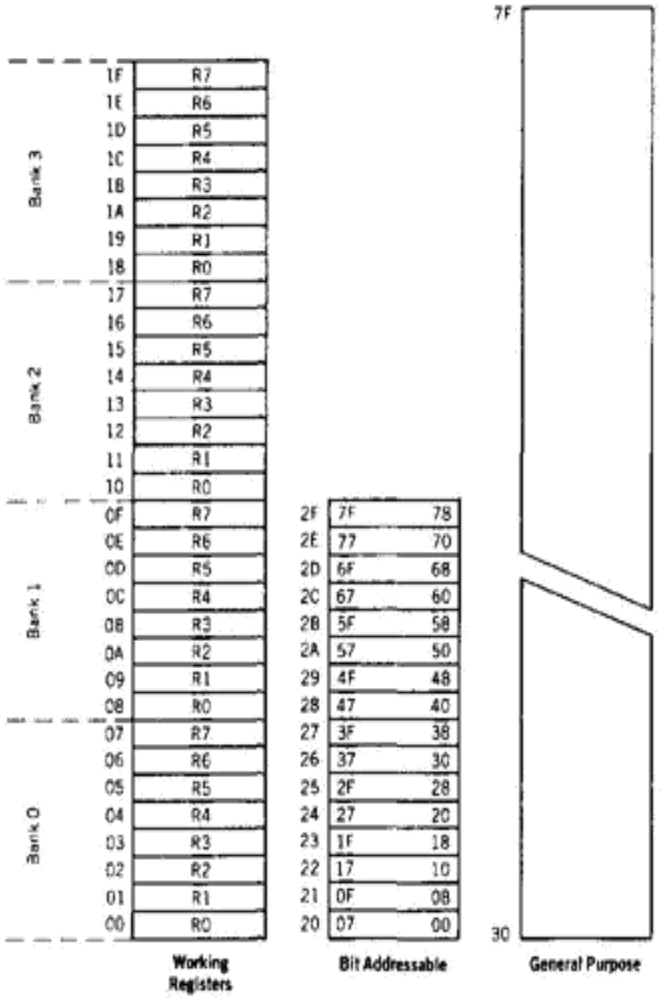
1. Thirty-two bytes from address 00h to 1Fh that make up 32 working registers organized as four banks of eight registers each. The four register banks are numbered 0 to 3 and are made up of eight registers named R0 to R7. Each register can be addressed by name (when its bank is selected) or by its RAM address. Thus R0 of bank 3 is R0 (if bank 3 is currently selected) or address 18h (whether bank 3 is selected or not). Bits RS0 and RS1 in the PSW determine which bank of registers is currently in use at any time when the program is running. Register banks not selected can be used as general-purpose RAM. Bank 0 is selected upon reset.
2. A *bit*-addressable area of 16 bytes occupies RAM *byte* addresses 20h to 2Fh, forming a total of 128 addressable bits. An addressable bit may be specified by its *bit* address of 00h to 7Fh, or 8 bits may form any *byte* address from 20h to 2Fh. Thus, for example, bit address 4Fh is also bit 7 of byte address 29h. Addressable bits are useful when the program need only remember a binary event (switch on, light off, etc.). Internal RAM is in short supply as it is, so why use a byte when a bit will do?
3. A general-purpose RAM area above the bit area, from 30h to 7Fh, addressable as bytes.

The Stack and the Stack Pointer

The stack refers to an area of internal RAM that is used in conjunction with certain opcodes to store and retrieve data quickly. The 8-bit stack pointer (SP) register is used by the 8051 to hold an internal RAM address that is called the "top of the stack." The address held in the SP register is the location in internal RAM where the last byte of data was stored by a stack operation.

When data is to be placed on the stack, the SP increments *before* storing data on the stack so that the stack grows *up* as data is stored. As data is retrieved from the stack, the byte is read from the stack, and then the SP decrements to point to the next available byte of stored data.

FIGURE 2.5 Internal RAM Organization



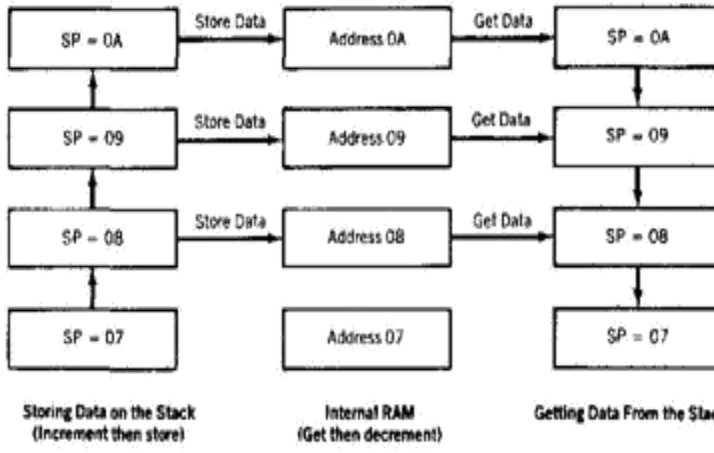
Note: Byte addresses are shown to the left; bit addresses registers are shown inside a location.

Operation of the stack and the SP is shown in Figure 2.6. The SP is set to 07h when the 8051 is reset and can be changed to any internal RAM address by the programmer.

The stack is limited in height to the size of the internal RAM. The stack has the potential (if the programmer is not careful to limit its growth) to overwrite valuable data in the register banks, bit-addressable RAM, and scratch-pad RAM areas. The programmer is responsible for making sure the stack does not grow beyond pre-defined bounds!

The stack is normally placed high in internal RAM, by an appropriate choice of the number placed in the SP register, to avoid conflict with the register, bit, and scratch-pad internal RAM areas.

FIGURE 2.6 Stack Operation



Special Function Registers

The 8051 operations that do not use the internal 128-byte RAM addresses from 00h to 7Fh are done by a group of specific internal registers, each called a special-function register (SFR), which may be addressed much like internal RAM, using addresses from 80h to FFh.

Some SFRs (marked with an asterisk * in Figure 2.1b) are also bit addressable, as is the case for the bit area of RAM. This feature allows the programmer to change only what needs to be altered, leaving the remaining bits in that SFR unchanged.

Not all of the addresses from 80h to FFh are used for SFRs, and attempting to use an address that is not defined, or "empty," results in unpredictable results. In Figure 2.1b, the SFR addresses are shown in the upper right corner of each block. The SFR names and equivalent internal RAM addresses are given in the following table:

NAME	FUNCTION	INTERNAL RAM ADDRESS (HEX)
A	Accumulator	0E0
B	Arithmetic	0F0
DPH	Addressing external memory	83
DPL	Addressing external memory	82
IE	Interrupt enable control	0A8
IP	Interrupt priority	0B8
P0	Input/output port latch	80
P1	Input/output port latch	90
P2	Input/output port latch	A0
P3	Input/output port latch	0B0
PCON	Power control	87
PSW	Program status word	0D0
SCON	Serial port control	98
SBUF	Serial port data buffer	99

Continued

NAME	FUNCTION	INTERNAL RAM ADDRESS (HEX)
		Continued
SP	Stack pointer	81
TMOD	Timer/counter mode control	89
TCON	Timer/counter control	88
TL0	Timer 0 low byte	8A
TH0	Timer 0 high byte	8C
TL1	Timer 1 low byte	8B
TH1	Timer 1 high byte	8D

Note that the PC is not part of the SFR and has no internal RAM address.

