UNIT V

8051 MICROCONTROLLERS

To make a complete microcomputer system, only microprocessor is not sufficient. It is necessary to add other peripherals such as ROM, RAM, decoders, drivers, number of I/O devices to make a complete microcomputer system. In addition, special purpose devices, such as interrupt controller, programmable timers, programmable I/O devices, DMA controllers may be added to improve the capability and performance and flexibility of a microcomputer system.

The key feature for microprocessor based design is that it has more flexibility to configure a system as large system or small system by adding suitable peripherals.

On the other hand, the microcontroller incorporates all the features that are found in microprocessor. The microcontroller has built-in ROM, RAM, parallel I/O, serial I/O, counters and a clock circuit. It has on-chip peripheral devices which makes it possible to have single microcomputer system.

Advantages of built-in peripherals:

Built-in peripherals have smaller access times hence speed is more.

Hardware reduces due to single chip microcomputer system.

Less hardware reduces PCB size and increases reliability of the system.

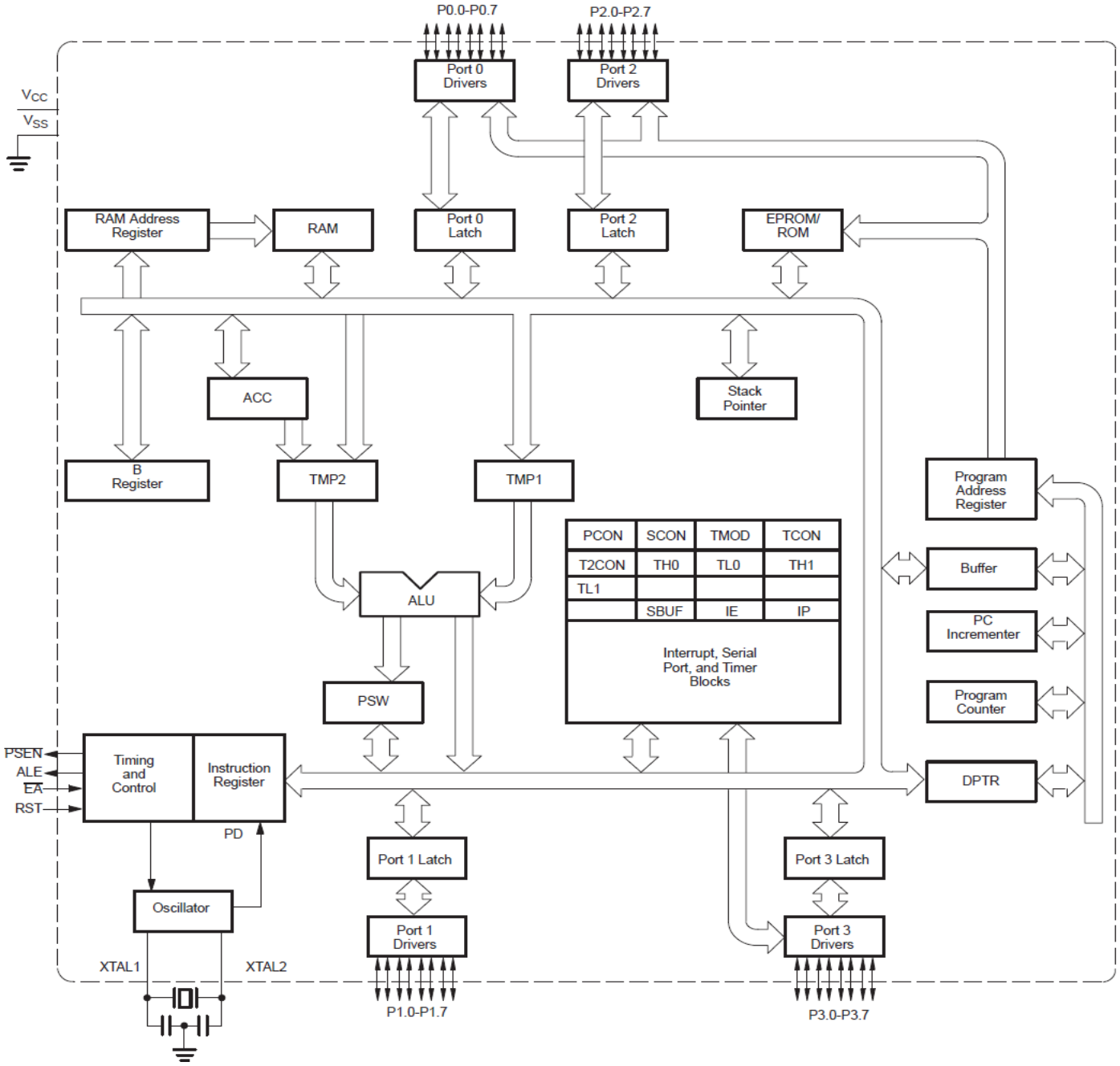
Comparison between Microprocessors and Microcontrollers:

No.	Microprocessor	Microcontroller
1.	Microprocessor contains ALU, control unit (clock and timing circuit), different register and interrupt circuit.	Microcontroller contains microprocessor, memory (ROM and RAM), I/O interfacing circuit and peripheral devices such as A/D converter, serial I/O, timer etc.
2.	It has many instructions to move data between memory and CPU.	It has one or two instructions to move data between memory and CPU.
3.	It has one or two bit handling instructions.	It has many bit handling instructions.
4.	Access times for memory and I/O devices are more.	Less access times for built-in memory and I/O devices.
5.	Microprocessor based system requires more hardware.	Microcontroller based system requires less hardware reducing PCB size and increasing the reliability.
6.	Microprocessor based system is more flexible in design point of view.	Less flexible in design point of view.
7.	It has single memory map for data and code.	It has separate memory map for data and code.
8.	Less number of pins are multifunctioned.	More number pins are multifunctioned.

