MICROCONTROLLER

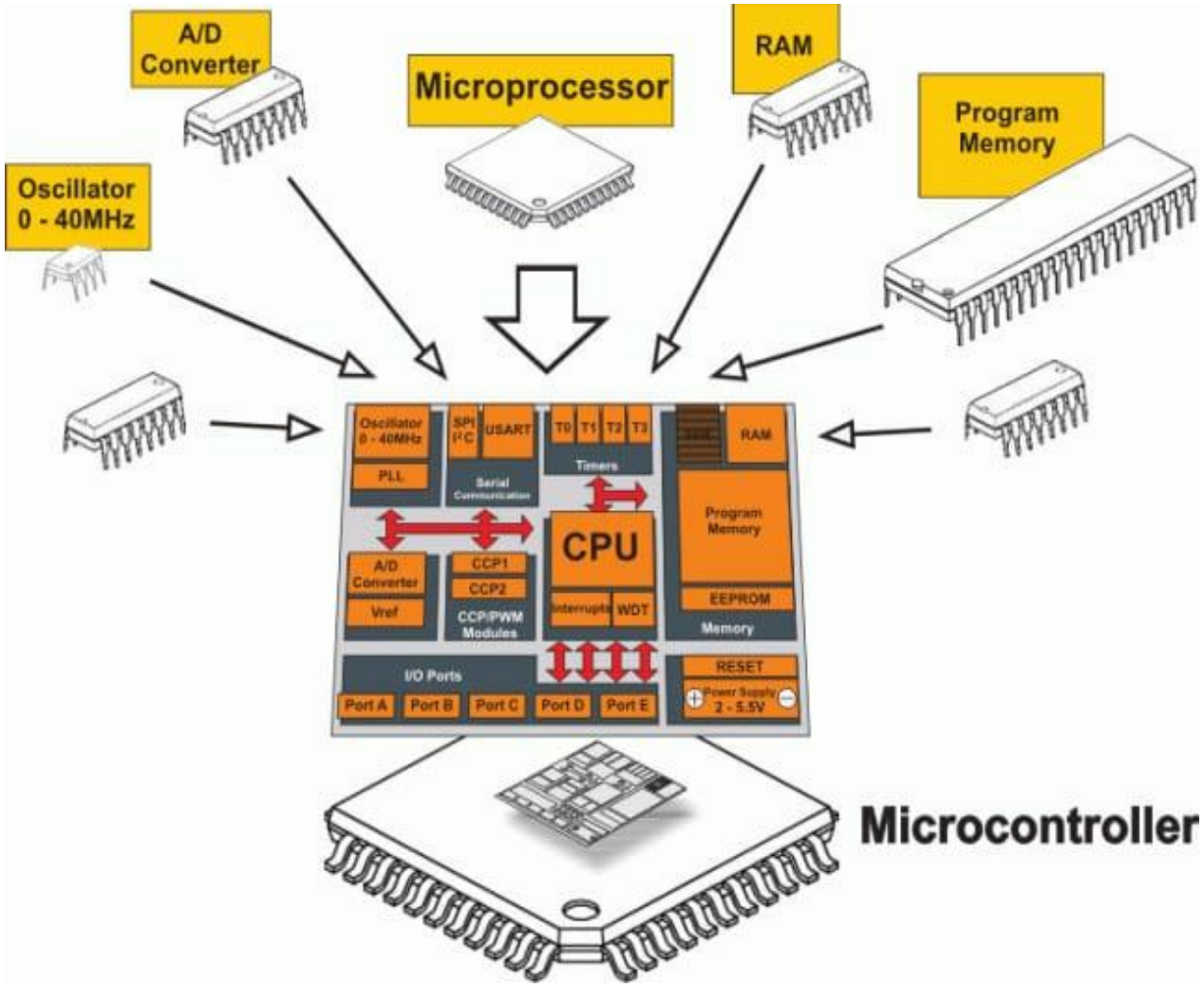
UNIT III

MR.UMESH KOYANDE

MR.KIRAN DATAR

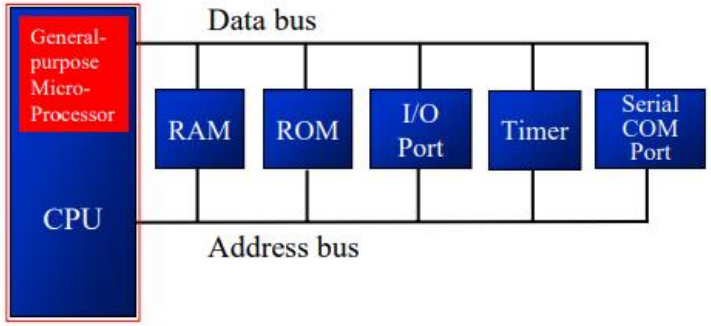
CONTENTS

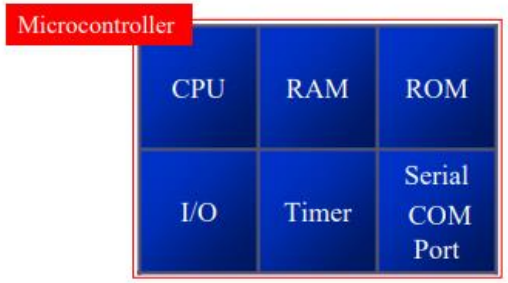
- Microcontrollers and Embedded processors
- Overview of 8051 family
- 8051 Microcontroller hardware
- Input/output pins
- Ports
- Circuits
- External Memory.



