MICROPROCESSORS AND MICROCONTROLLERS

Dr. P. Lachi Reddy

Professor in ECE

LBRCE, Mylavaram

UNIT-II

8051 - MICROCONTROLLER

TOPICS

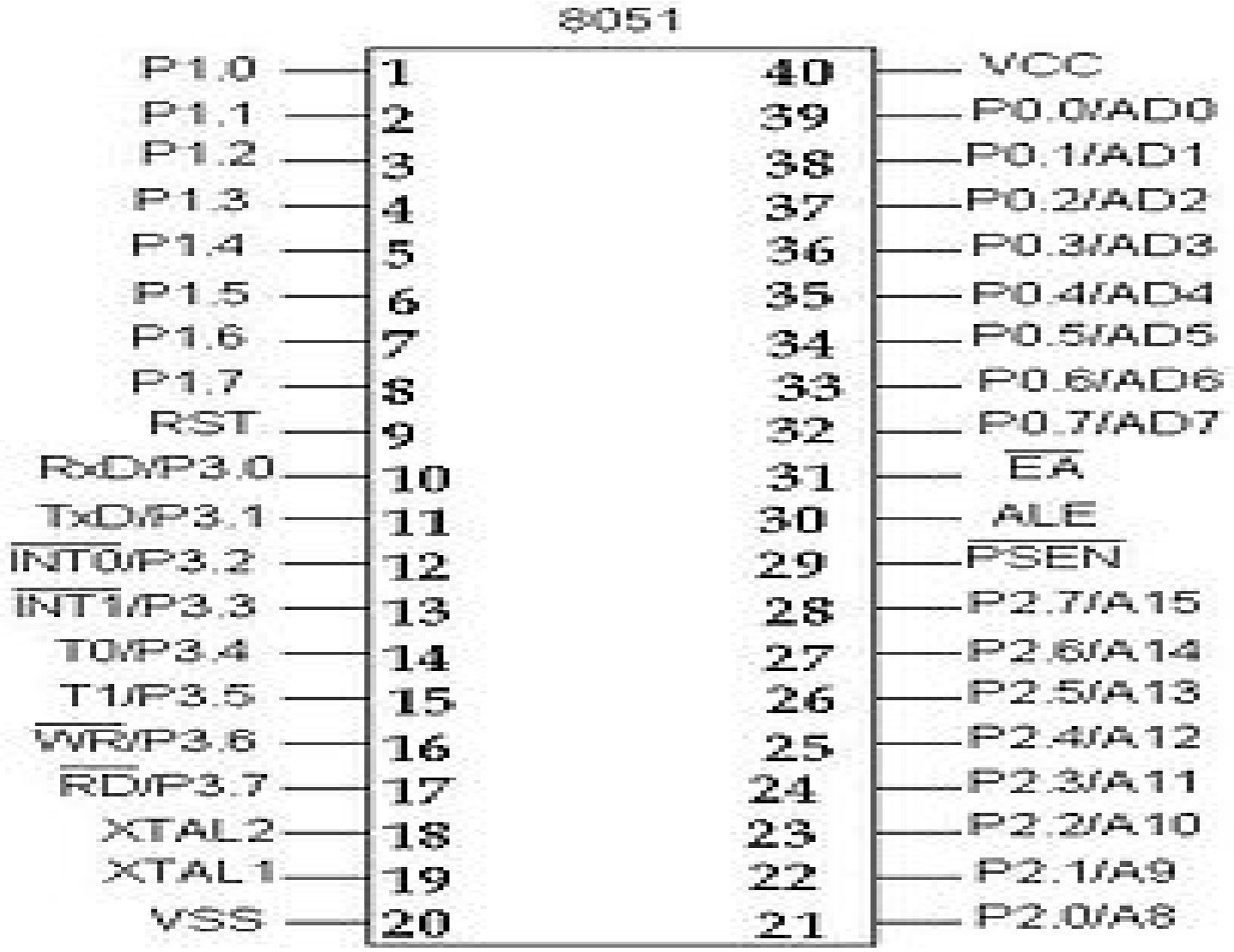
- 8051 Microcontroller Architecture
- Pin Diagram
- Input/Output Ports
- Registers
- Counter and Timers
- Serial Ports
- Interrupts
- Addressing modes
- Instruction Set and Programming:
  - Data Transfer
  - Arithmetic
  - Logical and Decision Making Operations

No.	Microprocessor	Microcontroller
1.	Microprocessor contains ALU, control unit (clock and timing circuit), different register and interrupt circuit.	Microcontroller contains microprocesso memory (ROM and RAM), I/O interfacin circuit and peripheral devices such as A/ 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 dat 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/ devices.
5.	Microprocessor based system requires more hardware.	Microcontroller based system requires les hardware reducing PCB size and increasin 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 an code.
8.	Less number of pins are multifunctioned.	More number pins are multifunctioned.

The features of the 8051 family are as follows :

1) 4096 bytes on - chip program memory.
2) 128 bytes on - chip data memory.
3) Four register banks.
4) 128 User-defined software flags.
5) 64 Kilobytes each program and external RAM addressability.
6) One microsecond instruction cycle with 12 MHz crystal.
7) 32 bidirectional I/O lines organized as four 8-bit ports (16 lines on 8031
8) Multiple mode, high-speed programmable serial port.
9) Two multiple mode, 16-bit Timers/Counters.
10) Two-level prioritized interrupt structure.
11) Full depth stack for subroutine return linkage and data storage.
12) Direct Byte and Bit addressability.
13) Binary or Decimal arithmetic.
14) Signed-overflow detection and parity computation.
15) Hardware Multiple and Divide in 4 μsec.
16) Integrated Boolean Processor for control applications.
17) Upwardly compatible with existing 8084 software.

PIN DIAGRAM


8051
P1.0 1
P1.1 2
P1.2 3
P1.3 4
P1.4 5
P1.5 6
P1.6 7
P1.7 8
RST 9
RxD/P3.0 10
TxD/P3.1 11
INT0/P3.2 12
INT1/P3.3 13
T0/P3.4 14
T1/P3.5 15
WR/P3.6 16
RD/P3.7 17
XTAL2 18
XTAL1 19
VSS 20
40 VCC
39 P0.0/AD0
38 P0.1/AD1
37 P0.2/AD2
36 P0.3/AD3
35 P0.4/AD4
34 P0.5/AD5
33 P0.6/AD6
32 P0.7/AD7
31 EA
30 ALE
29 PSEN
28 P2.7/A15
27 P2.6/A14
26 P2.5/A13
25 P2.4/A12
24 P2.3/A11
23 P2.2/A10
22 P2.1/A9
21 P2.0/A8

PIN DIAGRAM

ESET The reset input pin resets the 8051, only when it goes high for two or more machine cycles. For roper reinitialization after reset, the clock must be running.

ALE/PROG The address latch enable output pulse indicates that the valid address bits are available o eir respective pins. This ALE signal is valid only for external memory accesses. Normally, the ALE pulse e emitted at a rate of one-sixth of the oscillator frequency. This pin acts as program pulse input during hip EPROM programming. ALE may be used for external timing or clocking purpose. One ALE puls kipped during each access to external data memory.

$\overline{EA}$ /V_{pp} External access enable pin, if tied low, indicates that the 8051 can address external program nemory. In other words, the 8051 can execute a program in external memory, only if $\overline{EA}$ is tied low. For xecution of programs in internal memory, the $\overline{EA}$ must be tied high. This pin also receives 21 volts for rogramming of the on-chip EPROM.

$\overline{PSEN}$ Program store enable is an active-low output signal that acts as a strobe to read the external pro- gram memory. This goes low during external program memory accesses.