Features of 8051:

- 4KB on-chip program memory (ROM/EPROM).
- 128 bytes on-chip data memory.
- Four register banks.
- 64KB each program and external RAM addressability.
- One microsecond instruction cycle with 12MHz crystal.
- 32 bidirectional I/O lines organized as four 8-bit ports.
- Multiple modes, high-speed programmable serial port (UART).
- 16-bit Timers/Counters.
- Direct byte and bit addressability.

Block Diagram of 8051:



Accumulator: The Accumulator, as it's name suggests, is used as a general register to accumulate the results of a large number of instructions. It can hold an 8-bit (1-byte) value.

'B' Register: The "B" register is very similar to the Accumulator in the sense that it may hold an 8-bit (1-byte) value. The "B" register is only used by two 8051 instructions: MUL AB and DIV AB.

Aside from the MUL and DIV an instruction, the "B" register is often used as yet another temporary storage register much like a ninth "R" register.

Program Status Word

The PSW register contains program status information. It is a 8-bit flag register, out of 8-bits 6 bits are used and 2 bits are reserved. Out of 6 bits 4 bits are conditional bits and 2 bits are used for selecting register bank.

MSB							LSB
CY	AC	F0	RS1	RS0	OV	—	P

BIT	SYMBOL	FUNCTION
PSW.7	CY	Carry flag.
PSW.6	AC	Auxilliary Carry flag. (For BCD operations.)
PSW.5	F0	Flag 0. (Available to the user for general purposes.)
PSW.4	RS1	Register bank select control bit 1. Set/cleared by software to determine working register bank. (See Note.)
PSW.3	RS0	Register bank select control bit 0. Set/cleared by software todetermine working register bank. (See Note.)
PSW.2	OV	Overflow flag.
PSW.1	—	User-definable flag.
PSW.0	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.

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–17H)

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 locations 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.

Program Counter

The Program Counter (PC) is a 2-byte address which tells the 8051 where the next instruction to execute is found in memory. When the 8051 is initialized PC always starts at 0000h and is incremented each time an instruction is executed. It is important to note that PC isn't always incremented by one. Since some instructions require 2 or 3 bytes the PC will be incremented by 2 or 3 in these cases.

The Program Counter is special in that there is no way to directly modify its value. That is to say, we can't do something like PC=2430h. On the other hand, if we execute LJMP 2430h you've effectively accomplished the same thing.

Ports 0 to 3

P0, P1, P2, and P3 are the SFR latches of Ports 0, 1, 2, and 3, respectively. Writing a one to a bit of a port SFR (P0, P1, P2, or P3) causes the corresponding port output pin to switch high. Writing a zero causes the port output pin to switch low. When used as an input, the external state of a port pin will be held in the port SFR (i.e., if the external state of a pin is low, the corresponding port SFR bit will contain a 0; if it is high, the bit will contain a 1).

Serial Data Buffer

The Serial Buffer is actually two separate registers, a transmit buffer and a receive buffer. When data is moved to SBUF, it goes to the transmit buffer and 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 Basic to 80C51

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

Control Register for the 80C51

Special Function Registers IP, IE, TMOD, TCON, SCON, and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port.

Register Banks

The 8051 uses 8 "R" registers which are used in many of its instructions. These "R" registers are numbered from 0 through 7 (R0, R1, R2, R3, R4, R5, R6, and R7). These registers are generally used to assist in manipulating values and moving data from one memory location to another.

PSEN (Program Store Enable)

The 8051 has four dedicated bus control signals. It is a control signal that enables external program (code) memory. It usually connects to an EPROM's Output Enable (OE) pin to permit reading of program bytes.

The PSEN signal pulses low during the fetch stage of an instruction. When executing a program from internal ROM (8051/8052), PSEN remains in the inactive (high) state.

ALE (Address Latch Enable)

The 8051 similarly uses ALE for demultiplexing the address and data bus. When Port 0 is used in its alternate mode—as the data bus and the low-byte of the address bus—ALE is the signal that latches the address into an external register during the first half of a memory cycle.

EA (External Access)

The EA input signal is generally tied high (+5 V) or low (ground). If high, the 8051 executes programs from internal ROM when executing in the lower 4K of memory. If low, programs execute from external memory only (and PSEN pulses low accordingly).

RST (Reset)

The RST input is the master reset for the 8051. When this signal is brought high for at least two machine cycles, the 8051 internal registers are loaded with appropriate values for an orderly system start-up.

On-chip Oscillator Inputs

The 8051 features an on-chip oscillator. The nominal crystal frequency is 12 MHz for most ICs in the MCS-51™ family.

Memory Organization

Most microprocessors implement a shared memory space for data and programs. This is reasonable, since programs are usually stored on a disk and loaded into RAM for execution; thus both the data and programs reside in the system RAM. Microcontrollers have limited memory, and there is no disk drive or disk operating system. The control program must reside in. For this reason, the 8051 implements a separate memory space for programs (code) and data. Both the code and data may be internal; however, both expand using external components to a maximum of 64K code memory and 64K data memory.

The internal memory consists of on-chip ROM (8051/8052 only) and on-chip data RAM. ***The on-chip RAM contains a rich arrangement of general-purpose storage, bit-addressable storage, register banks, and special function registers.***

The internal memory space is divided between register banks (00H-1FH), bit-addressable RAM (20H-2FH), general-purpose RAM (30H-7FH), and special function registers (80H-FFH).

Any location in the general-purpose RAM can be accessed freely using the direct or indirect addressing modes.

Bit-addressable RAM

The 8051 contains 210 bit-addressable locations, of which 128 are at byte addresses 20H through 2FH, and the rest are in the special function registers.

The idea of individually accessing bits through software is a powerful feature of most microcontrollers. Bits can be set, cleared, ANDed, ORed, etc., with a single instruction.

Most microprocessors require a read-modify-write sequence of instructions to achieve the same effect. Furthermore, the 8051 I/O ports are bit-addressable, simplifying the software interface to single-bit inputs and outputs.