MICROCONTROLLERS AND EMBEDDED PROCESSORS

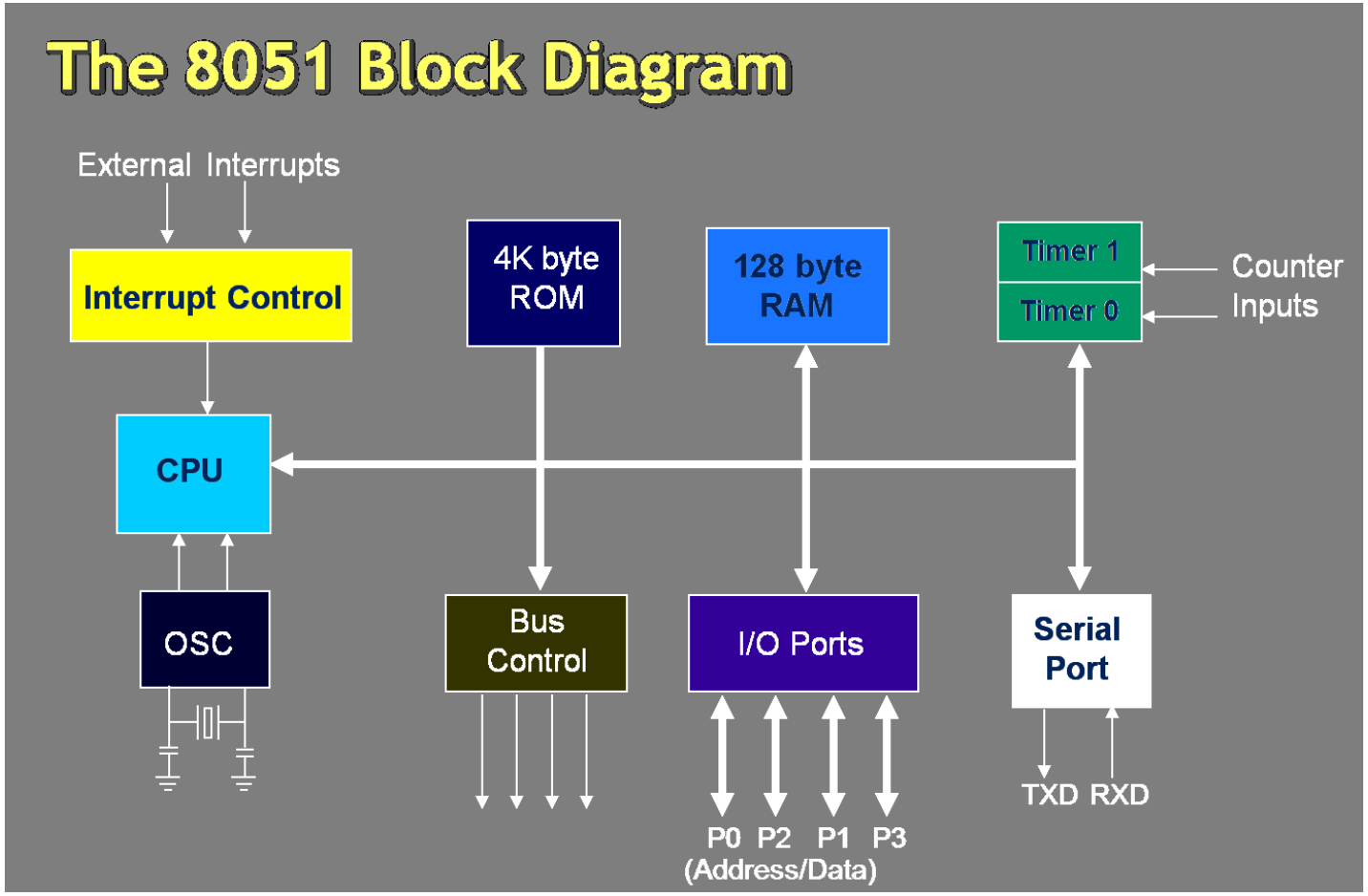
- Microcontroller has
  - CPU (microprocessor)
  - RAM
  - ROM
  - I/O ports
  - Timer
  - ADC and other peripherals



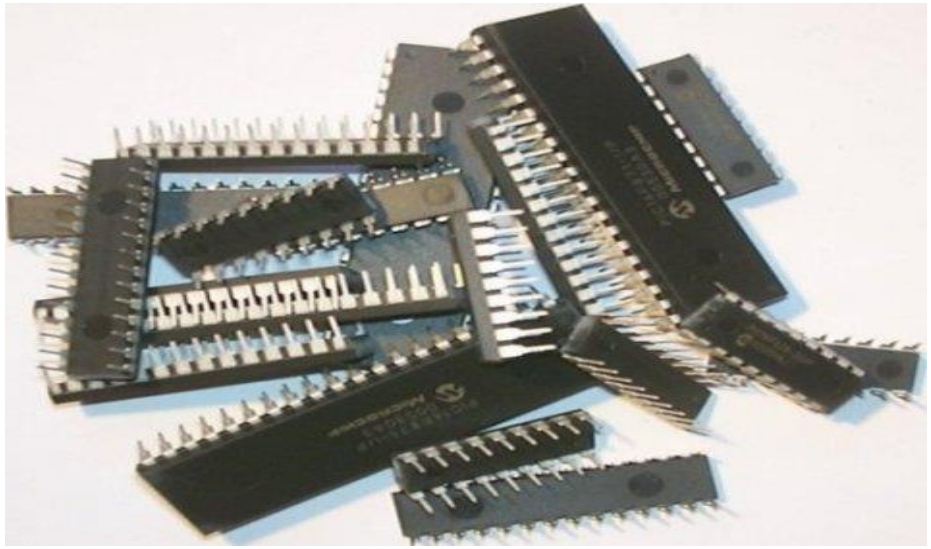


MICROCONTROLLERS AND EMBEDDED PROCESSORS

MICROCONTROLLERS AND EMBEDDED PROCESSORS



MICROCONTROLLERS AND EMBEDDED PROCESSORS



- Criteria for Choosing a Microcontroller
  - Speed
  - Packaging
  - Power consumption
  - Amount of RAM and ROM on chip
  - Number of I/O pins and the timer on chip
  - Easy to upgrade
  - Versions
  - Cost per unit

COMPARISON OF THE 8051 FAMILY MEMBERS

89XX	ROM	RAM	Timer	Int Source	IO pin	Other
8951	4k	128	2	6	32	-
8952	8k	256	3	8	32	-
8953	12k	256	3	9	32	WD
8955	20k	256	3	8	32	WD
898252	8k	256	3	9	32	ISP
891051	1k	64	1	3	16	AC
892051	2k	128	2	6	16	AC

WD: Watch Dog Timer
AC: Analog Comparator
ISP: In System Programable

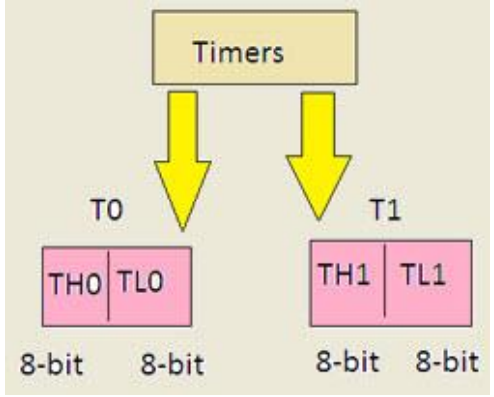
MICROCONTROLLERS AND EMBEDDED PROCESSORS

- 12 MHz clock
- 8 bit ALU with internal bus width of 8 bit
- CISC architecture
- Special instruction to manipulate individual bits
- Program counter
- Stack pointer
- Harvard memory architecture

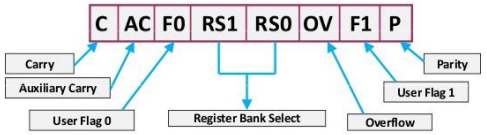
MICROCONTROLLERS AND EMBEDDED PROCESSORS