PIN DIAGRAM

LE (Address Latch Enable, Pin 30)

AD_0 to AD_7 lines are multiplexed. To demultiplex these lines and for obtaining low
alf of an address, an external latch and ALE signal of 8051 is used.

ST (Reset, Pin 9)

This pin is used to reset 8051. For proper reset operation, reset signal must be hei
gh at least for two machine cycles, while oscillator is running.

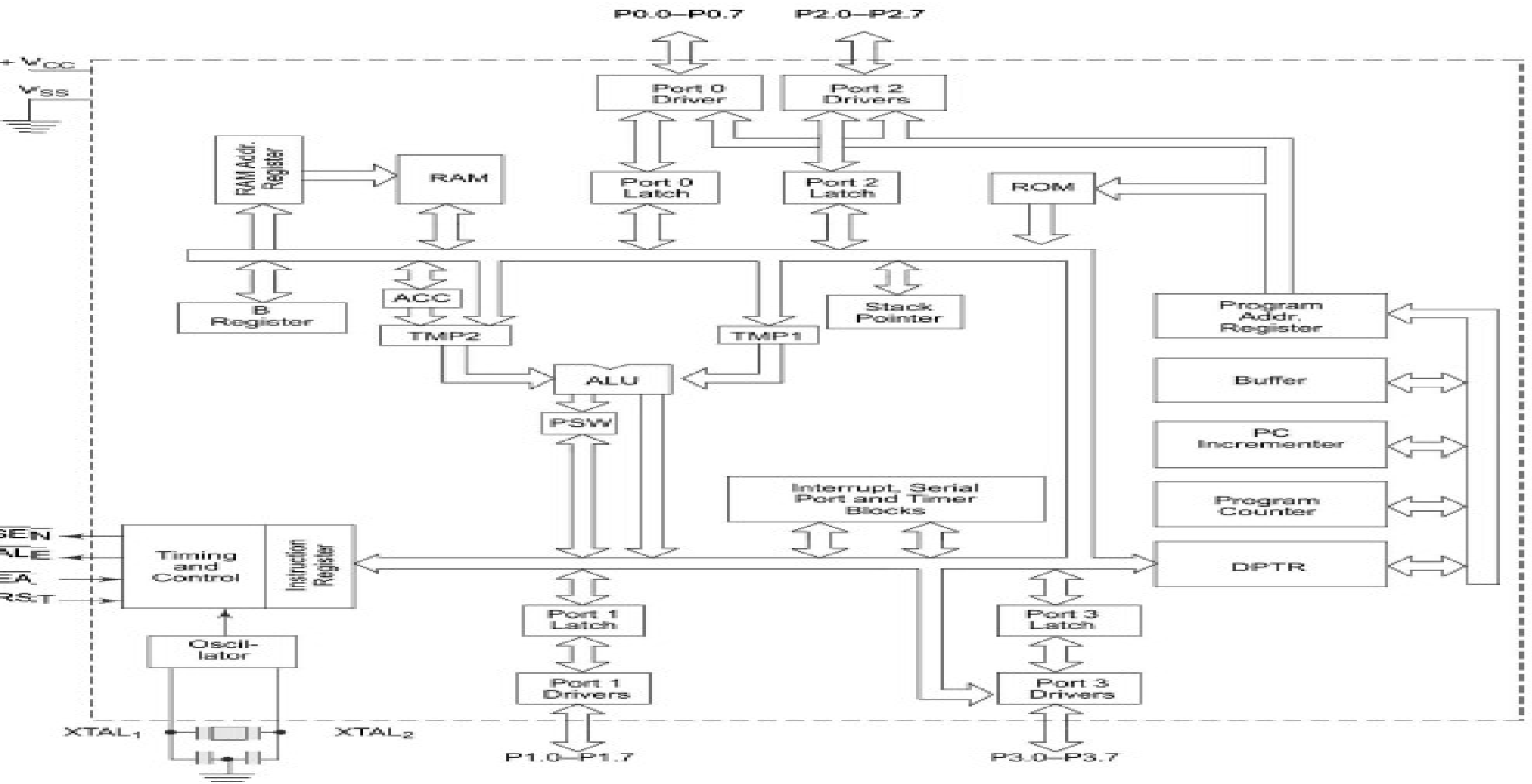
$\overline{\text{SEN}}$ (Program Store Enable, Pin 29)

It is the active low output control signal used to activate the enable signal of th
ternal ROM/EPROM. It is activated every six oscillator periods while reading th
ternal memory. Thus, this signal acts as the read strobe to external program memory.

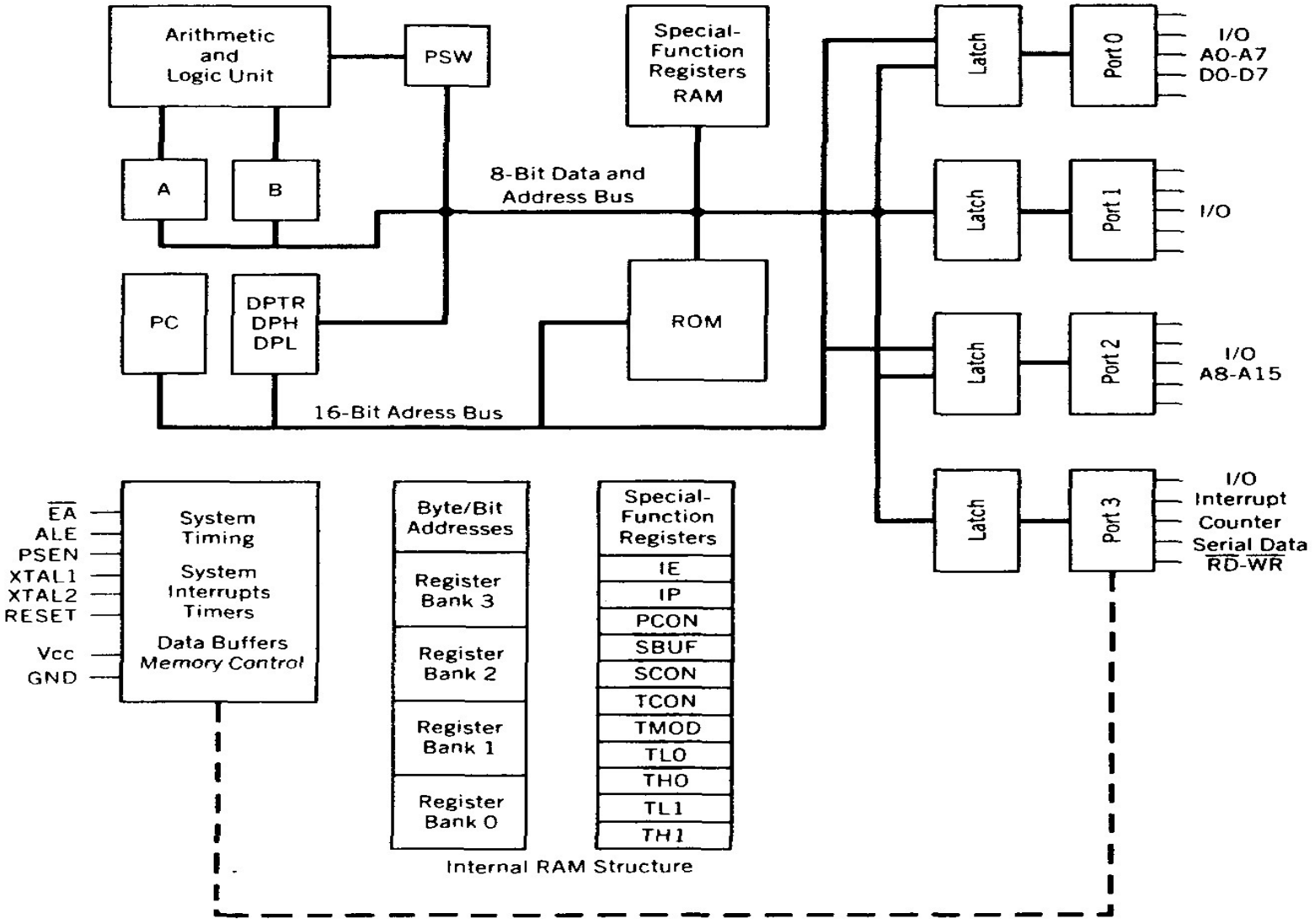
$\overline{\text{A}}$ (External Access, Pin 31)