There are 128 general-purpose bit-addressable locations at byte address 20H through 2FH (8 bits/byte X 16 bytes = 128 bits).

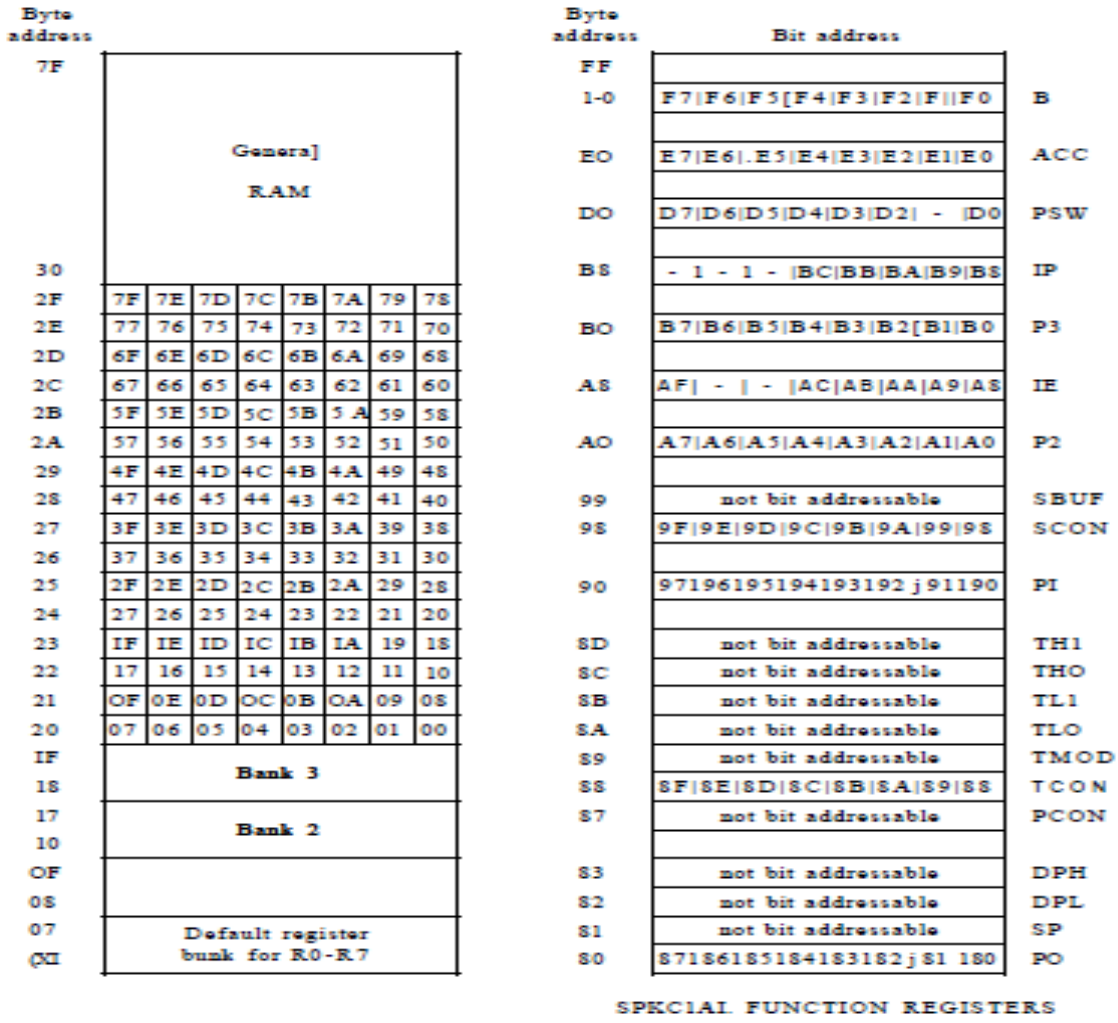
Register Banks

The bottom 32 locations of internal memory contain the register banks. The 8051 instruction set supports 8 registers, R0 through R7, and by default (after a system reset) these registers are at addresses OOH-07H.

Instructions using registers R0 to R7 are shorter and faster than the equivalent instructions using direct addressing. Data values used frequently should use one of these registers.

Special Function Registers

The 8051 internal registers are configured as part of the on-chip RAM; therefore, each register also has an address. This is reasonable for the 8051, since it has so many registers. As well as R0 to R7, there are 21 special function registers (SFRs) at the top of internal RAM, from addresses 80H to FFH.



Internal RAM memory

EXTERNAL MEMORY

The MCS-51 architecture provides expansion in the form of a 64K external code memory space and a 64K external data memory space. Extra ROM and RAM can be added as needed. Peripheral interface ICs can also be added to expand the I/O capability. These *become* part of the external data memory space using memory-mapped I/O.

When external memory is used, Port 0 is unavailable as an I/O port. It becomes a multiplexed address (A0-A7) and data (D0-D7) bus, with ALE latching the low-byte of the address at the beginning of each external memory cycle. Port 2 is usually (but not always) employed for the high-byte of the address bus.

Addressing Modes

The CPU can access data in various ways, which are called addressing modes

1. Immediate 2. Register 3. Direct 4.Register indirect 5.Indexed

Immediate Addressing

When a source operand is a constant rather than a variable then the constant can be incorporated into the instruction as a byte of "immediate" data. In assembly language, immediate operands are preceded by a number sign (#). The operand may be a numeric constant, a symbolic variable, or an arithmetic expression using constants, symbols, and operators. The assembler computes the value and substitutes the immediate data into the instruction.