- RAM of 128b bytes
- 32 bytes of RAM are also used as four bank of registers
- Special Function Register(SFR's)
- Two external interrupt pins INT0 and INT1
- Four ports of 8 bits each in single chip mode
- Two timers T0 and T1

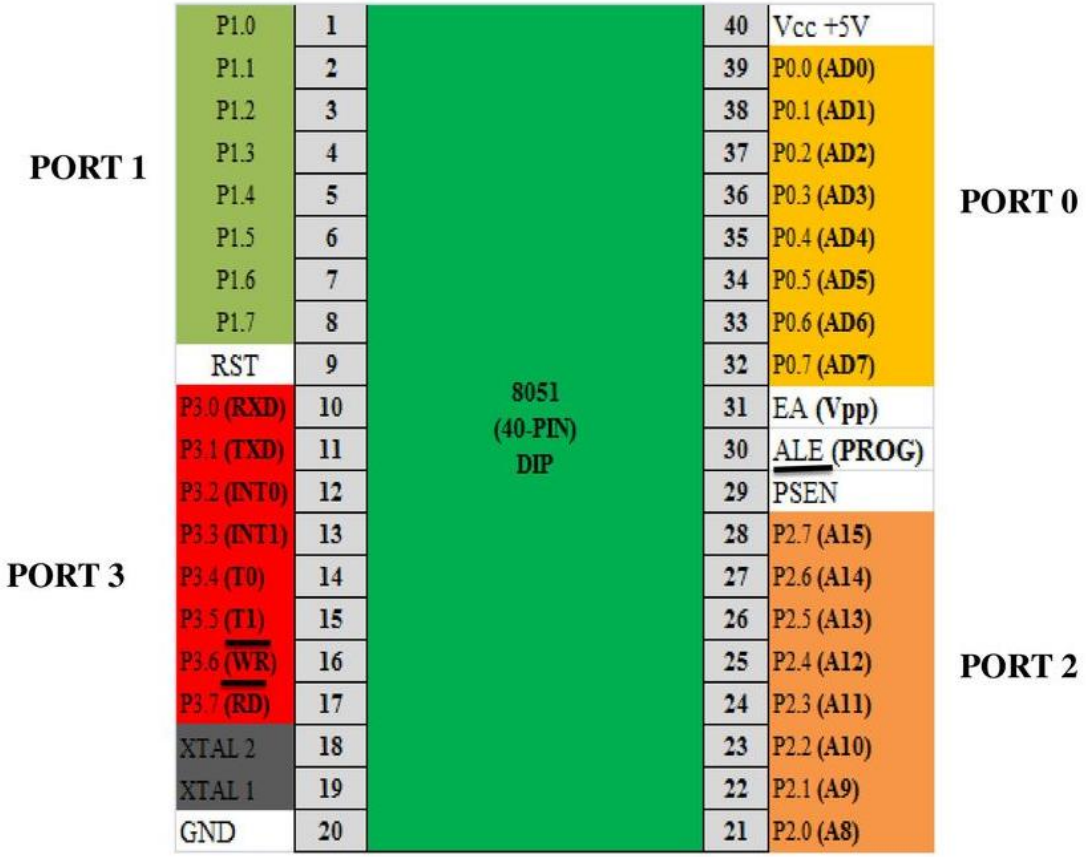
	Register bank 0		Register bank 1		Register bank 2		Register bank 3
00	R0	08	R0	10	R0	18	R0
01	R1	09	R1	11	R1	19	R1
02	R2	0A	R2	12	R2	1A	R2
03	R3	0B	R3	13	R3	1B	R3
04	R4	0C	R4	14	R4	1C	R4
05	R5	0D	R5	15	R5	1D	R5
06	R6	0E	R6	16	R6	1E	R6
07	R7	0F	R7	17	R7	1F	R7



Program Status Word



8051 Pin - out



INPUT / OUTPUT PORTS

Port 0 （ pins 32-39 ）
P0 (P0.0 ~ P0.7)

- 8-bit R/W - General Purpose I/O
- Multiplexed low byte address and data bus for external memory design

Port 1 （ pins 1-8 P1
(P1.0 ~ P1.7)

- Only 8-bit R/W - General Purpose I/O

Port 2 （ pins 21-28 ）
P2 (P2.0 ~ P2.7)

- 8-bit R/W - General Purpose I/O
- High byte of the address bus for external memory design

Port 3 （ pins 10-17 ）
P3 (P3.0 ~ P3.7)

- General Purpose I/O
- Timers or external interrupts.

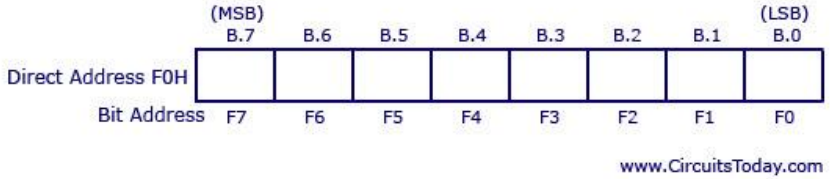
SPECIAL FUNCTION REGISTER

F8								FF
F0	**B**							F7
E8								EF
E0	**ACC**							E7
D8								DF
D0	**PSW**					SPCR		D7
C8	**T2CON**	T2MOD	RCAP2L	RCAP2H	TL2	TH2		CF
C0								C7
B8	**IP**	SADEN						BF
B0	**P3**							IPH B7
A8	**IE**	SADDR	SPSR					AF
A0	**P2**						WDTRST	WDTCON A7
98	**SCON**	SBUF						9F
90	**P1**						EECON	97
88	**TCON**	TMOD	TL0	TL1	TH0	TH1	AUXR	CLKREG 8F
80	**P0**	SP	DP0L	DP0H	DP1L	DP1H	SPDR	PCON 87