SFRs are named in certain opcodes by their functional names, such as A or TH0, and are referenced by other opcodes by their addresses, such as 0E0h or 8Ch. Note that *any* address used in the program *must* start with a number; thus address E0h for the A SFR begins with 0. Failure to use this number convention will result in an assembler error when the program is assembled.

Internal ROM

The 8051 is organized so that data memory and program code memory can be in two entirely different physical memory entities. *Each* has the same address ranges.

The structure of the internal RAM has been discussed previously. A corresponding block of internal program code, contained in an internal ROM, occupies code address space 0000h to 0FFFh. The PC is ordinarily used to address program code bytes from addresses 0000h to FFFFh. Program addresses higher than 0FFFh, which exceed the internal ROM capacity, will cause the 8051 to automatically fetch code bytes from external program memory. Code bytes can also be fetched exclusively from an external memory, addresses 0000h to FFFFh, by connecting the external access pin ($\overline{EA}$ pin 31 on the DIP) to ground. The PC does not care where the code is; the circuit designer decides whether the code is found totally in internal ROM, totally in external ROM, or in a combination of internal and external ROM.

Input/Output Pins, Ports, and Circuits

One major feature of a microcontroller is the versatility built into the input/output (I/O) circuits that connect the 8051 to the outside world. As noted in Chapter 1, microprocessor designs must add additional chips to interface with external circuitry; this ability is built into the microcontroller.

To be commercially viable, the 8051 had to incorporate as many functions as were technically and economically feasible. The main constraint that limits numerous functions is the number of pins available to the 8051 circuit designers. The DIP has 40 pins, and the success of the design in the marketplace was determined by the flexibility built into the use of these pins.

For this reason, 24 of the pins may each be used for one of two entirely different functions, yielding a total pin configuration of 64. The function a pin performs at any given instant depends, first, upon what is physically connected to it and, then, upon what software commands are used to "program" the pin. Both of these factors are under the complete control of the 8051 programmer and circuit designer.

Given this pin flexibility, the 8051 may be applied simply as a single component with I/O only, or it may be expanded to include additional memory, parallel ports, and serial data communication by using the alternate pin assignments. The key to programming an alternate pin function is the port pin circuitry shown in Figure 2.7.

Each port has a D-type output latch for each pin. The SFR for each port is made up of these eight latches, which can be addressed at the SFR address for that port. For instance, the eight latches for port 0 are addressed at location 80h; port 0 pin 3 is bit 2 of the P0 SFR. The port latches should not be confused with the port pins; the data on the latches does *not* have to be the same as that on the pins.

The two data paths are shown in Figure 2.7 by the circuits that read the latch or pin data using two entirely separate buffers. The top buffer is enabled when latch data is read, and the lower buffer, when the pin state is read. The status of each latch may be read from a latch buffer, while an input buffer is connected directly to each pin so that the pin status may be read independently of the latch state.

Different opcodes access the latch or pin states as appropriate. Port operations are determined by the manner in which the 8051 is connected to external circuitry.

Programmable port pins have completely different alternate functions. The configuration of the control circuitry between the output latch and the port pin determines the nature of any particular port pin function. An inspection of Figure 2.7 reveals that only port 1 cannot have alternate functions; ports 0, 2, and 3 can be programmed.

The ports are not capable of driving loads that require currents in the tens of milliamperes (mA). As previously mentioned, the 8051 has many family members, and many are fabricated in varying technologies. An example range of logic-level currents, voltages, and total device power requirements is given in the following table:

Parameter	V_{oh}	I_{oh}	V_{ol}	I_{ol}	V_{il}	I_{il}	V_{ih}	I_{ih}	P_t
CMOS	2.4 V	−60 μA	.45 V	1.6 mA	.9 V	|10 μA|	1.9 V	|10 μA|	50 mW
NMOS	2.4 V	−80 μA	.45 V	1.6 mA	.8 V	−800 μA	2.0 V	10 μA	800 mW

These figures tell us that driving more than two LSTTL inputs degrades the noise immunity of the ports and that careful attention must be paid to buffering the ports when they must drive currents in excess of those listed. Again, one must refer to the manufacturers' data books when designing a "real" application.

Port 0

Port 0 pins may serve as inputs, outputs, or, when used together, as a bi-directional low-order address and data bus for external memory. For example, when a pin is to be used as an input, a 1 *must be* written to the corresponding port 0 latch by the program, thus turning both of the output transistors off, which in turn causes the pin to "float" in a high-impedance state, and the pin is essentially connected to the input buffer.

When used as an output, the pin latches that are programmed to a 0 will turn on the lower FET, grounding the pin. All latches that are programmed to a 1 still float; thus, external pullup resistors will be needed to supply a logic high when using port 0 as an output.

When port 0 is used as an address bus to external memory, internal control signals switch the address lines to the gates of the Field Effect Transistories (FETs). A logic 1 on an address bit will turn the upper FET on and the lower FET off to provide a logic high at the pin. When the address bit is a zero, the lower FET is on and the upper FET off to

provide a logic low at the pin. After the address has been formed and latched into external circuits by the Address Latch Enable (ALE) pulse, the bus is turned around to become a data bus. Port 0 now reads data from the external memory and must be configured as an input, so a logic 1 is automatically written by internal control logic to all port 0 latches.

Port 1

Port 1 pins have no dual functions. Therefore, the output latch is connected directly to the gate of the lower FET, which has an FET circuit labeled "Internal FET Pullup" as an active pullup load.

Used as an input, a 1 is written to the latch, turning the lower FET off; the pin and the input to the pin buffer are pulled high by the FET load. An external circuit can overcome the high impedance pullup and drive the pin low to input a 0 or leave the input high for a 1.

If used as an output, the latches containing a 1 can drive the input of an external circuit high through the pullup. If a 0 is written to the latch, the lower FET is on, the pullup is off, and the pin can drive the input of the external circuit low.

To aid in speeding up switching times when the pin is used as an output, the internal FET pullup has another FET in parallel with it. The second FET is turned on for two oscillator time periods during a low-to-high transition on the pin, as shown in Figure 2.7. This arrangement provides a low impedance path to the positive voltage supply to help reduce rise times in charging any parasitic capacitances in the external circuitry.

Port 2

Port 2 may be used as an input/output port similar in operation to port 1. The alternate use of port 2 is to supply a high-order address byte in conjunction with the port 0 low-order byte to address external memory.

Port 2 pins are momentarily changed by the address control signals when supplying the high byte of a 16-bit address. Port 2 latches remain stable when external memory is addressed, as they do not have to be turned around (set to 1) for data input as is the case for port 0.