For example, the instruction

MOV A, #12 ; loads the value 12 (OCH) into the accumulator. (It is assumed the constant "12" is in decimal notation, since it is not followed by "H.")

```
MOV A, #25H          ;    load 25H into A
MOV R4, #62          ;    load 62 into R4
MOV B, #40H          ;    load 40H into B
MOV DPTR, #4521H     ;    DPTR=4512H
MOV DPL, #21H        ;    This is the same
```

```
MOV DPH, #45H          ;    as above
MOV DPTR, #68975 ;     illegal!! Value > 65535 (FFFFH)
MOV P1, #55H           ;    immediate addressing mode can be used for ports also
```

Register Addressing Mode

Certain register names may be used as part of the opcode mnemonic as sources or destinations of data. Registers A, DPTR, and RO to R7 may be named as part of the opcode mnemonic.

```
MOV A, R0              ;    copy contents of R0 into A
MOV R2, A              ;    copy contents of A into R2
ADD A, R5              ;    add contents of R5 to A
ADD A, R7              ;    add contents of R7 to A
MOV R6, A              ;    save accumulator in R6
MOV DPTR, A            ;    will give an error
MOV DPTR,#25F5H ;
MOV R7, DPL                 ;
MOV R6, DPH            ;
MOV R4, R7             ;    The data transfer between Rn registers is not allowed
```

Direct Addressing Mode

All 128 bytes of internal RAM and the SFRs may be addressed directly using the single byte address assigned to each RAM location and each special-function register.

```
MOV R0, 40H            ;    save content of 40H in R0
MOV 56H, A             ;    save content of A in 56H
MOV A, 4               ;    is same as
MOV A, R4              ;    which means copy R4 into A
```

The SFR (Special Function Register) can be accessed by their names or by their addresses

```
MOV 0E0H, #55H         ;    is the same as
MOV A, #55h            ;    load 55H into A
MOV 0F0H, R0           ;    is the same as
MOV B, R0              ;    copy R0 into B
```

Note: The SFR registers have addresses between 80H and FFH. Not all the address space of 80 to FF is used by SFR. The unused locations 80H to FFH are reserved and must not be used by the 8051 programmer.

Register Indirect Addressing Mode

The indirect addressing mode uses a register to *hold* the actual address that will finally be used in the data move; the register itself is *not* the address, but rather the number in the register. Indirect addressing for MOV opcodes uses register R0 or R1, often called "data pointers," to hold the address of one of the data locations, which could be a RAM or an SFR address.

```
MOV A,@RO          ;    Copy the contents of the address in RO to the A register
MOV @R1, # 35h     ;    Copy the number 35h to the address in R1
MOV add,@R0            ;    Copy the contents of the address in RO to add
MOV @R1, A             ;    Copy the contents of A to the address in R1
MOV @R0, 80h           ;    Copy the contents of the port 0 pins to the address in R0
```

Note: The advantage is that it makes accessing data dynamic rather than static as in direct addressing mode. Looping is not possible in direct addressing mode.

Indexed Addressing

Indexed addressing uses a base register (either the program counter or the data pointer) and an offset (the accumulator) in forming the effective address for a JMP or MOVC instruction. Jump tables or look-up tables are easily created using indexed addressing.

```
MOVC A,@A+DPTR         ;    Copy the code byte, found at the ROM address
                            formed by adding A and the DPTR, to A
MOVC A,@A + PC         ;    Copy the code byte, found at the ROM address formed
                            by adding A and the PC, to A
```

I/O Ports:

The four 8-bit I/O ports P0, P1, P2 and P3 each uses 8 pins. All the ports upon RESET are configured as output, ready to be used as input ports.

Port 0 is also designated as AD0-AD7, allowing it to be used for both address and data. When connecting an 8051/31 to an external memory, port 0 provides both address and data. The 8051 multiplexes address and data through port 0 to save pins.

ALE indicates if P0 has address or data.
 When ALE=0, it provides data D0-D7
 When ALE=1, it has address A0-A7

In 8051-based systems with no external memory connection. Both P1 and P2 are used as simple I/O. In 8031/51-based systems with external memory connections, Port 2 must be used along with P0 to provide the 16-bit address for the external memory.
 P0 provides the lower 8 bits via A0 – A7
 P2 is used for the upper 8 bits of the 16-bit address, designated as A8 – A15, and it cannot be used for I/O Port 3 can be used as input or output.

Port 3 does not need any pull-up resistors. Port 3 has the additional function of providing some extremely important signals.

P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

Serial communications

External interrupts

Timers

Read/Write signals of external memories

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. To relieve the processor 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. The timers are used for (a) interval timing, (b) event counting, or (c) baud rate generation for the built-in serial port.

The 8051 has two timers/counters, they can be used either as:

Timers to generate a time delay or as Event counters to count events happening outside the microcontroller

In interval timing applications, a timer is programmed to overflow at a regular interval and set the timer overflow flag. The flag is used to synchronize the program to perform an action such as checking the state of inputs or sending data to outputs.

Other applications can use the regular clocking of the timer to measure the elapsed time between two conditions (e.g., pulse width measurements).

Event counting is used to determine the number of occurrences of an event, rather than to measure the elapsed time between events. An "event" is any external stimulus that provides a 1-to-0 transition to a pin on the 8051 IC. The timers can also provide the baud rate clock for the 8051's internal serial port.

TIMER MODE REGISTER (TMOD)

The TMOD register contains two groups of four bits that set the operating mode for Timer 0 and Timer 1. TMOD is not bit-addressable. Generally, it is loaded once by software at the beginning of a program to initialize the timer mode. In each case, the lower 2 bits are used to set the timer mode and the upper 2 bits to specify the operation.

(MSB) (LSB)

GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer1				Timer0			

M1	M0	Mode	Operating Mode
0	0	0	**13-bit timer mode** 8-bit timer/counter THx with TLx as 5-bit prescaler
0	1	1	**16-bit timer mode** 16-bit timer/counter THx and TLx are cascaded; there is no prescaler
1	0	2	**8-bit auto reload** 8-bit auto reload timer/counter; THx holds a value which is to be reloaded TLx each time it overfolws
1	1	3	**Split timer mode**

C/T:

Timer or counter selected, Cleared for timer operation (input from internal system clock) and Set for counter operation (input from Tx input pin).