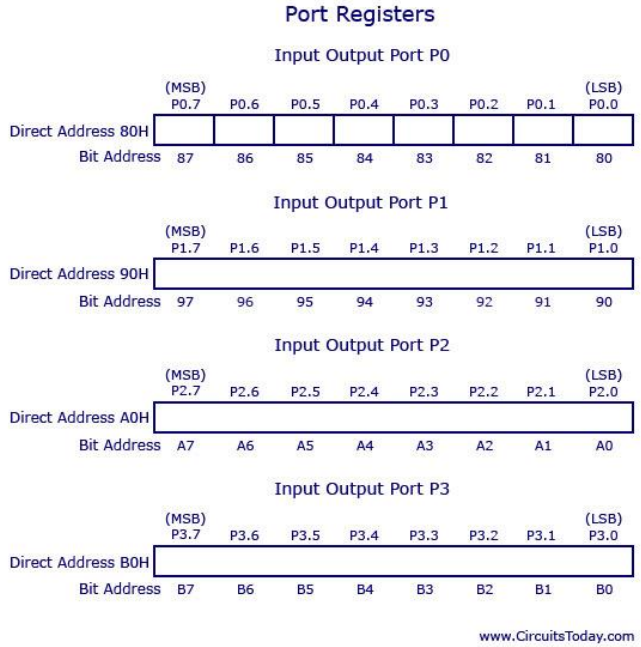
Bit adresibilni registri

SFR IN 8051

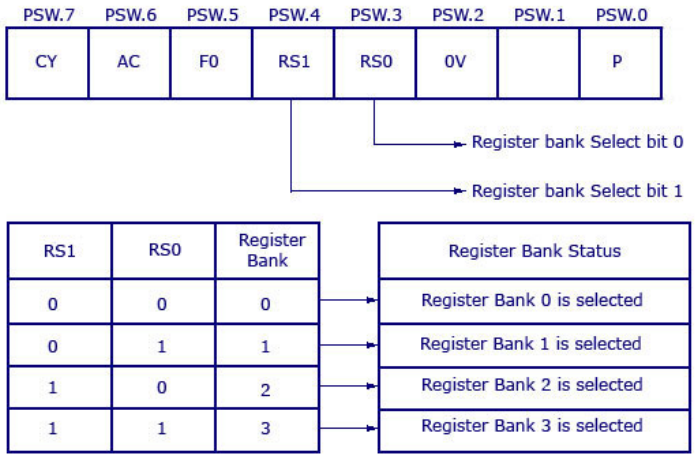
Register B



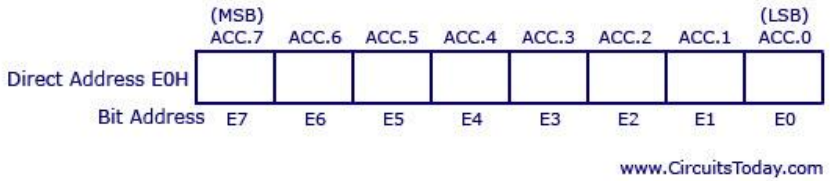
Port Registers



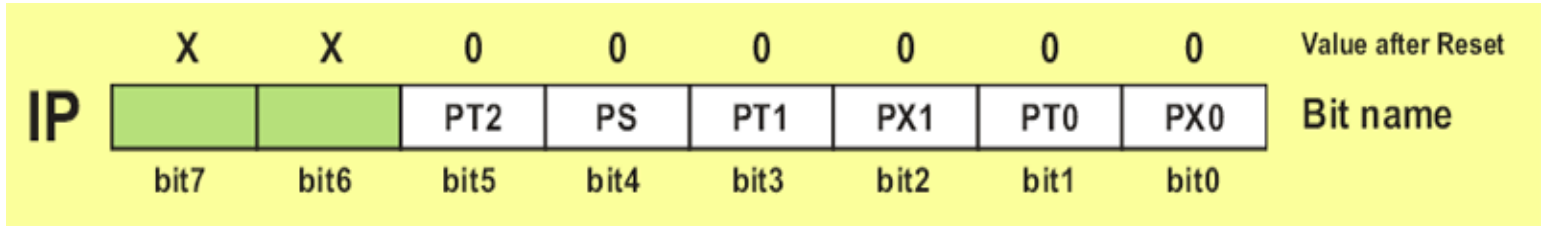
Processor Status Word

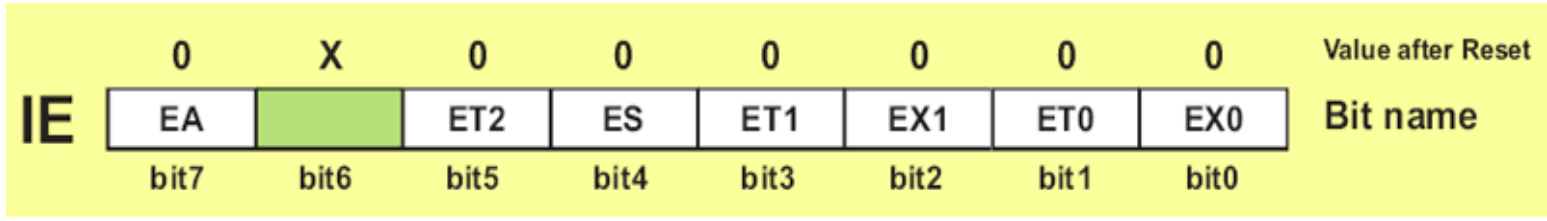


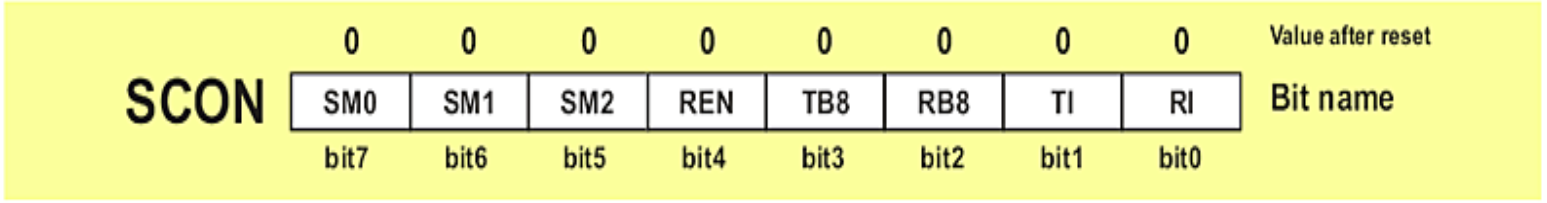
Accumulator Register A

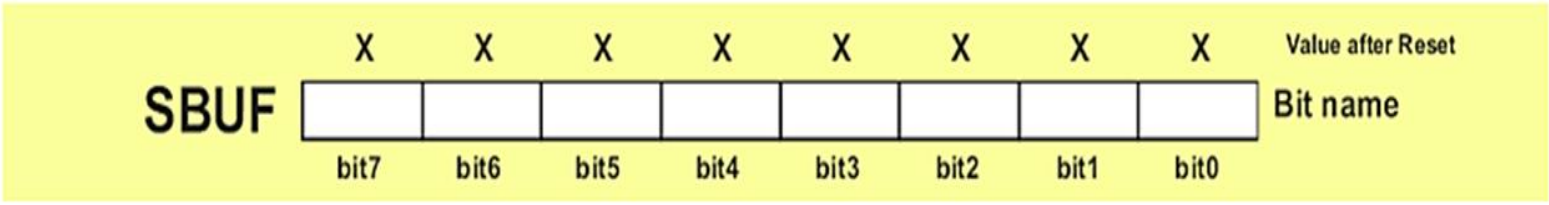


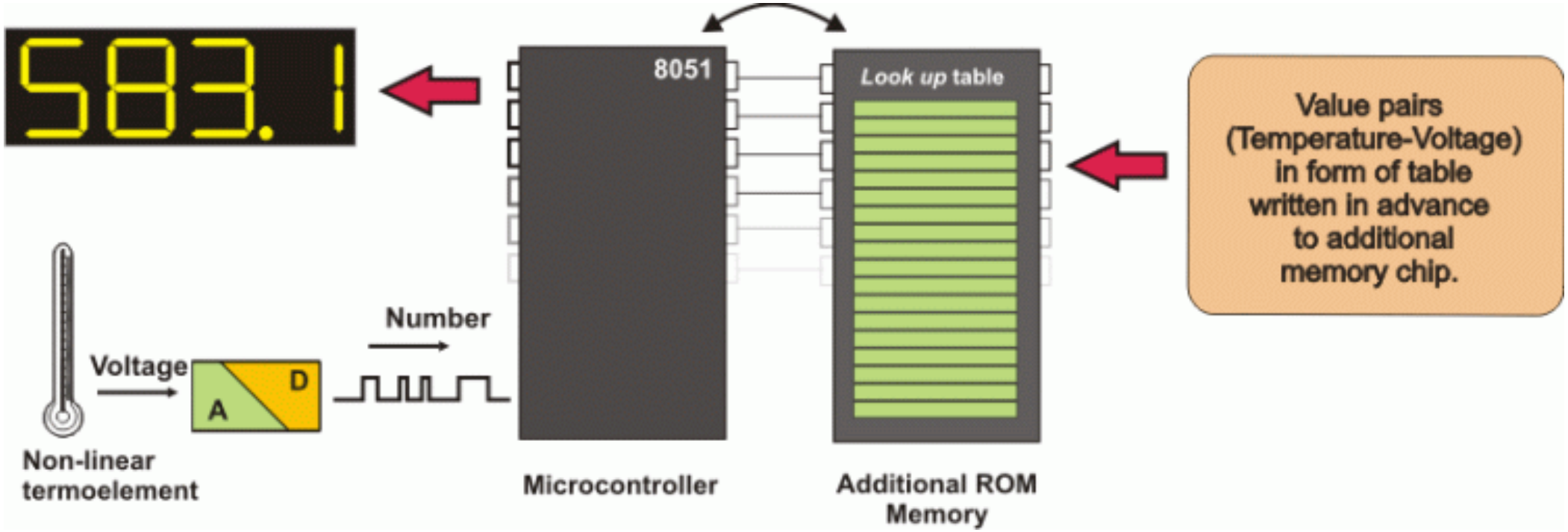
SFR IN 8051


IP