Port 3

Port 3 is an input/output port similar to port 1. The input and output functions can be programmed under the control of the P3 latches or under the control of various other special function registers. The port 3 alternate uses are shown in the following table:

PIN	ALTERNATE USE	SFR
P3.0–RXD	Serial data input	SBUF
P3.1–TXD	Serial data output	SBUF
P3.2–$\overline{\text{INT0}}$	External interrupt 0	TCON.1
P3.3–$\overline{\text{INT1}}$	External interrupt 1	TCON.3
P3.4–T0	External timer 0 input	TMOD
P3.5–T1	External timer 1 input	TMOD
P3.6–$\overline{\text{WR}}$	External memory write pulse	—
P3.7–$\overline{\text{RD}}$	External memory read pulse	—

Unlike ports 0 and 2, which can have external addressing functions and change all eight port bits when in alternate use, each pin of port 3 may be individually programmed to be used either as I/O or as one of the alternate functions.

External Memory

The system designer is not limited by the amount of internal RAM and ROM available on chip. Two separate external memory spaces are made available by the 16-bit PC and DPTR and by different control pins for enabling external ROM and RAM chips. Internal control circuitry accesses the correct physical memory, depending upon the machine cycle state and the opcode being executed.

There are several reasons for adding external memory, particularly program memory, when applying the 8051 in a system. When the project is in the prototype stage, the expense—in time and money—of having a masked internal ROM made for each program "try" is prohibitive. To alleviate this problem, the manufacturers make available an EPROM version, the 8751, which has 4K of on-chip EPROM that may be programmed and erased as needed as the program is developed. The resulting circuit board layout will be identical to one that uses a factory-programmed 8051. The only drawbacks to the 8751 are the specialized EPROM programmers that must be used to program the non-standard 40-pin part, and the limit of "only" 4096 bytes of program code.

The 8751 solution works well if the program will fit into 4K bytes. Unfortunately, many times, particularly if the program is written in a high-level language, the program size exceeds 4K bytes, and an external program memory is needed. Again, the manufacturers provide a version for the job, the ROMless 8031. The $\overline{EA}$ pin is grounded when using the 8031, and all program code is contained in an external EPROM that may be as large as 64K bytes and that can be programmed using standard EPROM programmers.

External RAM, which is accessed by the DPTR, may also be needed when 128 bytes of internal data storage is not sufficient. External RAM, up to 64K bytes, may also be added to any chip in the 8051 family.

Connecting External Memory

Figure 2.8 shows the connections between an 8031 and an external memory configuration consisting of 16K bytes of EPROM and 8K bytes of static RAM. The 8051 accesses external RAM whenever certain program instructions are executed. External ROM is accessed whenever the $\overline{EA}$ (external access) pin is connected to ground or when the PC contains an address higher than the last address in the internal 4K bytes ROM (0FFFh). 8051 designs can thus use internal and external ROM automatically; the 8031, having no internal ROM, must have $\overline{EA}$ grounded.

Figure 2.9 shows the timing associated with an external memory access cycle. During any memory access cycle, port 0 is time multiplexed. That is, it first provides the lower byte of the 16-bit memory address, then acts as a bidirectional data bus to write or read a byte of memory data. Port 2 provides the high byte of the memory address during the entire memory read/write cycle.

The lower address byte from port 0 must be latched into an external register to save the byte. Address byte save is accomplished by the ALE clock pulse that provides the correct timing for the '373 type data latch. The port 0 pins then become free to serve as a data bus.

If the memory access is for a byte of program code in the ROM, the $\overline{PSEN}$ (program store enable) pin will go low to enable the ROM to place a byte of program code on the data bus. If the access is for a RAM byte, the $\overline{WR}$ (write) or $\overline{RD}$ (read) pins will go low, enabling data to flow between the RAM and the data bus.

The ROM may be expanded to 64K by using a 27512 type EPROM and connecting the remaining port 2 upper address lines A14–A15 to the chip.

At this time the largest static RAMs available are 32K in size; RAM can be expanded to 64K by using two 32K RAMs that are connected through address A14 of port 2. The

FIGURE 2.8 External Memory Connections

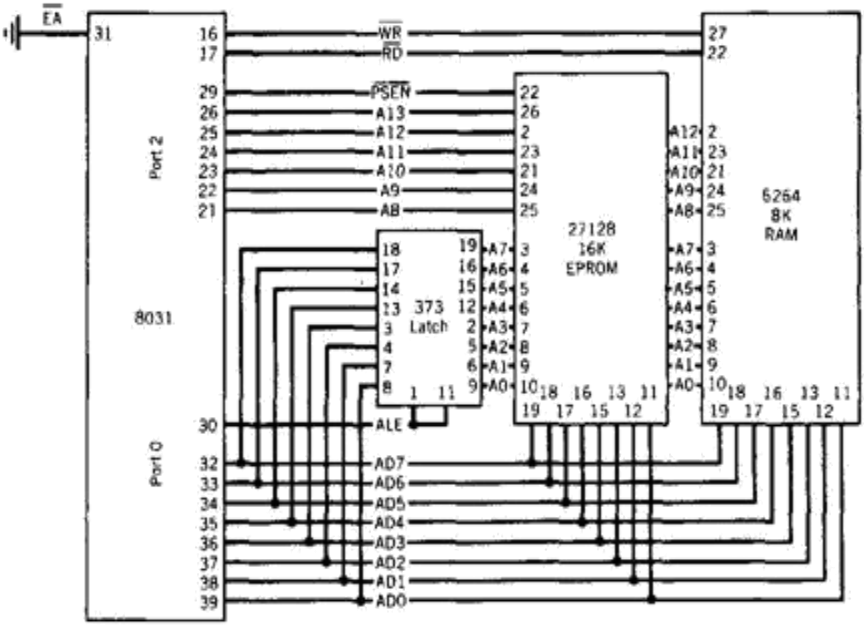


FIGURE 2.9 External Memory Timing

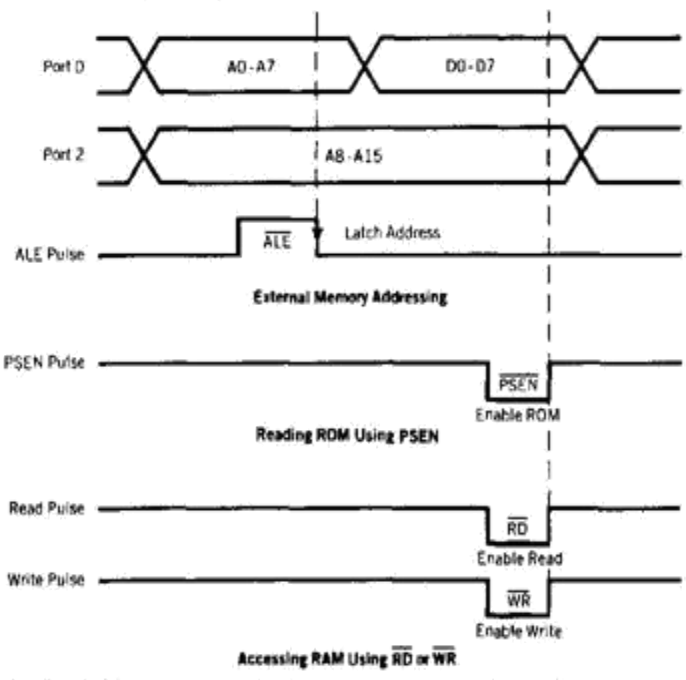
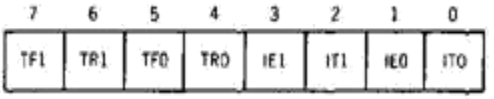