GATE: Timers of 8051 do starting and stopping by either software or hardware control. In using software to start and stop the timer where GATE=0. The start and stop of the timer are controlled by way of software by the TR (timer start) bits TR0 and TR1.

The SETB instruction starts it, and it is stopped by the CLR instruction. These instructions start and stop the timers as long as GATE=0 in the TMOD register.

The hardware way of starting and stopping the timer by an external source is achieved by making GATE=1 in the TMOD register.

Timer Modes:

13-Bit Timer Mode (Mode 0)

Mode 0 is a 13-bit timer mode that provides compatibility with the 8051's predecessor, the 8048. It is not generally used in new designs. The timer high-byte (THx) is cascaded with the five least-significant bits of the timer low-byte (TLx) to form a 13-bit timer. The upper three bits of TLx are not used.

16-Bit Timer Mode (Mode 1)

Mode 1 is a 16-bit timer mode and is the same as mode 0, except the timer is operating as a full 16-bit timer. The clock is applied to the combined high and low timer registers (TLx/THx). As clock pulses are received, the timer counts up: OOOOH, 0001H, 0002H, etc. An overflow occurs on the FFFFH-to-OOOOH transition of the count and sets the timer overflow flag. The timer continues to count. The overflow flag is the TFx bit in TCON that is read or written by software.

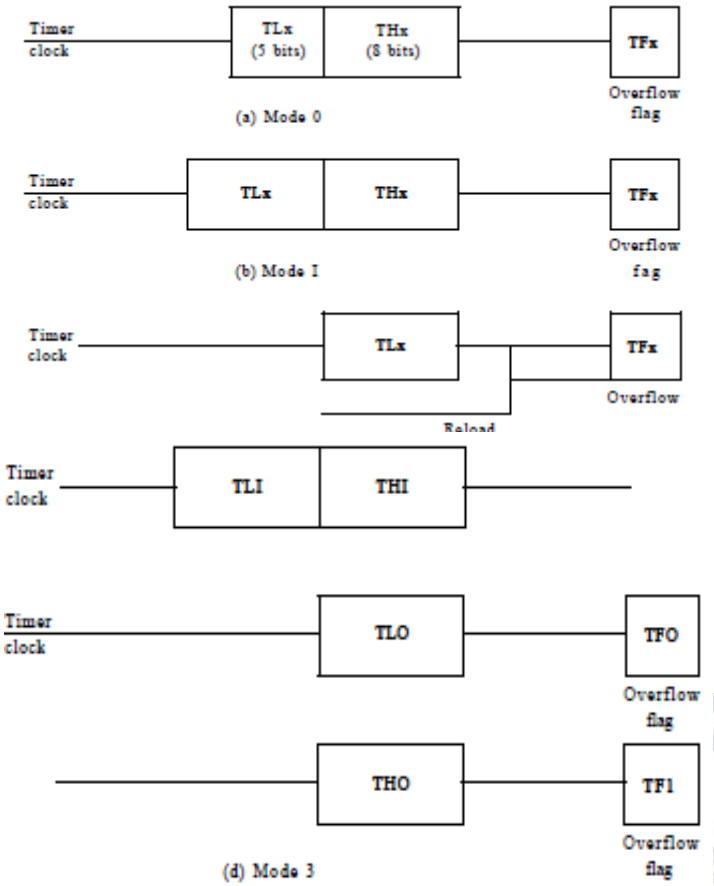
8-Bit Auto-Reload Mode (Mode 2)

Mode 2 is 8-bit auto-reload mode. The timer low-byte (TLx) operates as an 8-bit timer while the timer high-byte (THx) holds a reload value. When the count overflows from FFH to 00H, not only is the timer flag set, but the value in THx is loaded into TLx; counting continues from this value up to the next FFH-to-OOH transition, and so on. This mode is convenient, since timer overflows occur at specific, periodic intervals once TMOD and THx are initialized.

Split Timer Mode (Mode 3)

Mode 3 is the split timer mode and is different for each timer. Timer 0 in mode 3 is split into two 8-bit timers. **TLO and THO act as separate timers** with overflows setting the TFO and TF1 bits respectively.

Timer 1 is stopped in mode 3, but can be started by switching it into one of the other modes. The only limitation is that the usual Timer 1 overflow flag, TF1, is not affected by Timer 1 overflows, since it is connected to THO. Mode 3 essentially provides an extra 8-bit timer: The 8051 appears to have a third timer. 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. It can still be used by the serial port as a baud rate generator, or it can be used in any way not requiring interrupts (since it is no longer connected to TF1).



TCON Register

7	6	5	4	3	2	1	0
TF1	TR1	TFO	TRO	IE1	IT1	IE0	ITO

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.

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.

TFO 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.

TRO 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.

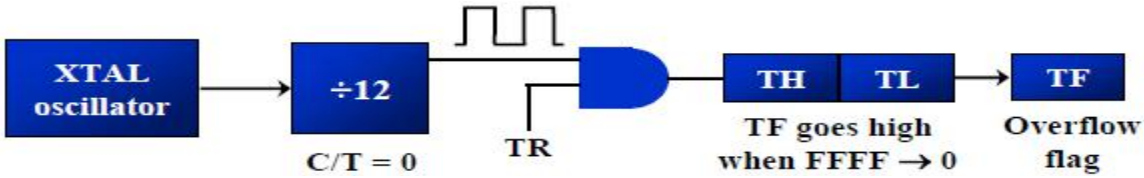
IE 1 External interrupt 1 edge flag. Set to 1 when a high to low edge signal is received on port 3 pin 3.3 (INTO. Cleared when processor vectors to interrupt service routine located at program address 0013h. Not related to timer operations.

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.

IE0 External interrupt 0 edge flag Set to 1 when a high to low edge signal is received on port 3 pin 3.2 (INTO). Cleared when processor vectors to interrupt service routine located at program address 0003h. Not related to timer operations.