X X 0 0 0 0 0 0
Value after Reset
PT2 PS PT1 PX1 PT0 PX0
Bit name
bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0

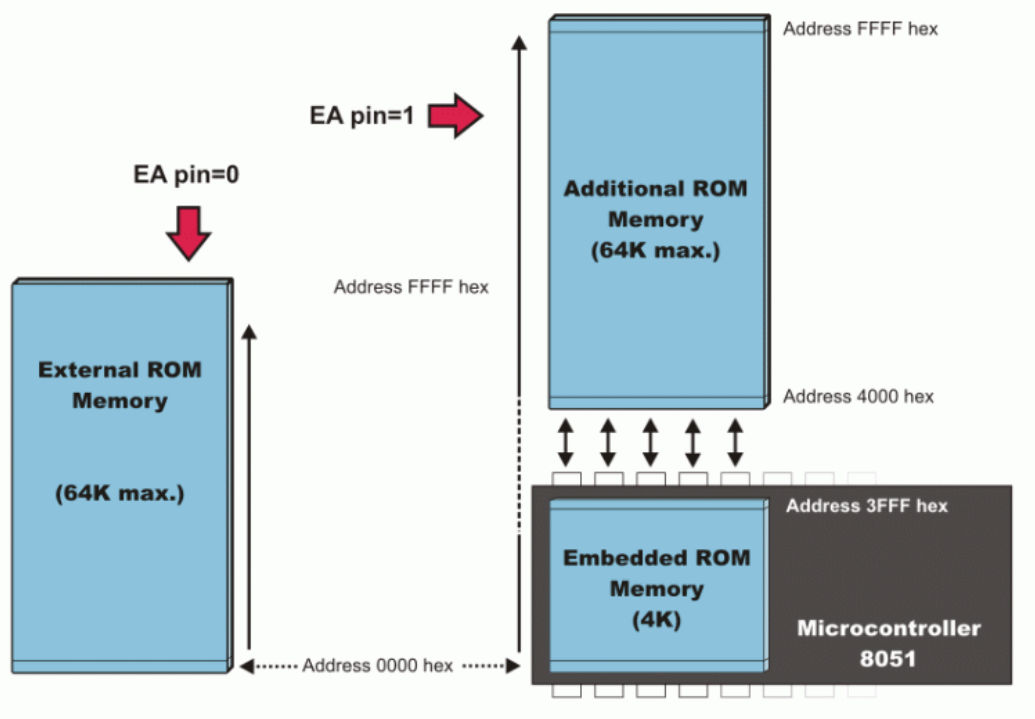

IE
0 X 0 0 0 0 0 0
Value after Reset
EA ET2 ES ET1 EX1 ET0 EX0
Bit name
bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0

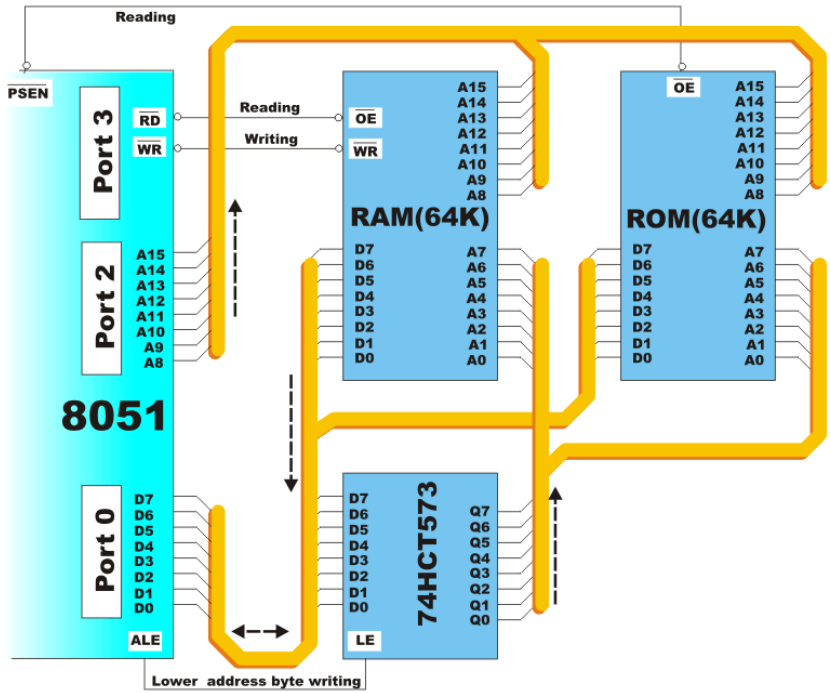

SCON
0 0 0 0 0 0 0 0
Value after reset
SM0 SM1 SM2 REN TB8 RB8 TI RI
Bit name
bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0