FIGURE 2.10 TCON and TMOD Function Registers



THE TIMER CONTROL (TCON) SPECIAL FUNCTION REGISTER

Bit	Symbol	Function
7	TF1	Timer 1 Overflow flag. Set when timer rolls from all ones to zero. Cleared when processor vectors to execute interrupt service routine located at program address 001Bh.
6	TR1	Timer 1 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to halt timer. Does not reset timer.
5	TF0	Timer 0 Overflow flag. Set when timer rolls from all ones to zero. Cleared when processor vectors to execute interrupt service routine located at program address 000Bh.
4	TR0	Timer 0 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to halt timer. Does not reset timer.
3	IE1	External interrupt 1 edge flag. Set to 1 when a high to low edge signal is received on port 3 pin 3.3 ($\overline{INT1}$). Cleared when processor vectors to interrupt service routine located at program address 0013h. Not related to timer operations.
2	IT1	External interrupt 1 signal type control bit. Set to 1 by program to enable external interrupt 1 to be triggered by a falling edge signal. Set to 0 by program to enable a low level signal on external interrupt 1 to generate an interrupt.
1	IE0	External interrupt 0 edge flag. Set to 1 when a high to low edge signal is received on port 3 pin 3.2 ($\overline{INT0}$). Cleared when processor vectors to interrupt service routine located at program address 0003h. Not related to timer operations.

Continued

first 32K RAM (0000h–7FFFh) can then be enabled when A15 of port 2 is low, and the second 32K RAM (8000h–FFFFh) when A15 is high, by using an inverter.

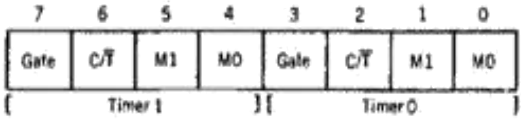
Note that the $\overline{WR}$ and $\overline{RD}$ signals are alternate uses for port 3 pins 16 and 17. Also, port 0 is used for the lower address byte and data; port 2 is used for upper address bits. The use of external memory consumes many of the port pins, leaving only port 1 and parts of port 3 for general I/O.

Counters and Timers

Many microcontroller applications require the counting of external events, such as the frequency of a pulse train, or the generation of precise internal time delays between computer actions. Both of these tasks can be accomplished using software techniques, but software loops for counting or timing keep the processor occupied so that other, perhaps more important, functions are not done. To relieve the processor of this burden, two 16-bit *up* counters, named **T0** and **T1**, are provided for the general use of the programmer. Each counter may be programmed to count internal clock pulses, acting as a timer, or programmed to count external pulses as a counter.

Bit	Symbol	Function
0	IT0	External interrupt 0 signal type control bit. Set to 1 by program to enable external interrupt 0 to be triggered by a falling edge signal. Set to 0 by program to enable a low level signal on external interrupt 0 to generate an interrupt.

Bit addressable as TCON.0 to TCON.7



THE TIMER MODE CONTROL (TMOD) SPECIAL FUNCTION REGISTER

Bit	Symbol	Function
7/3	Gate	OR gate enable bit which controls RUN/STOP of timer 1/0. Set to 1 by program to enable timer to run if bit TR1/0 in TCON is set and signal on external interrupt $\overline{INT1}$/0 pin is high. Cleared to 0 by program to enable timer to run if bit TR1/0 in TCON is set.
6/2	C/$\overline{T}$	Set to 1 by program to make timer 1/0 act as a counter by counting pulses from external input pins 3.5 (T1) or 3.4 (T0). Cleared to 0 by program to make timer act as a timer by counting internal frequency.
5/1	M1	Timer/counter operating mode select bit 1. Set/cleared by program to select mode.
4/0	M0	Timer/counter operating mode select bit 0. Set/cleared by program to select mode.

M1	M0	Mode
0	0	0
0	1	1
1	0	2
1	1	3

TMOD is not bit addressable

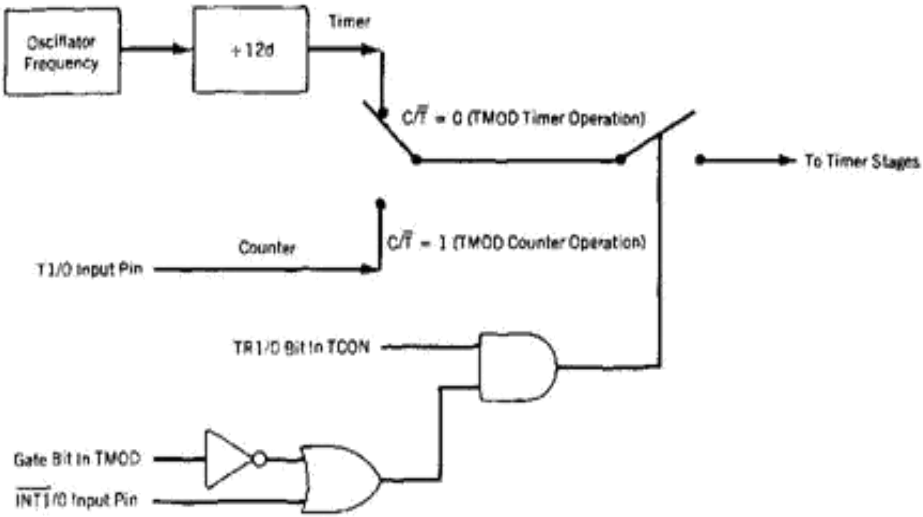
The counters are divided into two 8-bit registers called the timer low (TL0, TL1) and high (TH0, TH1) bytes. All counter action is controlled by bit states in the timer mode control register (TMOD), the timer/counter control register (TCON), and certain program instructions.

TMOD is dedicated solely to the two timers and can be considered to be two duplicate 4-bit registers, each of which controls the action of one of the timers. TCON has control bits and flags for the timers in the upper nibble, and control bits and flags for the external interrupts in the lower nibble. Figure 2.10 shows the bit assignments for TMOD and TCON.

Timer Counter Interrupts

The counters have been included on the chip to relieve the processor of timing and counting chores. When the program wishes to count a certain number of internal pulses or external events, a number is placed in one of the counters. The number represents the maximum count *less* the desired count, plus one. The counter increments from the initial number to the maximum and then rolls over to zero on the final pulse and also sets a timer flag. The flag condition may be tested by an instruction to tell the program that the count has been accomplished, or the flag may be used to interrupt the program.

FIGURE 2.11 Timer/Counter Control Logic



Timing

If a counter is programmed to be a timer, it will count the internal clock frequency of the 8051 oscillator divided by 12d. As an example, if the crystal frequency is 6.0 megahertz, then the timer clock will have a frequency of 500 kilohertz.

The resultant timer clock is gated to the timer by means of the circuit shown in Figure 2.11. In order for oscillator clock pulses to reach the timer, the C/$\overline{T}$ bit in the TMOD register must be set to 0 (timer operation). Bit TRX in the TCON register must be set to 1 (timer run), and the *gate* bit in the TMOD register must be 0, or external pin $\overline{INTX}$ must be a 1. In other words, the counter is configured as a timer, then the timer pulses are gated to the counter by the run bit *and* the gate bit *or* the external input bits $\overline{INTX}$.