The following are the characteristics and operations of mode1:

1. It is a 16-bit timer; therefore, it allows value of 0000 to FFFFH to be loaded into the timer's register TL and TH.
2. After TH and TL are loaded with a 16-bit initial value, the timer must be started. This is done by SETB TR0 for timer 0 and SETB TR1 for timer 1.
3. After the timer is started, it starts to count up. It counts up until it reaches its limit of FFFFH. When it rolls over from FFFFH to 0000, it sets high a flag bit called TF (timer flag).
4. Each timer has its own timer flag: TF0 for timer 0, and TF1 for timer 1. This timer flag can be monitored. When this timer flag is raised, one option would be to stop the timer with the instructions CLR TR0 or CLR TR1, for timer 0 and timer 1, respectively.



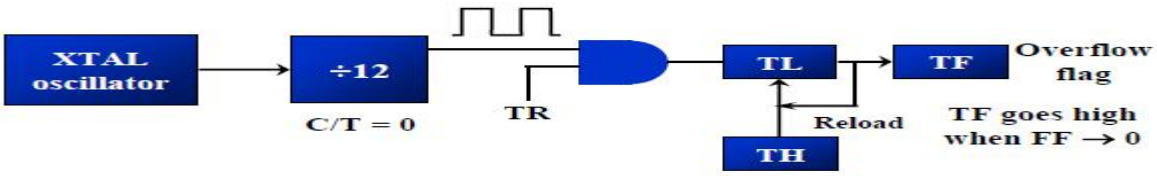
5. After the timer reaches its limit and rolls over, in order to repeat the process, TH and TL must be reloaded with the original value, and TF must be reloaded to 0.

To generate a time delay

1. Load the TMOD value register indicating which timer (timer 0 or timer 1) is to be used and which timer mode (0 or 1) is selected. Load registers TL and TH with initial count value
3. Start the timer
4. Keep monitoring the timer flag (TF) with the JNB TFx, target instruction to see if it is raised. Get out of the loop when TF becomes high
5. Stop the timer
6. Clear the TF flag for the next round
7. Go back to Step 2 to load TH and TL again.

The following are the characteristics and operations of mode 2:

1. It is an 8-bit timer; therefore, it allows only values of 00 to FFH to be loaded into the timer's register TH
2. After TH is loaded with the 8-bit value, the 8051 gives a copy of it to TL. Then the timer must be started. This is done by the instruction SETB TR0 for timer 0 and SETB TR1 for timer 1
3. After the timer is started, it starts to count up by incrementing the TL registers. It counts up until it reaches its limit of FFH. When it rolls over from FFH to 00, it sets high the TF (timer flag)



4. When the TL register rolls from FFH to 0 and TF is set to 1, TL is reloaded automatically with the original value kept by the TH register. To repeat the process, we must simply clear TF and let it go without any need by the programmer to reload the original value. This makes mode 2 an auto-reload, in contrast with mode 1 in which the programmer has to reload TH and TL.

To generate a time delay:

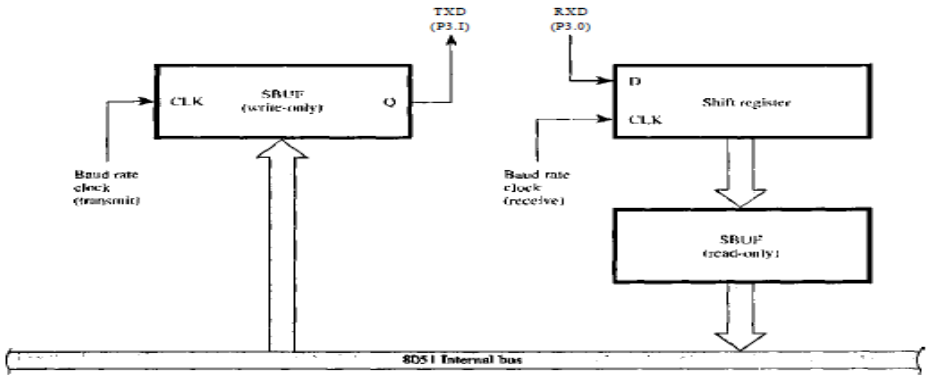
1. Load the TMOD value register indicating which timer (timer 0 or timer 1) is to be used, and the timer mode (mode 2) is selected.
2. Load the TH registers with the initial count value.
3. Start timer
4. Keep monitoring the timer flag (TF) with the JNB TFx, target instruction to see whether it is raised. Get out of the loop when TF goes high
5. Clear the TF flag
6. Go back to Step 4; since mode 2 is auto reload.

SERIAL PORT OPERATION

The 8051 includes an on-chip serial port that can operate in several modes over a wide range of frequencies. The essential function of the serial port is to perform parallel-to-serial conversion for output data, and serial-to-parallel conversion for input data.

The serial port features **Full duplex** operation (simultaneous transmission and reception), and **receive buffering allowing** one character to be received and held in a buffer while a second character is received. If the CPU reads the first character before the second is fully received, data are not lost.

Two special function registers provide software access to the serial port, SBUF and SCON. The serial port buffer (SBUF) at address 99H is really two buffers. Writing to SBUF loads data to be transmitted, and reading SBUF accesses received data. These are two separate and distinct registers, the transmit write-only register, and the receive read-only register.



Serial Port Block Diagram

The serial port control register (SCON) at address 98H is a bit-addressable register containing status bits and control bits. Control bits set the operating mode for the serial port, and status bits indicate the end of a character transmission or reception. The status bits are tested in software or programmed to cause an interrupt.

The serial port frequency of operation, or **baud rate,** can be fixed (derived from the 8051 on-chip oscillator) or variable. If a variable baud rate is used, Timer 1 supplies the baud rate clock and must be programmed accordingly.