When the $\overline{\text{EA}}$ pin is high (connected to V_{CC}), program fetches to addresses 0000
rough 0FFFH are directed to the internal ROM and program fetches to addresses 1000
rough FFFFH are directed to external ROM/EPROM. When $\overline{\text{EA}}$ is low (gorunded), a
ldresses (0000H to FFFFH) fetched by program are directed to the extern
OM/EPROM.

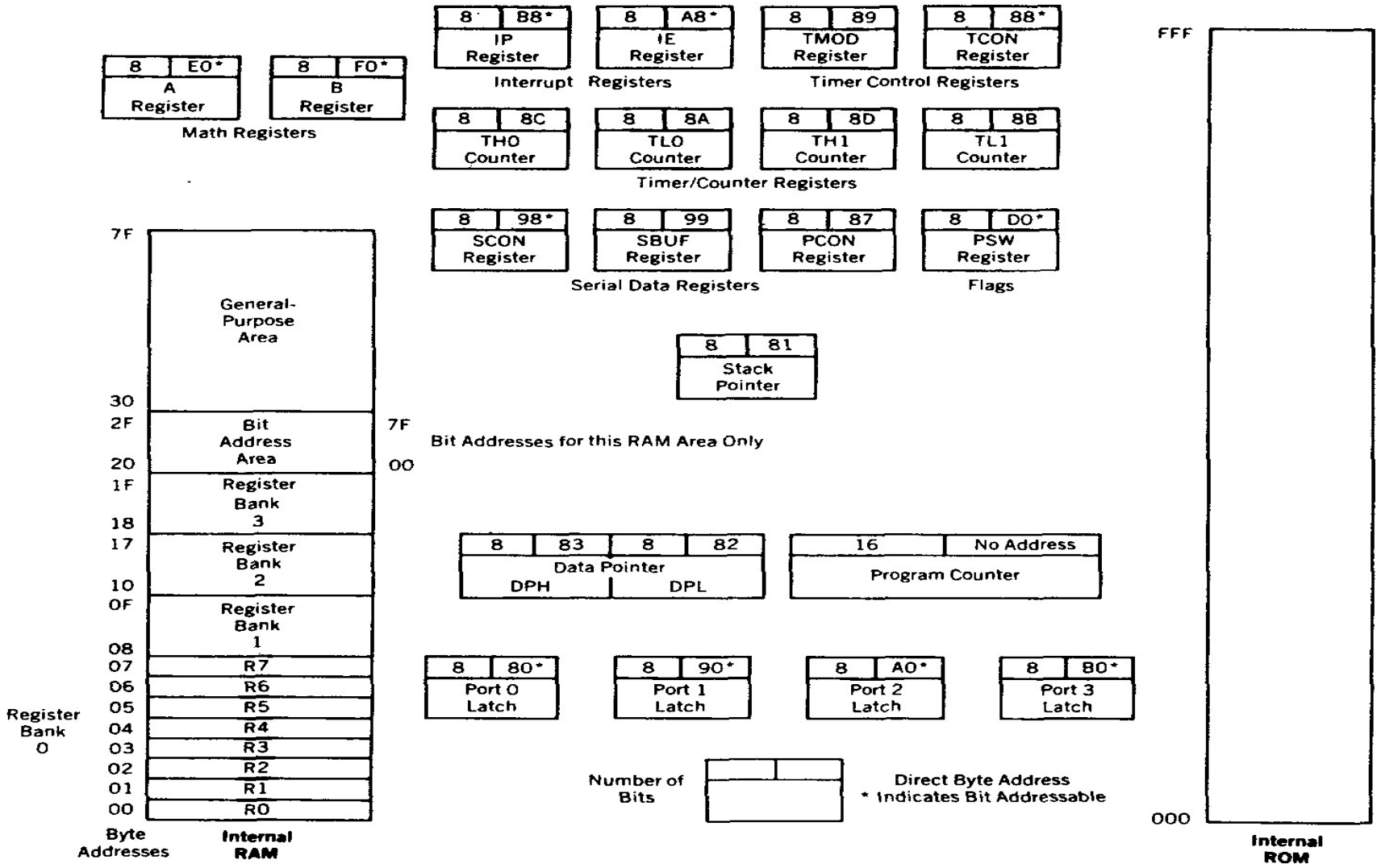
8051 Architecture



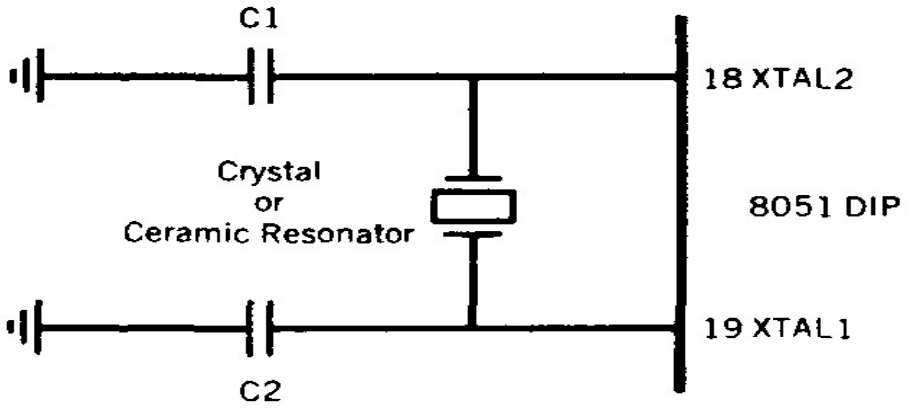
8051 Block Diagram



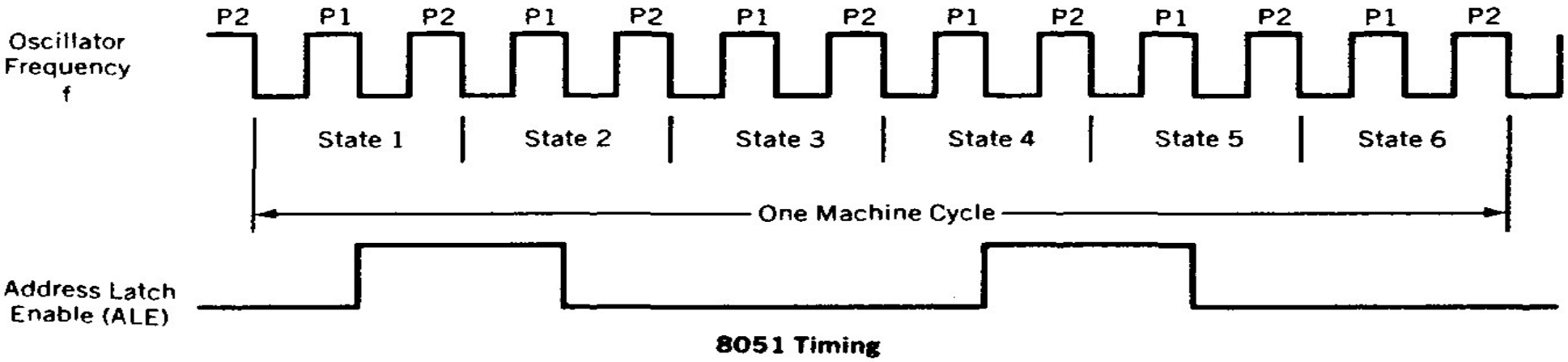
8051 Programming Model



Oscillator Circuit and Timing

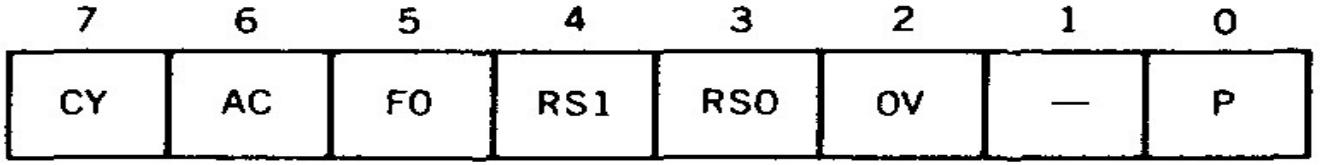