SBUF
X X X X X X X X
Value after Reset
Bit name
bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0



EXTERNAL MEMORY





EXTERNAL MEMORY

8051 PROGRAMMING IN C

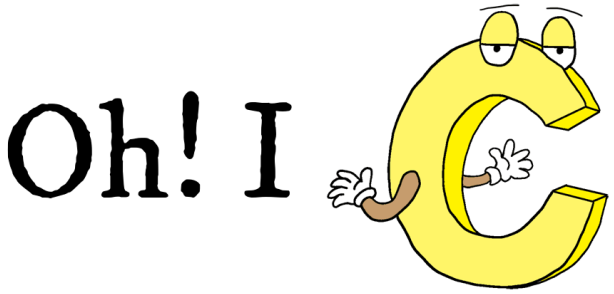
Why Program 8051 In C?

Compilers produce hex files that is downloaded to ROM of microcontroller

The size of hex file is the main concern

- Microcontrollers have limited on-chip ROM
- Code space for 8051 is limited to 64K bytes

C programming is less time consuming, but has larger hex file size



8051 PROGRAMMING IN C

- The reasons for writing programs in C
  - Easier and less time consuming to write in C than Assembly
  - C is easier to modify and update
  - Use code available in function libraries
  - C code is portable to other microcontroller with little of no modification

DATA TYPES

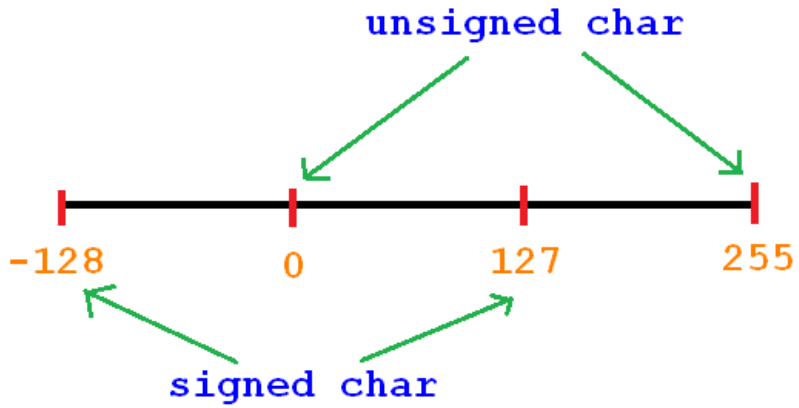
- Unsigned char
- Signed char
- Unsigned int
- Signed int
- Sbit (single bit)
- Bit and sfr

Sr.No.	Data Type	Size in bits	Data Range/Usage
1	unsigned char	unsigned char	0 to 255
2	signed char	8 bit	-128 to +127
3	unsigned int	16 bit	0 to 65535
4	signed int	16 bit	-32,768 to +32767
5	sbit	1- Bit	SFR bit addressable only
6	bit	1- Bit	RAM bit addressable only

DATA TYPES

Unsigned char

- Unsigned char is an 8-bit data type in the range of 0 – 255 (00 – FFH)
  - Counter value
  - ASCII characters
- C compilers use the signed char as the default if we do not put the keyword unsigned



DATA TYPES

Write an 8051 C program to send values 00 – FF to port P1.

Solution:

```
#include <reg51.h>
void main(void)
  {
    unsigned char z;
    for (z=0;z<=255;z++)
       P1=z;
  }
```

1. Pay careful attention to the size of the data
2. Try to use unsigned *char* instead of *int* if possible

Write an 8051 C program to send hex values for ASCII characters of 0, 1, 2, 3, 4, 5, A, B, C, and D to port P1.

Solution:

```
#include <reg51.h>
void main(void)
  {
    unsigned char mynum[]="012345ABCD";
    unsigned char z;
    for (z=0;z<=10;z++)
       P1=mynum[z];
  }
```

Write an 8051 C program to toggle all the bits of P1 continuously.

Solution:

```
//Toggle P1 forever
#include <reg51.h>
void main(void)
  {
    for (;;)
      {
        p1=0x55;
        p1=0xAA;
      }
  }
```

Write an 8051 C program to send values of –4 to +4 to port P1.

Solution:

```
//Singed numbers
#include <reg51.h>
void main(void)
  {
    char mynum[]={+1,-1,+2,-2,+3,-3,+4,-4};
    unsigned char z;
    for (z=0;z<=8;z++)
       P1=mynum[z];
  }
```

Write an 8051 C program to toggle bit D0 of the port P1 (P1.0) 50,000 times.

Solution:

sbit keyword allows access to the single bits of the SFR registers

```
#include <reg51.h>
sbit MYBIT=P1^0;

void main(void)
  {
    unsigned int z;
    for (z=0;z<=50000;z++)
       {
         MYBIT=0;
         MYBIT=1;
       }
  }
```

DATA TYPES

TIME DELAY

Write an 8051 C program to toggle bits of P1 continuously forever with some delay.

Solution:

```
//Toggle P1 forever with some delay in between
//"on" and "off"
#include <reg51.h>
void main(void)
  {
    unsigned int x;
    for (;;)                    //repeat forever
      {
        p1=0x55;
        for (x=0;x<40000;x++); //delay size
                               //unknown
        p1=0xAA;
        for (x=0;x<40000;x++);
      }
  }
```

We must use the oscilloscope to measure the exact duration

Write an 8051 C program to toggle bits of P1 ports continuously with a 250 ms.