Timer Modes of Operation

The timers may operate in any one of four modes that are determined by the mode bits, M1 and M0, in the TMOD register. Figure 2.12 shows the four timer modes.

Timer Mode 0

Setting timer X mode bits to 00b in the TMOD register results in using the THX register as an 8-bit counter and TLX as a 5-bit counter; the pulse input is divided by 32d in TL so that TH counts the original oscillator frequency reduced by a total 384d. As an example, the 6 megahertz oscillator frequency would result in a final frequency to TH of 15625 hertz. The timer flag is set whenever THX goes from FFh to 00h, or in .0164 seconds for a 6 megahertz crystal if THX starts at 00h.

Timer Mode 1

Mode 1 is similar to mode 0 except TLX is configured as a full 8-bit counter when the mode bits are set to 01b in TMOD. The timer flag would be set in .1311 seconds using a 6 megahertz crystal.

Timer Mode 2

Setting the mode bits to 10b in TMOD configures the timer to use only the TLX counter as an 8-bit counter. THX is used to hold a value that is loaded into TLX every time TLX overflows from FFh to 00h. The timer flag is also set when TLX overflows.

This mode exhibits an auto-reload feature: TLX will count up from the number in THX, overflow, and be initialized again with the contents of THX. For example, placing

9Ch in THX will result in a delay of exactly .0002 seconds before the overflow flag is set if a 6 megahertz crystal is used.

Timer Mode 3

Timers 0 and 1 may be programmed to be in mode 0, 1, or 2 independently of a similar mode for the other timer. This is not true for mode 3; the timers do not operate independently if mode 3 is chosen for timer 0. Placing timer 1 in mode 3 causes it to stop counting; the control bit TR1 and the timer 1 flag TF1 are then used by timer 0.

Timer 0 in mode 3 becomes two completely separate 8-bit counters. TL0 is controlled by the gate arrangement of Figure 2.11 and sets timer flag TF0 whenever it overflows from FFh to 00h. TH0 receives the timer clock (the oscillator divided by 12) under the control of TR1 only and sets the TF1 flag when it overflows.

Timer 1 may still be used in modes 0, 1, and 2, while timer 0 is in mode 3 with one important exception: *No interrupts* will be generated by timer 1 while timer 0 is using the TF1 overflow flag. Switching timer 1 to mode 3 will stop it (and hold whatever count is in timer 1). Timer 1 can be used for baud rate generation for the serial port, or any other mode 0, 1, or 2 function that does not depend upon an interrupt (or any other use of the TF1 flag) for proper operation.

Counting

The only difference between counting and timing is the source of the clock pulses to the counters. When used as a timer, the clock pulses are sourced from the oscillator through the divide-by-12d circuit. When used as a counter, pin T0 (P3.4) supplies pulses to counter 0, and pin T1 (P3.5) to counter 1. The C/$\overline{T}$ bit in TMOD must be set to 1 to enable pulses from the TX pin to reach the control circuit shown in Figure 2.11.

The input pulse on TX is sampled during P2 of state 5 every machine cycle. A change on the input from high to low between samples will increment the counter. Each high and low state of the input pulse must thus be held constant for at least one machine cycle to ensure reliable counting. Since this takes 24 pulses, the maximum input frequency that can be accurately counted is the oscillator frequency divided by 24. For our 6 megahertz crystal, the calculation yields a maximum external frequency of 250 kilohertz.

Serial Data Input/Output

Computers must be able to communicate with other computers in modern multiprocessor distributed systems. One cost-effective way to communicate is to send and receive data bits serially. The 8051 has a serial data communication circuit that uses register SBUF to hold data. Register SCON controls data communication, register PCON controls data rates, and pins RXD (P3.0) and TXD (P3.1) connect to the serial data network.

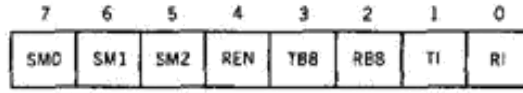
SBUF is physically two registers. One is write only and is used to hold data to be transmitted *out* of the 8051 via TXD. The other is read only and holds received data *from* external sources via RXD. Both mutually exclusive registers use address 99h.

There are four programmable modes for serial data communication that are chosen by setting the SMX bits in SCON. Baud rates are determined by the mode chosen. Figure 2.13 shows the bit assignments for SCON and PCON.

Serial Data Interrupts

Serial data communication is a relatively slow process, occupying many milliseconds per data byte to accomplish. In order not to tie up valuable processor time, serial data flags are

FIGURE 2.13 SCON and PCON Function Registers

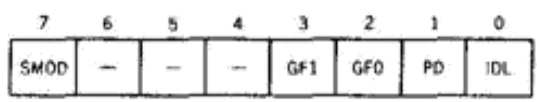


THE SERIAL PORT CONTROL (SCON) SPECIAL FUNCTION REGISTER

Bit	Symbol	Function
7	SM0	Serial port mode bit 0. Set/cleared by program to select mode.
6	SM1	Serial port mode bit 1. Set/cleared by program to select mode.
5	SM2	Multiprocessor communications bit. Set/cleared by program to enable multiprocessor communications in modes 2 and 3. When set to 1 an interrupt is generated if bit 9 of the received data is a 1; no interrupt is generated if bit 9 is a 0. If set to 1 for mode 1, no interrupt will be generated unless a valid stop bit is received. Clear to 0 if mode 0 is in use.
4	REN	Receive enable bit. Set to 1 to enable reception; cleared to 0 to dissable reception.
3	TB8	Transmitted bit 8. Set/cleared by program in modes 2 and 3.
2	RB8	Received bit 8. Bit 8 of received data in modes 2 and 3; stop bit in mode 1. Not used in mode 0.
1	TI	Transmit interrupt flag. Set to one at the end of bit 7 time in mode 0, and at the beginning of the stop bit for other modes. Must be cleared by the program.
0	RI	Receive interrupt flag. Set to one at the end of bit 7 time in mode 0, and halfway through the stop bit for other modes. Must be cleared by the program.

SM0	SM1	Mode	Description
0	0	0	Shift register; baud = f/12
0	1	1	8-bit UART; baud = variable
1	0	2	9-bit UART; baud = f/32 or f/64
1	1	3	9-bit UART; baud = variable

Bit addressable as SCON.0 to SCON.7



THE POWER MODE CONTROL (PCON) SPECIAL FUNCTION REGISTER

Bit	Symbol	Function
7	SMOD	Serial baud rate modify bit. Set to 1 by program to double baud rate using timer 1 for modes 1, 2, and 3. Cleared to 0 by program to use timer 1 baud rate.
6-4	—	Not implemented.
3	GF1	General purpose user flag bit 1. Set/cleared by program.
2	GF0	General purpose user flag bit 0. Set/cleared by program.
1	PD	Power down bit. Set to 1 by program to enter power down configuration for CHMOS processors.
0	IDL	Idle mode bit. Set to 1 by program to enter idle mode configuration for CHMOS processors. PCON is not bit addressable.

included in SCON to aid in efficient data transmission and reception. Notice that data transmission is under the complete control of the program, but reception of data is unpredictable and at random times that are beyond the control of the program.