Crystal or Ceramic Resonator Oscillator Circuit



8051 Timing

Program Status Word (PSW)



THE PROGRAM STATUS WORD (PSW) SPECIAL FUNCTION REGISTER

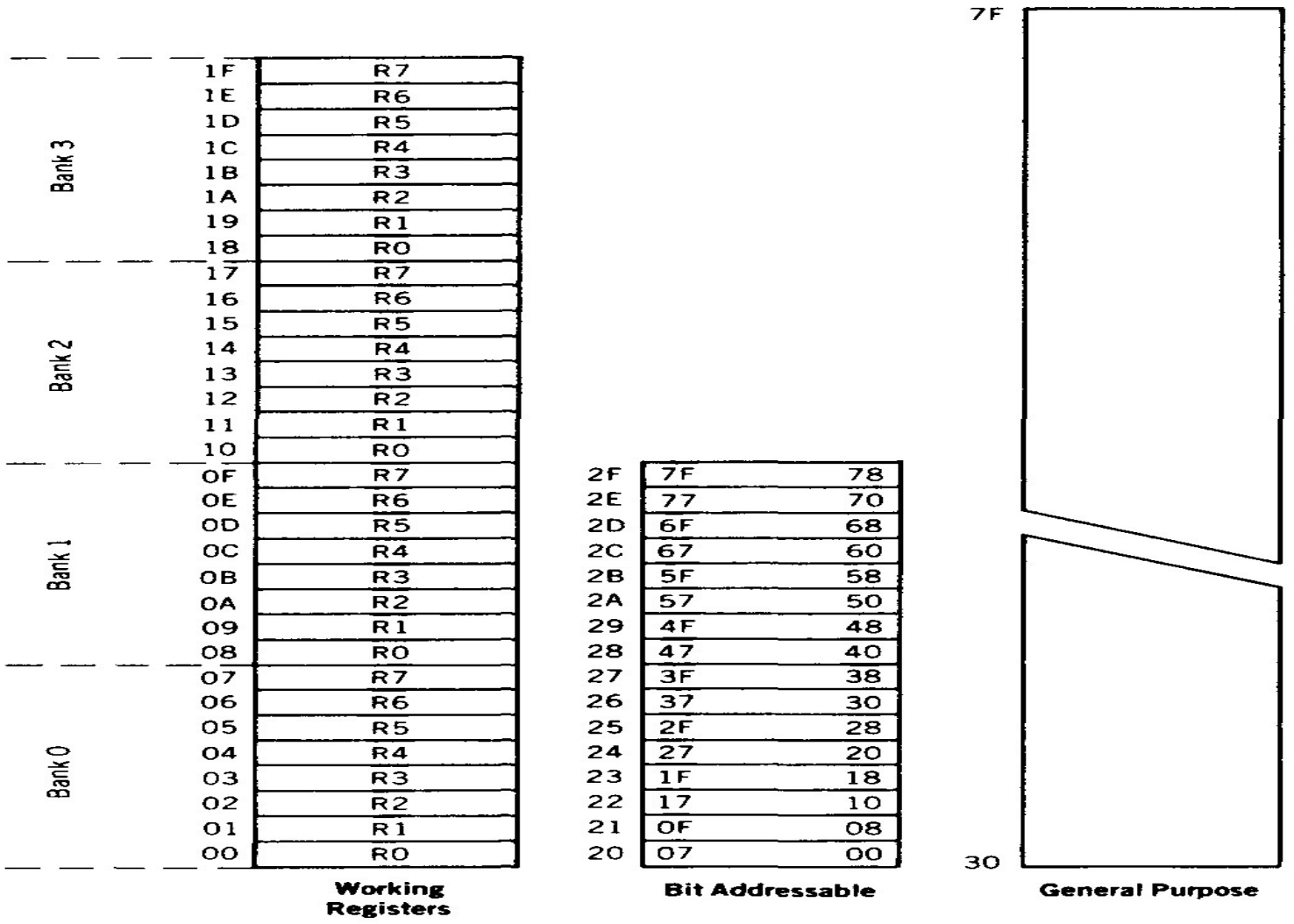
Bit	Symbol	Function
7	CY	Carry flag; used in arithmetic, JUMP, ROTATE, and BOOLEAN instructions
6	AC	Auxilliary carry flag; used for BCD arithmetic
5	F0	User flag 0
4	RS1	Register bank select bit 1
3	RS0	Register bank select bit 0

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

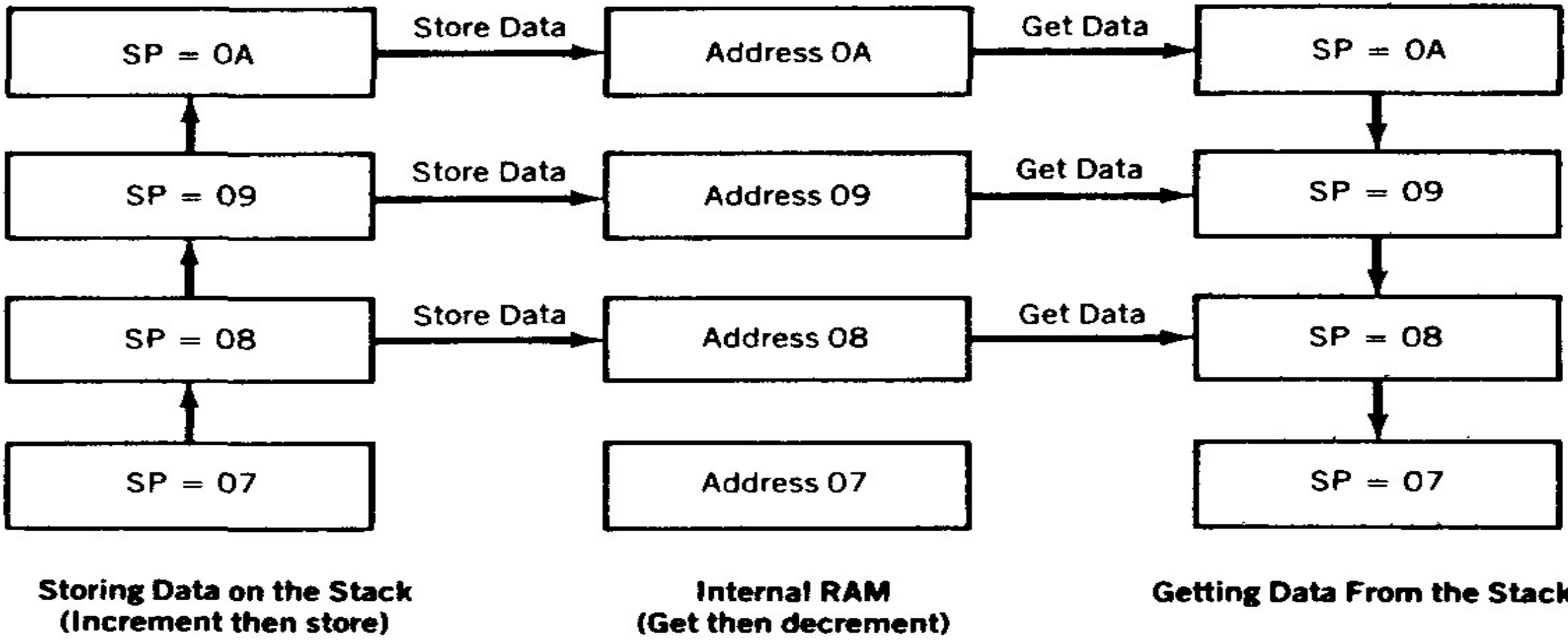
2	OV	Overflow flag; used in arithmetic instructions
1	—	Reserved for future use
0	P	Parity flag; shows parity of register A: 1 = Odd Parity