Solution:

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
  {
    while (1)                   //repeat forever
      {
        p1=0x55;
        MSDelay(250);
        p1=0xAA;
        MSDelay(250);
      }
  }

void MSDelay(unsigned int itime)
  {
    unsigned int i,j;
    for (i=0;i<itime;i++)
      for (j=0;j<1275;j++);
  }
```

I/O PROGRAMMING

LEDs are connected to bits P1 and P2. Write an 8051 C program that shows the count from 0 to FFH (0000 0000 to 1111 1111 in binary) on the LEDs.

Solution:

```
#include <reg51.h>
#defind LED P2;

void main(void)
  {
    P1=00;              //clear P1
    LED=0;              //clear P2
    for (;;)            //repeat forever
       {
         P1++;          //increment P1
         LED++;         //increment P2
       }
  }
```

Ports P0 – P3 are byte-accessable and we use the P0 – P3 labels as defined in the 8051/52 header file.

I/O PROGRAMMING

Write an 8051 C program to get a byte of data form P1, wait 1/2 second, and then send it to P2.

Solution:

```
#include <reg51.h>
void MSDelay(unsigned int);

void main(void)
  {
    unsigned char mybyte;
    P1=0xFF;                    //make P1 input port
    while (1)
       {
         mybyte=P1;             //get a byte from P1
         MSDelay(500);
         P2=mybyte;             //send it to P2
       }
  }
```

I/O PROGRAMMING

Write an 8051 C program to get a byte of data form P0. If it is less than 100, send it to P1; otherwise, send it to P2.

Solution:

```
#include <reg51.h>

void main(void)
  {
    unsigned char mybyte;
    P0=0xFF;                  //make P0 input port
    while (1)
       {
         mybyte=P0;           //get a byte from P0
         if (mybyte<100)
            P1=mybyte;        //send it to P1
         else
            P2=mybyte;        //send it to P2
       }
  }
```

LOGIC OPERATIONS

Logical operators

- AND (&&), OR (||), and NOT (!)

Bit-wise operators

- AND (&), OR (|), EX-OR (^), Inverter (~),Shift Right (>>), and Shift Left (<<)
  - These operators are widely used in software engineering for embedded systems and control

Write an 8051 C program to toggle all the bits of P0 and P2 continuously with a 250 ms delay. Using the inverting and Ex-OR operators, respectively.

Solution:

```
#include <reg51.h>
void MSDelay(unsigned int);

void main(void)
  {
    P0=0x55;
    P2=0x55;
    while (1)
       {
         P0=~P0;
         P2=P2^0xFF;
         MSDelay(250);
       }
  }
```

Run the following program on your simulator and examine the results.

Solution:

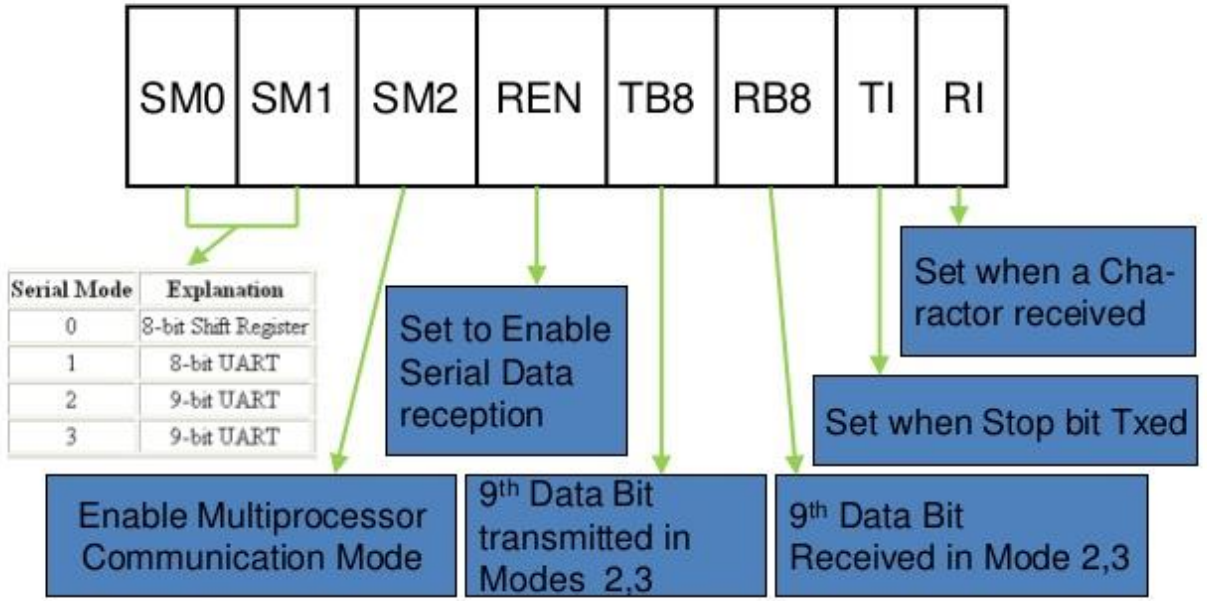
```
#include <reg51.h>