The serial data flags in SCON, TI and RI, are set whenever a data byte is transmitted (TI) or received (RI). These flags are ORed together to produce an interrupt to the program. The program must read these flags to determine which caused the interrupt and then clear the flag. This is unlike the timer flags that are cleared automatically; it is the responsibility of the programmer to write routines that handle the serial data flags.

Data Transmission

Transmission of serial data bits begins *anytime* data is written to SBUF. TI is set to a 1 when the data has been transmitted and signifies that SBUF is empty (for transmission purposes) and that another data byte can be sent. If the program fails to wait for the TI flag and overwrites SBUF while a previous data byte is in the process of being transmitted, the results will be unpredictable (a polite term for "garbage out").

Data Reception

Reception of serial data will begin *if* the receive enable bit (REN) in SCON is set to 1 for all modes. In addition, for mode 0 *only*, RI must be cleared to 0 also. Receiver interrupt flag RI is set after data has been received in all modes. Setting REN is the only direct program control that limits the reception of unexpected data; the requirement that RI also be 0 for mode 0 prevents the reception of new data until the program has dealt with the old data and reset RI.

Reception can *begin* in modes 1, 2, and 3 if RI is set when the serial stream of bits begins. RI must have been reset by the program before the *last* bit is received or the *incoming* data will be lost. Incoming data is not transferred to SBUF until the last data bit has been received so that the previous transmission can be read from SBUF while new data is being received.

Serial Data Transmission Modes

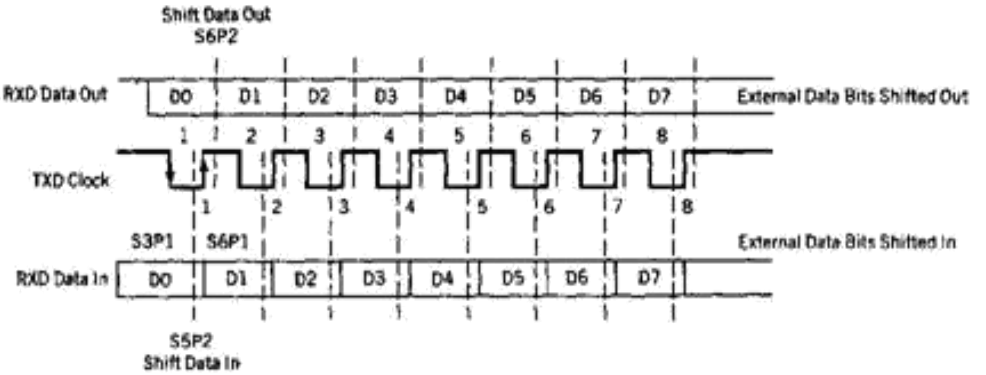
The 8051 designers have included four modes of serial data transmission that enable data communication to be done in a variety of ways and a multitude of baud rates. Modes are selected by the programmer by setting the mode bits SM0 and SM1 in SCON. Baud rates are fixed for mode 0 and variable, using timer 1 and the serial baud rate modify bit (SMOD) in PCON, for modes 1, 2, and 3.

Serial Data Mode 0—Shift Register Mode

Setting bits SM0 and SM1 in SCON to 00b configures SBUF to receive or transmit eight data bits using pin RXD for *both* functions. Pin TXD is connected to the internal shift frequency pulse source to supply shift pulses to external circuits. The shift frequency, or baud rate, is fixed at 1/12 of the oscillator frequency, the same rate used by the timers when in the timer configuration. The TXD shift clock is a square wave that is low for machine cycle states S3–S4–S5 and high for S6–S1–S2. Figure 2.14 shows the timing for mode 0 shift register data transmission.

When transmitting, data is shifted *out* of RXD; the data changes on the *falling* edge of S6P2, or one clock pulse after the *rising* edge of the output TXD shift clock. The system designer must design the external circuitry that receives this transmitted data to receive the data reliably based on this timing.

FIGURE 2.14 Shift Register Mode 0 Timing



Received data comes *in* on pin RXD and should be synchronized with the shift clock produced at TXD. Data is sampled on the *falling* edge of S5P2 and shifted in to SBUF on the *rising* edge of the shift clock.

Mode 0 is intended *not* for data communication between computers, but as a high-speed serial data-collection method using discrete logic to achieve high data rates. The baud rate used in mode 0 will be much higher than standard for any reasonable oscillator frequency; for a 6 megahertz crystal, the shift rate will be 500 kilohertz.

Serial Data Mode 1—Standard UART

When SM0 and SM1 are set to 01b, SBUF becomes a 10-bit full-duplex receiver/transmitter that may receive and transmit data at the same time. Pin RXD receives all data, and pin TXD transmits all data. Figure 2.15 shows the format of a data word.

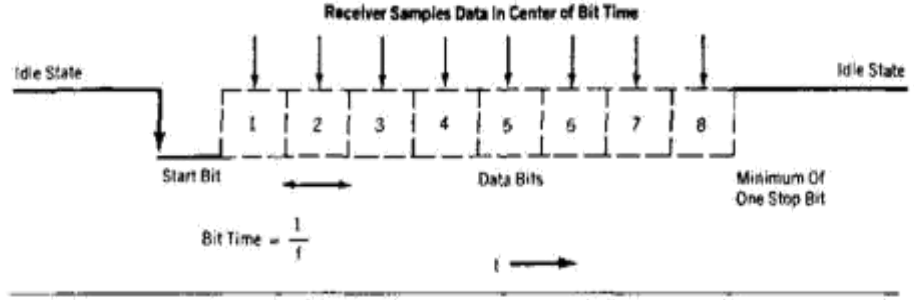
Transmitted data is sent as a start bit, eight data bits (Least Significant Bit, LSB, first), and a stop bit. Interrupt flag TI is set once all ten bits have been sent. Each bit interval is the inverse of the baud rate frequency, and each bit is maintained high or low over that interval.

Received data is obtained in the same order; reception is triggered by the falling edge of the start bit and continues if the stop bit is true (0 level) halfway through the start bit interval. This is an anti-noise measure; if the reception circuit is triggered by noise on the transmission line, the check for a low after half a bit interval should limit false data reception.

Data bits are shifted into the receiver at the programmed baud rate, and the data word will be loaded to SBUF *if* the following conditions are true: RI *must* be 0, *and* mode bit SM2 is 0 *or* the stop bit is 1 (the normal state of stop bits). RI set to 0 implies that the program has read the previous data byte and is ready to receive the next; a normal stop bit will then complete the transfer of data to SBUF regardless of the state of SM2. SM2 set to 0 enables the reception of a byte with any stop bit state, a condition which is of limited use in this mode, but very useful in modes 2 and 3. SM2 set to 1 forces reception of only "good" stop bits, an anti-noise safeguard.

Of the original ten bits, the start bit is discarded, the eight data bits go to SBUF, and the stop bit is saved in bit RB8 of SCON. RI is set to 1, indicating a new data byte has been received.

FIGURE 2.15 Standard UART Data Word



If RI is found to be set at the end of the reception, indicating that the previously received data byte has not been read by the program, or if the other conditions listed are not true, the *new* data will not be loaded and will be *lost*.