SERIAL PORT CONTROL REGISTER (SCON)

The mode of operation of the 8051 serial port is set by writing to the serial port mode register (SCON) at address 99H.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial port mode specifier
SM1	SCON.6	Serial port mode specifier
SM2	SCON.5	Used for multiprocessor communication
REN	SCON.4	Set/cleared by software to enable/disable reception
TB8	SCON.3	Not widely used
RB8	SCON.2	Not widely used
TI	SCON.1	Transmit interrupt flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW
RI	SCON.0	Receive interrupt flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW

Note: Make SM2, TB8, and RB8 =0

INITIALIZATION AND ACCESSING SERIAL PORT REGISTERS

Receiver Enable

The receiver enable bit (REN) in SCON must be set by software to enable the reception of characters. This is usually done at the beginning of a program when the serial port, timers, etc., are initialized. This can be done in two ways. The instruction sets REN and sets or clears the other bits in SCON, as required.

SETB REN ; explicitly sets REN, or the instruction

Interrupt Flags

The receive and transmit interrupt flags (**RI and TI**) in SCON play an important role in 8051 serial communications. Both bits are set by hardware, but must be cleared by software.

Typically, **RI** is set at the end of character reception and indicates "receive buffer full." This condition is tested in software or programmed to cause an interrupt.

TI is set at the end of character transmission and indicates "transmit buffer empty." If software wishes to send a character to the device connected to the serial port, it must first check that the serial port is ready. In other words, if a previous character was sent, wait until transmission is finished before sending the next character.

MODES OF OPERATION

The 8051 serial port has four modes of operation, selectable by writing 1's or 0's into the SMO and SM1 bits in SCON. Three of the modes enable asynchronous communications, with each character received or transmitted framed by a start bit and a stop bit. In the fourth mode, the serial port operates as a simple shift register.

8-Bit Shift Register (Mode 0)

Mode 0, selected by writing 0's into bits SM1 and SM0 of SCON, puts the serial port into 8-bit shift register mode. Serial data enter and exit through RXD, and TXD outputs the shift clock. Eight bits are transmitted or received with the least-significant (LSB) first. The baud rate is fixed at 1/12th the on-chip oscillator frequency.

Transmission is initiated by any instruction that writes data to SBUF. Reception is initiated when the receiver enable bit (REN) is 1 and the receive interrupt bit (RI) is 0.

One possible application of shift register mode is to expand the output capability of the 8051.

8-Bit UART with Variable Baud Rate (Mode 1)

In **mode 1**, the 8051 serial port operates as an 8-bit UART with variable baud rate. In mode 1, 10 bits are transmitted on TXD or received on RXD. These consist of a start bit (always 0), eight data bits (LSB first), and a stop bit (always 1).

For a receive operation, the stop bit goes into RB8 in SCON. In the 8051, the baud rate is set by the Timer 1 overflow rate.

Transmission is initiated by writing to SBUF, but does not actually start until the next rollover of the divide-by-16 counter supplying the serial port baud rate. Shifted data are outputted on the TXD line beginning with the start bit, followed by the eight data bits, then the stop bit.

The period for each bit is the reciprocal of the baud rate as programmed in the timer. The transmit interrupt flag (TI) is set as soon as the stop bit appears on TXD. Reception is initiated by a 1-to-0 transition on RXD. The divide-by-16 counter is immediately reset to align the counts with the incoming bit stream (the next bit arrives on the next divide-by-16 rollover, and so on).

The incoming bit stream is sampled in the middle of the 16 counts. The receiver includes "false start bit detection" by requiring a 0 state eight counts after the first 1-to-0 transition. If this does not occur, it is assumed that the receiver was triggered by noise rather than by a valid character. The receiver is reset and returns to the idle state, looking for the next 1-to-0 transition.

Assuming a valid start bit was detected, character reception continues. The start bit is skipped and eight data bits are clocked into the serial port shift register. When all eight bits have been clocked in, the following occur:

1. The ninth bit (the stop bit) is clocked into RB8 in SCON,
2. SBUF is loaded with the eight data bits, and
3. The receiver interrupt flag (RI) is set.

9-Bit UART with Fixed Baud Rate (Mode 2)

When SM1 = 1 and SM0 = 0, the serial port operates in mode 2 as a 9-bit UART with a fixed baud rate. Eleven bits are transmitted or received: a start bit, eight data bits, a programmable ninth data bit, and a stop bit. On transmission, the ninth bit is whatever has been put in TB8 in SCON (perhaps a parity bit). On reception, the ninth bit received is placed in RB8. The baud rate in mode 2 is either 1/32nd or 1/64th the on-chip oscillator frequency.

Mode 3, 9-bit UART with variable baud rate

In **Mode 3**, is the same as mode 2 except the baud rate is programmable and provided by the timer. In fact, modes 1, 2, and 3 are very similar. The differences lie in the baud rates (fixed in mode 2, variable in modes 1 and 3) and in the number of data bits (eight in mode 1, nine in modes 2 and 3).

MULTIPROCESSOR COMMUNICATIONS

Modes 2 and 3 have a special provision for multiprocessor communications. In these modes, nine data bits are received and the ninth bit goes into RB8. The port can be programmed so that when the stop bit is received, the serial port interrupt is activated only if RB8 = 1. This feature is enabled by setting the SM2 bit in SCON. An application of this is in a networking environment using multiple 8051s in a master/slave arrangement.

SERIAL PORT BAUD RATES is fixed in modes 0 and 2. In mode 0 it is always the on-chip oscillator frequency divided by 12. Usually a crystal drives the 8051's on-chip oscillator, but another clock source can be used as well. Assuming a nominal oscillator frequency of 12 MHz, the mode 0 baud rate is 1 MHz

By default following a system reset, the mode 2 baud rate is the oscillator frequency divided by 64. The baud rate is also affected by a bit in the power control register, PCON.