void main(void)
  {
    P0=0x35 & 0x0F;          //ANDing
    P1=0x04 | 0x68;          //ORing
    P2=0x54 ^ 0x78;          //XORing
    P0=~0x55;                //inversing
    P1=0x9A >> 3;            //shifting right 3
    P2=0x77 >> 4;            //shifting right 4
    P0=0x6 << 4;             //shifting left 4
  }
```

LOGIC OPERATIONS

SERIAL COMMUNICATION

Serial Port Control (SCON) Register



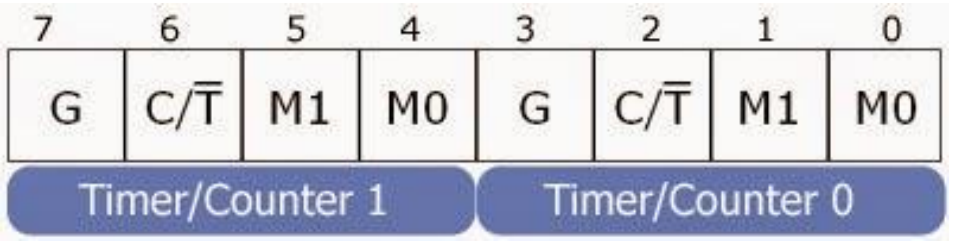
Serial Mode	Explanation
0	8-bit Shift Register
1	8-bit UART
2	9-bit UART
3	9-bit UART

(MSB) (LSB)

SMO	SM1	SM2	REN	TB8	RB8	TI	RI
0	1	0	1	0	0	0	0

SCON = 0x50

SERIAL COMMUNICATION



0	0	1	0	0	0	0	0

TMOD = 0x20

M1 and M0 specify the mode as follows:

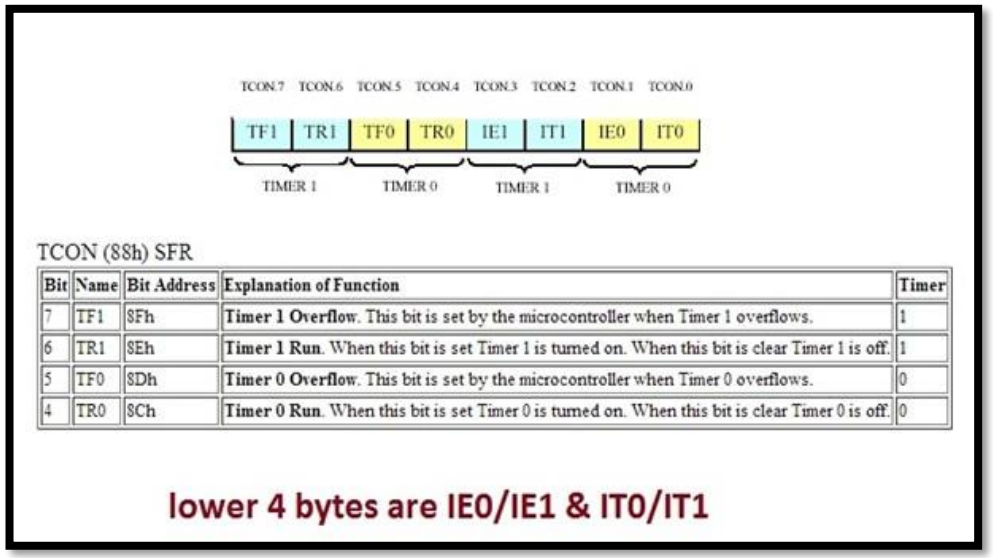
M1	M0	Mode	*Description in brief*
0	0	0	13-bit counter
0	1	1	16-bit counter
1	0	2	8-bit counter with autoreload
1	1	3	Split Timer 0 into two 8-bit counters or to stop Timer 1

For 9600 baud rate:
TH1 = 256 – ((Crystal / 384) / Baud)
TH1 = 256 – ((11.59MHz/ 384) / 9600)
TH1 = 256 – 3 = 253

Hex value of 253 is FD which should be loaded to TH1

SERIAL COMMUNICATION

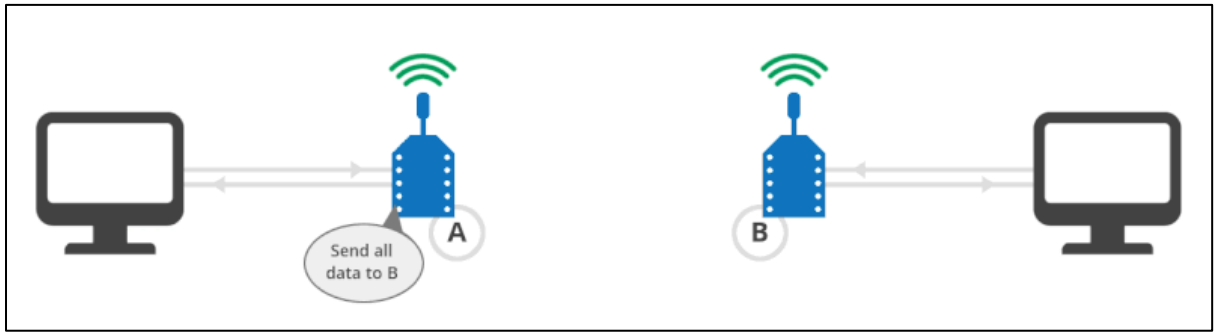
```
void main()
{
SCON=0x50;    //starting of a serial communication//
TMOD=0x20;    //selected the timer mode//
TH1=FD;         // load the baud rate//
TR1=1;          //Timer is ON//
SBUF='S';       //store the character inside a register//
while(TI==0);  //check the interrupt register//
TI=0;
TR1=0;          //OFF the timer//
while(1);       //continuous loop//
}
```



TCON (88h) SFR

Bit	Name	Bit Address	Explanation of Function	Timer
7	TF1	8Fh	**Timer 1 Overflow**. This bit is set by the microcontroller when Timer 1 overflows.	1
6	TR1	8Eh	**Timer 1 Run**. When this bit is set Timer 1 is turned on. When this bit is clear Timer 1 is off.	1
5	TF0	8Dh	**Timer 0 Overflow**. This bit is set by the microcontroller when Timer 0 overflows.	0
4	TR0	8Ch	**Timer 0 Run**. When this bit is set Timer 0 is turned on. When this bit is clear Timer 0 is off.	0

lower 4 bytes are IE0/IE1 & IT0/IT1



DELAY USING TIMER

Divide the time delay with timer clock period.

NNNN=time delay/1.085μs

Subtract the resultant value from 65536.

MMMM=65536-NNNN

Convert the difference value to the hexa decimal form.

MMMMd = XXYYh

Load this value to the timer register.

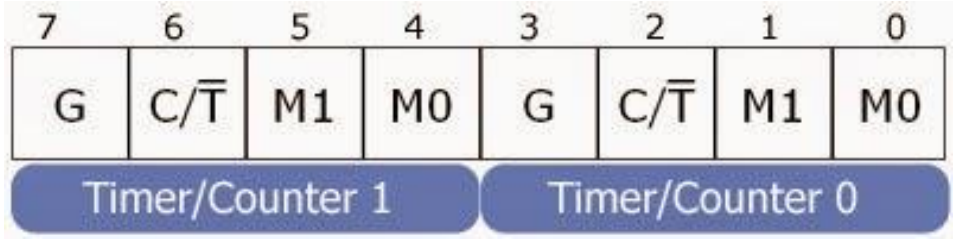
TH=XXh

TL=YYh

Delay Function to Generate 1 ms Delay

1. NNNN = 1ms/1.085μs ≈ 922.
2. MMMM = 65536-922 = 64614
3. 64614 in Hexadecimal = FC66h
4. Load
   - TH with 0xFC
   - TL with 0x66

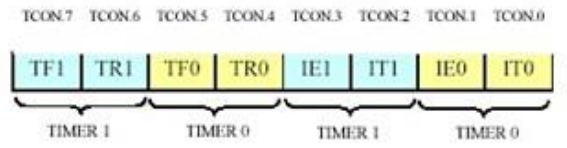
DELAY USING TIMER



0	0	0	0	0	0	0	1

TMOD = 0x01

```
   Void delay ()
{
   TMOD = 0x01;    // Timer 0 Mode 1
   TH0= 0xFC;      //initial value for 1ms
   TL0 = 0x66;
   TR0 = 1;        // timer start
   while (TF0 == 0); // check overflow condition
   TR0 = 0;        // Stop Timer
   TF0 = 0;        // Clear flag
   }
```



TCON (88h) SFR

Bit	Name	Bit Address	Explanation of Function	Timer
7	TF1	8Fh	**Timer 1 Overflow**. This bit is set by the microcontroller when Timer 1 overflows.	1
6	TR1	8Eh	**Timer 1 Run**. When this bit is set Timer 1 is turned on. When this bit is clear Timer 1 is off.	1
5	TF0	8Dh	**Timer 0 Overflow**. This bit is set by the microcontroller when Timer 0 overflows.	0
4	TR0	8Ch	**Timer 0 Run**. When this bit is set Timer 0 is turned on. When this bit is clear Timer 0 is off.	0

lower 4 bytes are IE0/IE1 & IT0/IT1


Thank
You