Mode 1 Baud Rates

Timer 1 is used to generate the baud rate for mode 1 by using the overflow flag of the timer to determine the baud frequency. Typically, timer 1 is used in timer mode 2 as an autoload 8-bit timer that generates the baud frequency:

$$f_{baud} = \frac{2^{SMOD}}{32d} \times \frac{\text{oscillator frequency}}{12d \times [256d - (TH1)]}$$

SMOD is the control bit in PCON and can be 0 or 1, which raises the 2 in the equation to a value of 1 or 2.

If timer 1 is not run in timer mode 2, then the baud rate is

$$f_{baud} = \frac{2^{SMOD}}{32d} \times (\text{timer 1 overflow frequency})$$

and timer 1 can be run using the internal clock or as a counter that receives clock pulses from any external source via pin T1.

The oscillator frequency is chosen to help generate both standard and nonstandard baud rates. If standard baud rates are desired, then an 11.0592 megahertz crystal could be selected. To get a standard rate of 9600 hertz then, the setting of TH1 may be found as follows:

$$TH1 = 256d - \left(\frac{2^0}{32d} \times \frac{11.0592 \times 10^6}{12 \times 9600d}\right) = 253.0000d = 0FDh$$

if SMOD is cleared to 0.

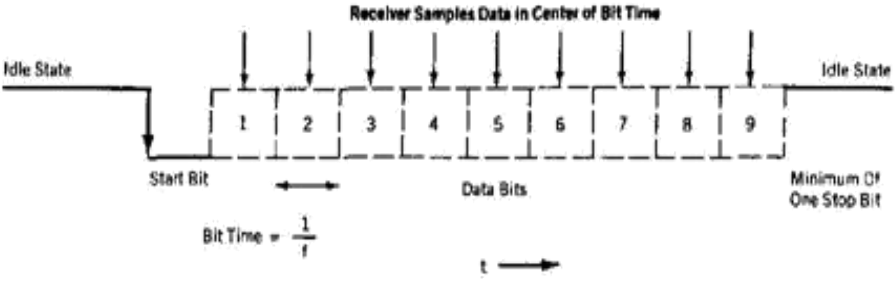
Serial Data Mode 2—Multiprocessor Mode

Mode 2 is similar to mode 1 except 11 bits are transmitted: a start bit, nine data bits, and a stop bit, as shown in Figure 2.16. The ninth data bit is gotten from bit TB8 in SCON during transmit and stored in bit RB8 of SCON when data is received. Both the start and stop bits are discarded.

The baud rate is programmed as follows:

$$f_{baud2} = \frac{2^{SMOD}}{64d} \times \text{oscillator frequency}$$

FIGURE 2.16 Multiprocessor Data Word



Here, as in the case for mode 0, the baud rate is much higher than standard communication rates. This high data rate is needed in many multi-processor applications. Data can be collected quickly from an extensive network of communicating microcontrollers if high baud rates are employed.

The conditions for setting RI for mode 2 are similar to mode 1: RI must be 0 before the last bit is received, *and* SM2 must be 0 *or* the ninth data bit must be a 1. Setting RI based upon the state of SM2 in the receiving 8051 and the state of bit 9 in the transmitted message makes multiprocessing possible by enabling some receivers to be interrupted by certain messages, while other receivers ignore those messages. Only those 8051's that have SM2 set to 0 will be interrupted by received data which has the ninth data bit set to 0; those with SM2 set to 1 will not be interrupted by messages with data bit 9 at 0. *All* receivers will be interrupted by data words that have the ninth data bit set to 1; the state of SM2 will not block reception of such messages.

This scheme allows the transmitting computer to "talk" to selected receiving computers without interrupting other receiving computers. Receiving computers can be commanded by the "talker" to "listen" or "deafen" by transmitting coded byte(s) with the ninth data bit set to 1. The 1 in data bit 9 interrupts all receivers, instructing those that are programmed to respond to the coded byte(s) to program the state of SM2 in their respective SCON registers. Selected listeners then respond to the bit 9 set to 0 messages, while all other receivers ignore these messages. The talker can change the mix of listeners by transmitting bit 9 set to 1 messages that instruct new listeners to set SM2 to 0, while others are instructed to set SM2 to 1.

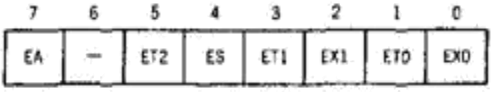
Serial Data Mode 3

Mode 3 is identical to mode 2 except that the baud rate is determined exactly as in mode 1, using Timer 1 to generate communication frequencies.

Interrupts

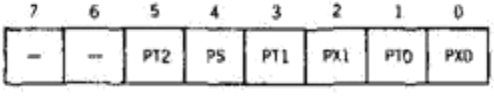
A computer program has only two ways to determine the conditions that exist in internal and external circuits. One method uses software instructions that jump on the states of flags and port pins. The second responds to hardware signals, called interrupts, that force the program to call a sub-routine. Software techniques use up processor time that could be devoted to other tasks; interrupts take processor time only when action by the program is needed. Most applications of microcontrollers involve responding to events quickly enough to control the environment that generates the events (generically termed "real-

FIGURE 2.17 IE and IP Function Registers



THE INTERRUPT ENABLE (IE) SPECIAL FUNCTION REGISTER

Bit	Symbol	Function
7	EA	Enable interrupts bit. Cleared to 0 by program to disable all interrupts; set to 1 to permit individual interrupts to be enabled by their enable bits.
6	—	Not implemented.
5	ET2	Reserved for future use.
4	ES	Enable serial port interrupt. Set to 1 by program to enable serial port interrupt; cleared to 0 to disable serial port interrupt.
3	ET1	Enable timer 1 overflow interrupt. Set to 1 by program to enable timer 1 overflow interrupt; cleared to 0 to disable timer 1 overflow interrupt.
2	EX1	Enable external interrupt 1. Set to 1 by program to enable $\overline{\text{INT1}}$ interrupt; cleared to 0 to disable $\overline{\text{INT1}}$ interrupt.
1	ET0	Enable timer 0 overflow interrupt. Set to 1 by program to enable timer 0 overflow interrupt; cleared to 0 to disable timer 0 overflow interrupt.
0	EX0	Enable external interrupt 0. Set to 1 by program to enable $\overline{\text{INT0}}$ interrupt; cleared to 0 to disable $\overline{\text{INT0}}$ interrupt.
		Bit addressable as IE.0 to IE.7



THE INTERRUPT PRIORITY (IP) SPECIAL FUNCTION REGISTER

Bit	Symbol	Function
7	—	Not implemented.
6	—	Not implemented.
5	PT2	Reserved for future use.
4	PS	Priority of serial port interrupt. Set/cleared by program.
3	PT1	Priority of timer 1 overflow interrupt. Set/cleared by program.
2	PX1	Priority of external interrupt 1. Set/cleared by program.
1	PT0	Priority of timer 0 overflow interrupt. Set/cleared by program.
0	PX0	Priority of external interrupt 0. Set/cleared by program.
		Note: Priority may be 1 (highest) or 0 (lowest)
		Bit addressable as IP.0 to IP.7

time programming"). Interrupts are often the only way in which real-time programming can be done successfully.