Bit addressable as PSW.0 to PSW.7

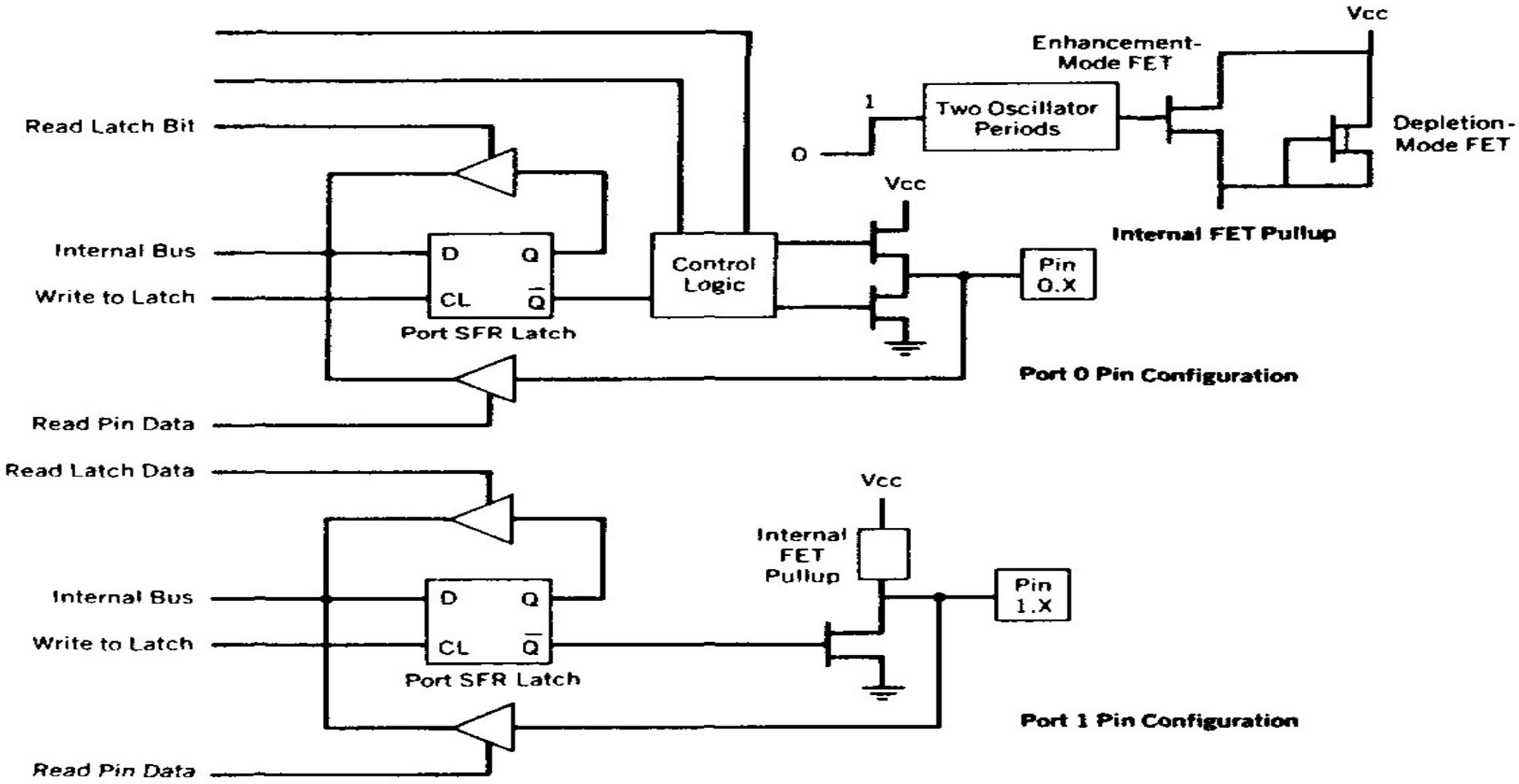
Internal RAM Organization



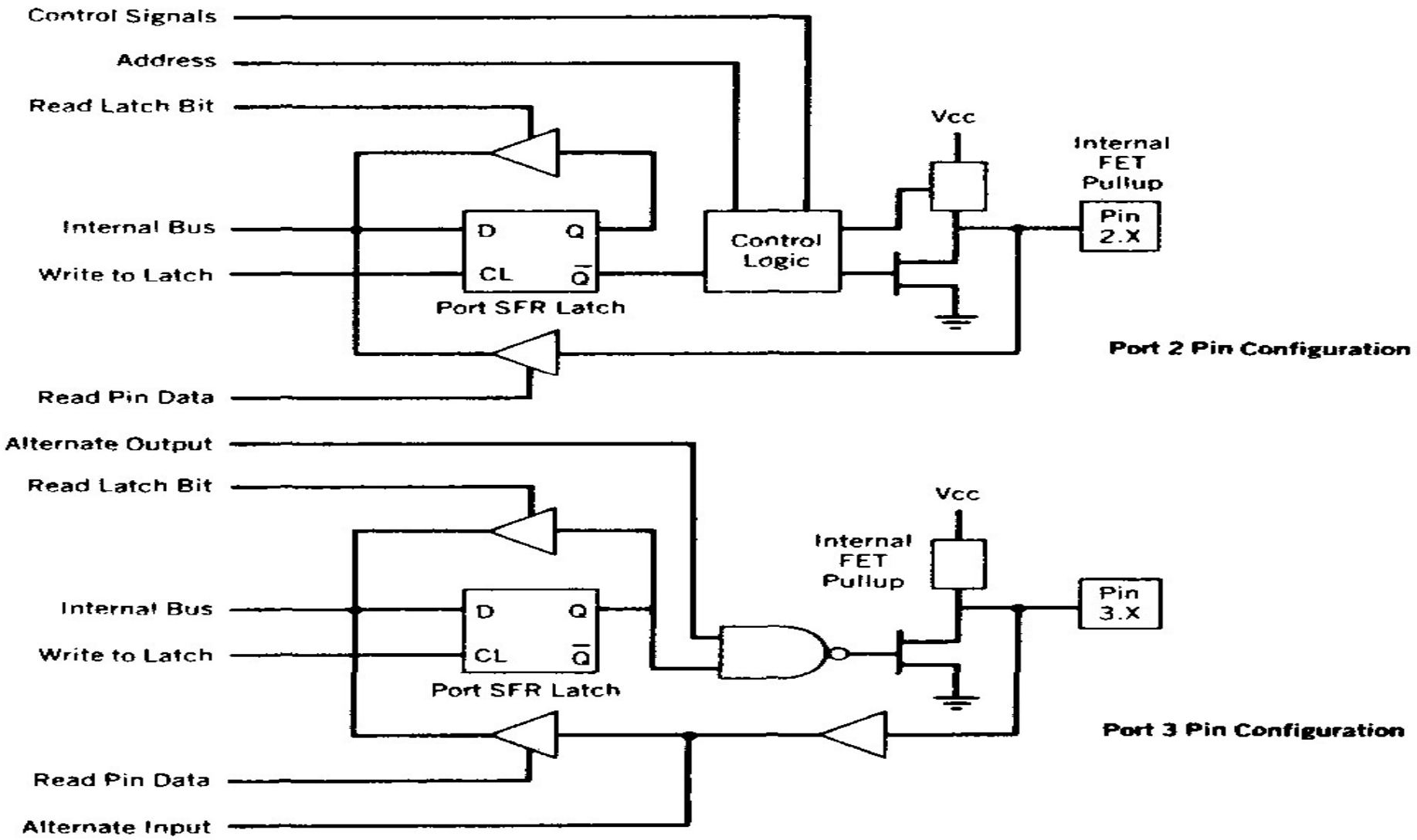
Stack Operation



Port Pin Circuits



Port Pin Circuits



ort 3 (P3.0–P3.7) Port 3 is an 8-bit bidirectional bit addressable I/O port which has been allotted an addre
he SFR address range of 8051. The port 3 pins also serve the alternative functions as listed

Alternate Functions of Pins of Port 3 (Intel Corp.)

Port 3 Pin	*Alternative Function*
P3.0	Acts as serial input data pin (RXD)
P3.1	Acts as serial output data pin (TXD)
P3.2	Acts as external interrupt pin 0 ($\overline{INT_0}$)
P3.3	Acts as external interrupt input pin 1 ($\overline{INT_1}$)
P3.4	Acts as external input to timer 0 (T0)
P3.5	Acts as external input to timer 1(T1)
P3.6	Acts as write control signal for external data memory ($\overline{WR}$)
P3.7	Acts as read control signal for external data memory read operation ($\overline{RD}$)

Addressing Modes

- Direct Addressing Mode
- Indirect Addressing Mode
- Register Instructions
- Register Specific (Register Implicit)
- Immediate Mode
- Indexed Addressing Mode

rect Addressing In this mode of addressing, the operands are specified using the 8-bit address field, ıe instruction format. Only internal data RAM and SFRS can be directly addressed.

ımple `MOV R0, 89H.`

re 89H is address of a special function register TMOD.

lirect Addressing In this mode of addressing, the 8-bit address of an operand is stored in a register the register, instead of the 8-bit address, is specified in the instruction. The registers R_0 and R_1 of the cted bank of registers or stack pointer can be used as address registers for storing the 8-bit addresses .

he address register for 16-bit addresses can only be 'data pointer' (DPTR).

xample `ADD A, @ R0`

gister Instructions In this addressing mode, operands are stored in the registers R_0–R_7 of the selected ter bank. One of these eight registers (R_0–R_7) is specified in the instruction using the 3-bit register specifica- ield of the opcode format. A register bank can be selected using the two bank select bits of the PSW.

ımple `ADD A, R7.`

Register Specific Instructions In this type of instructions, the operand is implicitly specified usi
ne of the registers. Some of the instructions always operate only on a specific register. These type of instru
ons fall under this category.

xample RLA; This instruction rotates accumulator left.

mediate Mode In this mode, an immediate data, i.e. a constant is specified in the instruction, a
opcode byte.

mple ADD A, #100

mmediate data 100 (decimal) is added to the contents of accumulator. For specifying a hex num
in this type of instruction, it should be followed by H.

Indexed Addressing Only program memory can be accessed using this addressing mode. Basica
this mode of addressing is accomplished in 8051 for look-up table manipulations. Program counter or d
pointer are the allowed 16-bit address storage registers, in this mode of addressing. These 16-bit registers po
to the base of the look-up table and the ACC register contains a code to be converted using the look-up tab
In other words, it contains the relative address of the code in the look-up table. The look-up table data addr
is found out by adding the contents of register ACC with that of the program counter or Data Pointer.
In case of jump instruction, the contents of accumulator are added with one of the specified 16-bit regist