There are two ways to increase the baud rate of data transfer

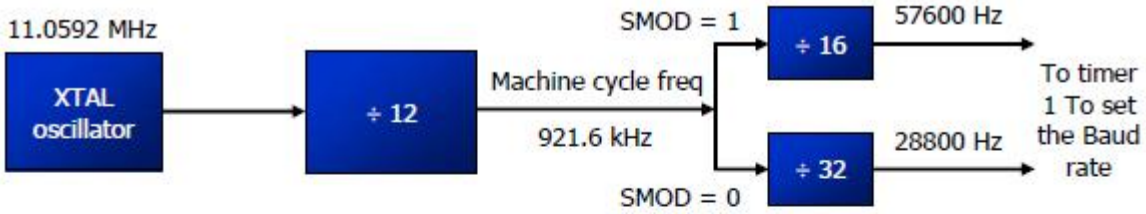
- To use a higher frequency crystal
- To change a bit in the PCON register

PCON register is an 8-bit register

When 8051 is powered up, SMOD is zero**.** We can set it to high by software and thereby double the baud rate

SMOD	--	--	--	GF1	GF0	PD	IDL

Bit 7 of PCON is the SMOD bit. Setting SMOD has the effect of doubling the baud rate in modes 1, 2, and 3. In mode 2, the baud rate can be doubled from a default value of l/64th the oscillator frequency (SMOD = 0), to l/32nd the oscillator frequency (SMOD = 1).



Baud Rate comparison for SMOD=0 and SMOD=1

TH1	(Decimal)	(Hex)	SMOD=0	SMOD=1
	-3	FD	9600	19200
	-6	FA	4800	9600
	-12	F4	2400	4800
	-24	E8	1200	2400

In programming the 8051 to transfer character bytes serially

1. TMOD register is loaded with the value 20H, indicating the use of timer 1 in mode 2 (8-bit auto-reload) to set baud rate
2. The TH1 is loaded with one of the values to set baud rate for serial data transfer
3. The SCON register is loaded with the value 50H, indicating serial mode 1, where an 8- bit data is framed with start and stop bits
4. TR1 is set to 1 to start timer 1
5. TI is cleared by CLR TI instruction
6. The character byte to be transferred serially is written into SBUF register
7. The TI flag bit is monitored with the use of instruction JNB TI, xx to see if the character has been transferred completely
8. To transfer the next byte, go to step 5

The steps that 8051 goes through in transmitting a character via TxD

1. The byte character to be transmitted is written into the SBUF register
2. The start bit is transferred
3. The 8-bit character is transferred on bit at a time
4. The stop bit is transferred. It is during the transfer of the stop bit that 8051 raises the TI flag, indicating that the last character was transmitted
5. By monitoring the TI flag, we make sure that we are not overloading the SBUF. If we write another byte into the SBUF before TI is raised, the untransmitted portion of the previous byte will be lost.

6. After SBUF is loaded with a new byte, the TI flag bit must be forced to 0 by CLR TI in order for this new byte to be transferred.
The TI bit can be checked by the instruction JNB TI, xx Using an interrupt

In programming the 8051 to receive character bytes serially

1. TMOD register is loaded with the value 20H, indicating the use of timer 1 in mode 2 (8-bit auto-reload) to set baud rate
2. TH1 is loaded to set baud rate
3. The SCON register is loaded with the value 50H, indicating serial mode 1, where an 8- bit data is framed with start and stop bits
4. TR1 is set to 1 to start timer 1
5. RI is cleared by CLR RI instruction
6. The RI flag bit is monitored with the use of instruction JNB RI, xx to see if an entire character has been received yet
7. When RI is raised, SBUF has the byte; its contents are moved into a safe place
8. To receive the next character, go to step 5

In receiving bit via its RxD pin, 8051 goes through the following steps

1. It receives the start bit. Indicating that the next bit is the first bit of the character byte it is about to receive
2. The 8-bit character is received one bit at time
3. The stop bit is received. When receiving the stop bit 8051 makes RI = 1 indicating that an entire character byte has been received and must be picked up before it gets overwritten by an incoming character
4. By checking the RI flag bit when it is raised, we know that a character has been received and is sitting in the SBUF register. We copy the SBUF contents to a safe place in some other register or memory before it is lost.
5. After the SBUF contents are copied into a safe place, the RI flag bit must be forced to 0 by CLR RI in order to allow the next received character byte to be placed in SBUF. Failure to do this causes loss of the received Character.

It must be cleared by the programmer with instruction CLR RI

If we copy SBUF into a safe place before the RI flag bit is raised, we risk copying garbage
The RI bit can be checked by the instruction JNB RI, xx using an interrupt

Architectural Features of 8096

SALIENT FEATURES:

- INTEL 8096, a second generation processor belongs to MCS 96 family. This is a high performance 16 bit microcontroller with register to register architecture.
- This is designed to handle high speed calculations and fast input/output operations which is preferred in high speed modern control applications.
- The 8096 with 16-bit CPU horse power, high speed math processing and high speed I/O is ideal for complex motor control and axis control systems. Hence it is used in 3 phase large horse power AC motors and robotics
- The 10-bit ADC option makes it most suitable candidate for data acquisition systems and closed loop analog controllers.
- 8096 can be configured in two modes.

(i) Single chip mode and (ii) Expanded mode.

In the single chip mode the internal ROM or EPROM is accessed by making the pin EA (Active low) HIGH. For ROM fewer chips to access the external memory the pin EA is made low.
In the expanded mode both internal and external (OFF CHIP) memory can be accessed using the multiplexed bus architecture.

- It has nearly 230 bytes of on-chip RAM and one 10-bit A/D converter with sample hold circuit.
- There are five on chip I/O ports each of 8-bit width.
- The 8096 bit microcontroller has a complete set of 16-bit arithmetic instructions including multiply and divide operations.
- It has Pulse Width Modulation Output with dedicated Baud Rate Generator.
- It has one on chip Full Duplex Serial Port.

- There are 20 interrupt sources and 8 interrupt vectors on 8096. It has two 16-bit Timers. Timer 1 and Timer 2 and one 16 bit watch dog timer.