Interrupts may be generated by internal chip operations or provided by external sources. Any interrupt can cause the 8051 to perform a hardware call to an interrupt-handling subroutine that is located at a predetermined (by the 8051 designers) absolute address in program memory.

Five interrupts are provided in the 8051. Three of these are generated automatically by internal operations: timer flag 0, timer flag 1, and the serial port interrupt (RI *or* TI). Two interrupts are triggered by external signals provided by circuitry that is connected to pins $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ (port pins P3.2 and P3.3).

All interrupt functions are under the control of the program. The programmer is able to alter control bits in the interrupt enable register (IE), the interrupt priority register (IP), and the timer control register (TCON). The program can block all or any combination of the interrupts from acting on the program by suitably setting or clearing bits in these registers. The IE and IP registers are shown in Figure 2.17.

After the interrupt has been handled by the interrupt subroutine, which is placed by the programmer at the interrupt location in program memory, the interrupted program must resume operation at the instruction where the interrupt took place. Program resumption is done by storing the interrupted PC address on the stack in RAM before changing the PC to the interrupt address in ROM. The PC address will be restored from the stack after an RETI instruction is executed at the end of the interrupt subroutine.

Timer Flag Interrupt

When a timer/counter overflows, the corresponding timer flag, TF0 or TF1, is set to 1. The flag is cleared to 0 when the resulting interrupt generates a program call to the appropriate timer subroutine in memory.

Serial Port Interrupt

If a data byte is received, an interrupt bit, RI, is set to 1 in the SCON register. When a data byte has been transmitted an interrupt bit, TI, is set in SCON. These are ORed together to provide a single interrupt to the processor: the serial port interrupt. These bits are *not* cleared when the interrupt-generated program call is made by the processor. The program that handles serial data communication *must* reset RI or TI to 0 to enable the next data communication operation.

External Interrupts

Pins $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ are used by external circuitry. Inputs on these pins can set the interrupt flags IE0 and IE1 in the TCON register to 1 by two different methods. The IEX flags may be set when the $\overline{\text{INTX}}$ pin signal reaches a low *level*, or the flags may be set when a high-to-low *transition* takes place on the $\overline{\text{INTX}}$ pin. Bits IT0 and IT1 in TCON program the $\overline{\text{INTX}}$ pins for low-level interrupt when set to 0 and program the $\overline{\text{INTX}}$ pins for transition interrupt when set to 1.

Flags IEX will be reset when a transition-generated interrupt is accepted by the processor and the interrupt subroutine is accessed. It is the responsibility of the system designer and programmer to reset *any* level-generated external interrupts when they are serviced by the program. The external circuit *must* remove the low level before an RETI is executed. Failure to remove the low will result in an immediate interrupt after RETI, from the same source.

Reset

A reset can be considered to be the ultimate interrupt because the program may not block the action of the voltage on the RST pin. This type of interrupt is often called "non-maskable," since no combination of bits in any register can stop, or mask the reset action. Unlike other interrupts, the PC is not stored for later program resumption; a reset is an absolute command to jump to program address 0000h and commence running from there.

Whenever a high level is applied to the RST pin, the 8051 enters a reset condition. After the RST pin is brought low, the internal registers will have the values shown in the following table:

REGISTER	VALUE(HEX)
PC	0000
DPTR	0000
A	00
B	00
SP	07
PSW	00
P0 3	FF
IP	XXX00000b
IE	0XX00000b
TCON	00
TMOD	00
TH0	00
TL0	00
TH1	00
TL1	00
SCON	00
SBUF	XX
PCON	0XXXXXXXb

Internal RAM is not changed by a reset; however, the states of the internal RAM when power is first applied to the 8051 are random. Register bank 0 is selected upon reset as all bits in PSW are 0.

Interrupt Control

The program must be able, at critical times, to inhibit the action of some or all of the interrupts so that crucial operations can be finished. The IE register holds the programmable bits that can enable or disable all the interrupts as a group, or if the group is enabled, each individual interrupt source can be enabled or disabled.

Often, it is desirable to be able to set priorities among competing interrupts that may conceivably occur simultaneously. The IP register bits may be set by the program to assign priorities among the various interrupt sources so that more important interrupts can be serviced first should two or more interrupts occur at the same time.

Interrupt Enable/Disable

Bits in the EI register are set to 1 if the corresponding interrupt source is to be enabled and set to 0 to disable the interrupt source. Bit EA is a master, or "global," bit that can enable or disable all of the interrupts.

Interrupt Priority

Register IP bits determine if any interrupt is to have a high or low priority. Bits set to 1 give the accompanying interrupt a high priority while a 0 assigns a low priority. Interrupts with a high priority can interrupt another interrupt with a lower priority; the low priority interrupt continues after the higher is finished.

If two interrupts with the same priority occur at the same time, then they have the following ranking:

1. IE0
2. TF0
3. IE1
4. TF1
5. Serial = RI OR TI

The serial interrupt could be given the highest priority by setting the PS bit in IP to 1, and all others to 0.

Interrupt Destinations

Each interrupt source causes the program to do a hardware call to one of the dedicated addresses in program memory. It is the responsibility of the programmer to place a routine at the address that will service the interrupt.

The interrupt saves the PC of the program, which is running at the time the interrupt is serviced on the stack in internal RAM. A call is then done to the appropriate memory location. These locations are shown in the following table:

INTERRUPT	ADDRESS(HEX)
IE0	0003
TF0	000B
IE1	0013
TF1	001B
SERIAL	0023

A RETI instruction at the end of the routine restores the PC to its place in the interrupted program and resets the interrupt logic so that another interrupt can be serviced. Interrupts that occur but are ignored due to any blocking condition (IE bit not set or a higher priority interrupt already in process) *must* persist until they are serviced, or they will be *lost*. This requirement applies primarily to the level-activated $\overline{INTX}$ interrupts.

Software Generated Interrupts

When *any* interrupt flag is set to 1 *by any means*, an interrupt is generated unless blocked. This means that the program itself can cause interrupts of any kind to be generated simply by setting the desired interrupt flag to 1 using a program instruction.

SPECIAL-FUNCTION REGISTERS

Register	Bit	Primary Function	Bit Addressable
A	8	Math, data manipulation	Y
B	8	Math	Y
PC	16	Addressing program bytes	N
DPTR	16	Addressing code and external data	N
SP	8	Addressing internal RAM stack data	N
PSW	8	Processor status	Y
P0–P3	8	Store I/O port data	Y
TH0/TL0	8/8	Timer/counter 0	N
TH1/TL1	8/8	Timer/counter 1	N
TCON	8	Timer/counter control	Y
TMOD	8	Timer/counter control	N
SBUF	8	Serial port data	N
SCON	8	Serial port control	Y
PCON	8	Serial port control, user flags	N
IE	8	Interrupt enable control	Y
IP	8	Interrupt priority control	Y

DATA AND PROGRAM MEMORY

Internal	Bytes	Function
RAM	128	R0–R7 registers, data storage, stack
ROM	4K	Program storage

External	Bytes	Function
RAM	64K	Data storage
ROM	64K	Program storage

EXTERNAL CONNECTION PINS

		Function
Port pins	36	I/O, external memory, interrupts
Oscillator	2	Clock
Power	2	