to form the jump destination address.

Example MOVC A, @ A+DPTR

JMP @ A+DPTR

8051 Instruction Set

8051 Instructions can be categorized in the following categories:

- 1. Data Transfer Instructions
- 2. Arithmetic Instructions
- 3.Logical Instructions
- 4. Boolean Instructions
- 5. Control Transfer Instructions

Data Transfer Instructions

These instructions implement a bit, byte or 16-bit data transfer operations between the 'SRC' (sour
and DST 'destination' operands.
Both operands can be internal direct data memory operands.
Both cannot be direct and/or indirect register operands R0 to R7.

Immediate operand can be only a source and not a destination.
Program counter is not accessible.
Restricted bit-transfer operations are allowed.
'Implicit' indicates not to be specified in the instruction or assumed internally by default, otherwise
operand is to be specified in the instruction. If an operand is not marked implicit, it is explicit.
Only R0 and R1 are used for indirect addressing mode.

Data Transfer Instructions

Finally, the following five types of opcodes are used to move data:

1. **MOV**
2. **MOVX**
3. **MOVC**
4. **PUSH and POP**
5. **XCH**

MOV A,#n	Copy the immediate data byte n to the A register
MOV A,Rr	Copy data from register Rr to register A
MOV Rr,A	Copy data from register A to register Rr
MOV Rr,#n	Copy the immediate data byte n to register Rr
MOV DPTR,#nn	Copy the immediate 16-bit number nn to the DPTR register

Data Transfer Instructions

MOV A,add	Copy data from direct address add to register A
MOV add,A	Copy data from register A to direct address add
MOV Rr,add	Copy data from direct address add to register Rr
MOV add,Rr	Copy data from register Rr to direct address add
MOV add,#n	Copy immediate data byte n to direct address add
MOV add1,add2	Copy data from direct address add2 to direct address add1
MOV @Rp,#n	Copy the immediate byte n to the address in Rp
MOV @Rp,add	Copy the contents of add to the address in Rp
MOV @Rp,A	Copy the data in A to the address in Rp
MOV add,@Rp	Copy the contents of the address in Rp to add
MOV A,@Rp	Copy the contents of the address in Rp to A
MOVX A,@Rp	Copy the contents of the external address in Rp to A
MOVX A,@DPTR	Copy the contents of the external address in DPTR to A
MOVX @Rp,A	Copy data from A to the external address in Rp
MOVX @DPTR,A	Copy data from A to the external address in DPTR

Data Transfer Instructions

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
PUSH add	Increment SP; copy the data in add to the internal RAM address contained in SP
POP add	Copy the data from the internal RAM address contained in SP to add; decrement the SP
XCH A,Rr	Exchange data bytes between register Rr and A
XCH A,add	Exchange data bytes between add and A
XCH A,@Rp	Exchange data bytes between A and address in Rp
XCHD A,@Rp	Exchange lower nibble between A and address in Rp

Arithmetic Instructions

- These instructions implement arithmetic and logical operations along with increment, decremen and decimal adjust operations.
- Accumulator is a compulsory destination operand for two-operand instructions.
- Immediate operand can be only source and not a destination operand.
- Program counter or its part cannot be an operand.
- Immediate data cannot be an operand for INC/DEC instructions.
- There are no SUB or Compare instructions. Programmer has to implement the subtraction withou borrow and comparison instructions using SUBB.

Arithmetic Instructions

INC destination	Increment destination by 1
DEC destination	Decrement destination by 1
ADD/ADDC destination,source	Add source to destination without/with carry (C) flag
SUBB destination,source	Subtract, with carry, source from destination
MUL AB	Multiply the contents of registers A and B
DIV AB	Divide the contents of register A by the contents of register B
DA A	Decimal Adjust the A register

INC A	Add a one to the A register
INC Rr	Add a one to register Rr
INC add	Add a one to the direct address
INC @ Rp	Add a one to the contents of the address in Rp
INC DPTR	Add a one to the 16-bit DPTR
DEC A	Subtract a one from register A
DEC Rr	Subtract a one from register Rr
DEC add	Subtract a one from the contents of the direct address
DEC @ Rp	Subtract a one from the contents of the address in register Rp

Arithmetic Instructions

ADD A,#n	Add A and the immediate number n; put the sum in A
ADD A,Rr	Add A and register Rr; put the sum in A
ADD A,add	Add A and the address contents; put the sum in A
ADD A,@Rp	Add A and the contents of the address in Rp; put the sum in A
ADDC A,#n	Add the contents of A, the immediate number n, and the C flag; put the sum in A
ADDC A,add	Add the contents of A, the direct address contents, and the C flag; put the sum in A
ADDC A,Rr	Add the contents of A, register Rr, and the C flag; put the sum in A
ADDC A,@Rp	Add the contents of A, the contents of the indirect address in Rp, and the C flag; put the sum in A

Arithmetic Instructions

SUBB A,#n	Subtract immediate number n and the C flag from A; put the result in A
SUBB A,add	Subtract the contents of add and the C flag from A; put the result in A
SUBB A,Rr	Subtract Rr and the C flag from A; put the result in A
SUBB A,@Rp	Subtract the contents of the address in Rp and the C flag from A; put the result in A
MUL AB	Multiply A by B; put the low-order byte of the product in A, put the high-order byte in B
DIV AB	Divide A by B; put the integer part of quotient in register A and the integer part of the remainder in B
DA A	Adjust the sum of two packed BCD numbers found in A register; leave the adjusted number in A.

Logical Instructions

These instructions implement basic logical operations along with rotate and clear operations.

A is not a compulsory destination operand for logical instructions excluding complement instructi

CPL. However, one of the operands must be A.

Any other DST operand can be a destination operand for logical instructions. Immediate operand c

only be a source operand.

Program counter or its part can't be an operand.

Immediate data can't be an operand for INC/DEC or any other single operand instructio

instructions.

There are no SUB or Compare instructions. The programmer has to implement the subtraction witho

borrow and comparison instructions using SUBB.

Logical Instructions

AND	ANL (AND logical)
OR	ORL (OR logical)
XOR	XRL (exclusive OR logical)
NOT	CPL (complement)

RL	Rotate a byte to the left; the Most Significant Bit (MSB) becomes the Least Significant Bit (LSB)
RLC	Rotate a byte and the carry bit left; the carry becomes the LSB, the MSB becomes the carry
RR	Rotate a byte to the right; the LSB becomes the MSB
RRC	Rotate a byte and the carry to the right; the LSB becomes the carry, and the carry the MSB
SWAP	Exchange the low and high nibbles in a byte

Logical Instructions (Byte Level)

ANL A,#n	AND each bit of A with the same bit of immediate number n; put the results in A
ANL A,add	AND each bit of A with the same bit of the direct RAM address; put the results in A
ANL A,Rr	AND each bit of A with the same bit of register Rr; put the results in A
ANL A,@Rp	AND each bit of A with the same bit of the contents of the RAM address contained in Rp; put the results in A
ANL add,A	AND each bit of A with the direct RAM address; put the results in the direct RAM address
ANL add,#n	AND each bit of the RAM address with the same bit in the number n; put the result in the RAM address
CLR A	Clear each bit of the A register to zero
CPL A	Complement each bit of A; every 1 becomes a 0, and each 0 becomes a 1

Logical Instructions (Bit Level)

ANL C,b	AND C and the addressed bit; put the result in C
ANL C,/b	AND C and the complement of the addressed bit; put the result in C; the addressed bit is not altered
ORL C,b	OR C and the addressed bit; put the result in C
ORL C,/b	OR C and the complement of the addressed bit; put the result in C; the addressed bit is not altered
CPL C	Complement the C flag
CPL b	Complement the addressed bit
CLR C	Clear the C flag to zero
CLR b	Clear the addressed bit to zero
MOV C,b	Copy the addressed bit to the C flag
MOV b,C	Copy the C flag to the addressed bit
SETB C	Set the flag to one
SETB b	Set the addressed bit to one

Logical Instructions

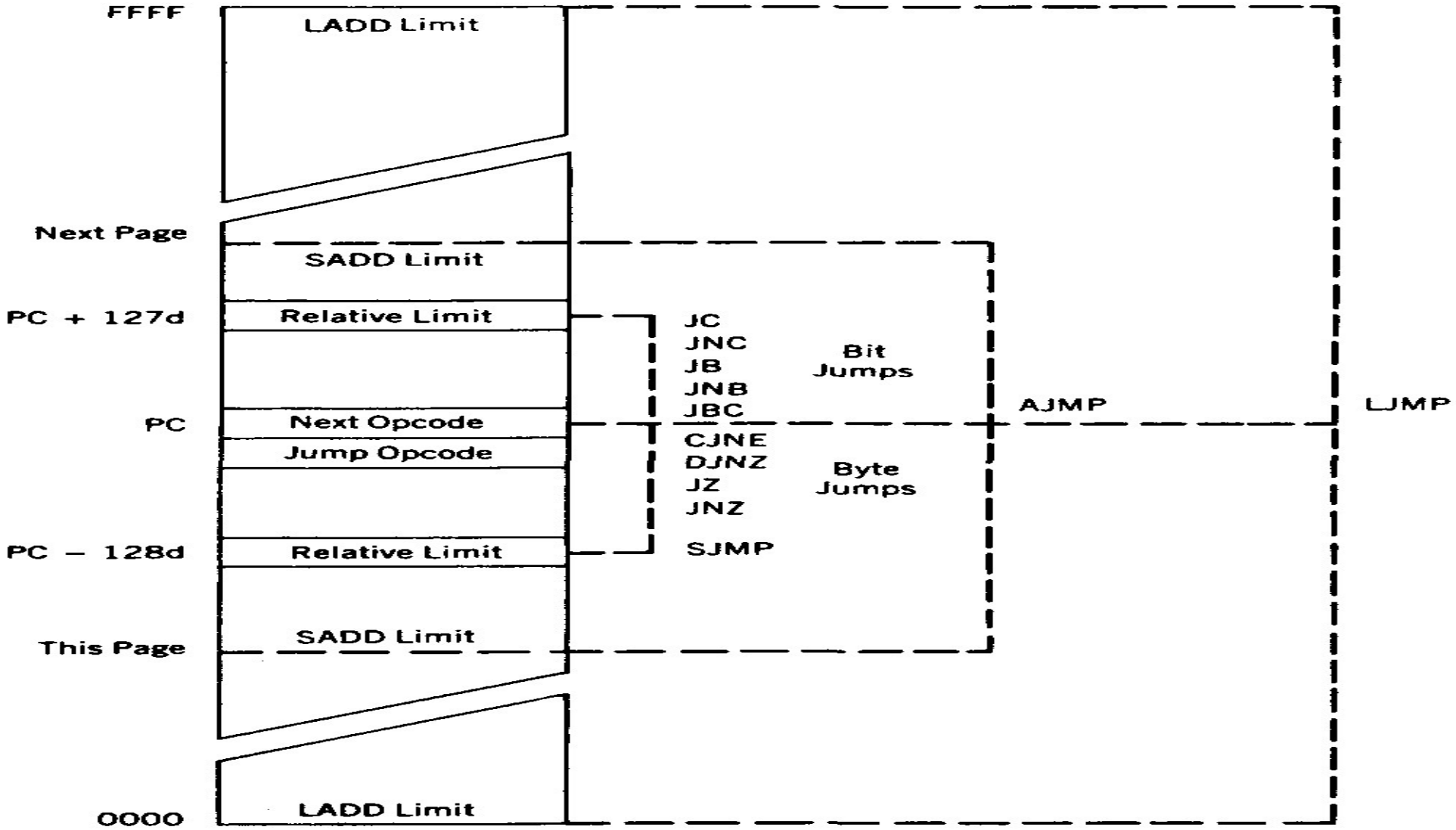
RL A	Rotate the A register one bit position to the left; bit A0 to bit A1, A1 to A2, A2 to A3, A3 to A4, A4 to A5, A5 to A6, A6 to A7, and A7 to A0
RLC A	Rotate the A register and the carry flag, as a ninth bit, one bit position to the left; bit A0 to bit A1, A1 to A2, A2 to A3, A3 to A4, A4 to A5, A5 to A6, A6 to A7, A7 to the carry flag, and the carry flag to A0
RR A	Rotate the A register one bit position to the right; bit A0 to bit A7, A6 to A5, A5 to A4, A4 to A3, A3 to A2, A2 to A1, and A1 to A0
RRC A	Rotate the A register and the carry flag, as a ninth bit, one bit position to the right; bit A0 to the carry flag, carry flag to A7, A7 to A6, A6 to A5, A5 to A4, A4 to A3, A3 to A2, A2 to A1, and A1 to A0
SWAP A	Interchange the nibbles of register A; put the high nibble in the low nibble position and the low nibble in the high nibble position

Control Transfer Instructions

he control Transfer instructions transfer the control of execution or change the sequence of execution eith
onditionally or unconditionally. All the jump, call and return instructions come under this group of contr
ansfer instructions. Control transfer instructions of 8051 are significantly different than the respective m
oprocessor instructions. The control transfer instructions of 8051 as usual can be divided in to two categ
es; 1) Conditional control transfer and 2) Unconditional control transfer.

The control transfer instructions of 8051 can have up to three operands.
The unconditional jumps only have one operand, i.e. the jump or call address.
Unconditional jumps with direct as well as indirect addressing are available.
Conditional jumps are always short while unconditional jumps can be short, long or absolu
There are separate instructions for return from a subroutine and interrupt service routine.

Jump Instruction Range



Bit Jumps

JC radd	Jump relative if the carry flag is set to 1
JNC radd	Jump relative if the carry flag is reset to 0
JB b,radd	Jump relative if addressable bit is set to 1
JNB b,radd	Jump relative if addressable bit is reset to 0
JBC b,radd	Jump relative if addressable bit is set, and clear the addressable bit to 0

Byte Jumps

CJNE A,add,radd	Compare the contents of the A register with the contents of the direct address; if they are *not* equal, then jump to the relative address; set the carry flag to 1 if A is less than the contents of the direct address; otherwise, set the carry flag to 0
CJNE A,#n,radd	Compare the contents of the A register with the immediate number n; if they are *not* equal, then jump to the relative address; set the carry flag to 1 if A is less than the number; otherwise, set the carry flag to 0
CJNE Rn,#n,radd	Compare the contents of register Rn with the immediate number n; if they are *not* equal, then jump to the relative address; set the carry flag to 1 if Rn is less than the number; otherwise, set the carry flag to 0
CJNE @Rp,#n,radd	Compare the contents of the address contained in register Rp to the number n; if they are *not* equal, then jump to the relative address; set the carry flag to 1 if the contents of the address in Rp are less than the number; otherwise, set the carry flag to 0

Byte Jumps

DJNZ Rn,radd	Decrement register Rn by 1 and jump to the relative address if the result is *not* zero; no flags are affected
DJNZ add,radd	Decrement the direct address by 1 and jump to the relative address if the result is *not* 0; no flags are affected unless the direct address is the PSW
JZ radd	Jump to the relative address if A is 0; the flags and the A register are not changed
JNZ radd	Jump to the relative address if A is *not* 0; the flags and the A register are not changed

Unconditional Jumps

JMP @A+DPTR	Jump to the address formed by adding A to the DPTR; this is an unconditional jump and will always be done; the address can be anywhere in program memory; A, the DPTR, and the flags are unchanged
AJMP sadd	Jump to absolute short range address sadd; this is an unconditional jump and is always taken; no flags are affected
LJMP ladd	Jump to absolute long range address ladd; this is an unconditional jump and is always taken; no flags are affected
SJMP radd	Jump to relative address radd; this is an unconditional jump and is always taken; no flags are affected
NOP	Do nothing and go to the next instruction; NOP (no operation) is used to waste time in a software timing loop; or to leave room in a program for later additions; no flags are affected

Call and Subroutines

ACALL sadd	Call the subroutine located on the same page as the address of the opcode immediately following the ACALL instruction; push the address of the instruction immediately after the call on the stack
LCALL ladd	Call the subroutine located anywhere in program memory space; push the address of the instruction immediately following the call on the stack
RET	Pop two bytes from the stack into the program counter
RETI	Pop two bytes from the stack into the program counter and reset the interrupt enable flip-flops

Write an assembly language program to find whether a given byte is available in the given sequence or not. If it is available, write FF in R3. Otherwise write 00 in R3.

Solution

```
// A program for finding a number in array of numbers stored in data memory
// for example in memory locations {20H,21H,22H,23H,24H}
// the contents are [15,04,06,45,55]
                ORG 0000H
                MOV R0,#20H          // 20H is starting address of array
                MOV R3,#00H          // in R3 register we monitor search
                                     // if R3=00H means number not found
                                     // if R3=FFH means number is found

                MOV R1,#05H          // This is counter, no of elements are 5 in array
AGAIN:
                MOV A,@R0            // A is stored with contents
                                     // of memory location(@r0)
                DEC R1               // Counter is decremented
                INC R0               // Memory address incremented
                CJNE A,#45H,AGAIN    // If number do not match
                                     // with 45H then again
                                     // jump to AGAIN label
                MOV R3,#0FFH         // Store FFH in R3
                                     // indicating 45 is present in array
                END
```

Write an assembly language program to count the number of 1s and 0s in a given 8-bit number.

Solution

```
// Program to compute number of 1s and 0s in 8 bit number
//  logic: initialize R1 and R2 with 00H
//  initialize R3 as a counter
//  clear carry flag (C ) and rotate A along with carry
//  if C=1, increment  R1, else increment R2 and decrement  the counter
//  if counter=0, store the contents of r1 and r2 and end the program
                    ORG 0000H
                    SJMP 30H            // Start execution at 30H
                    ORG 30H
                    MOV A,#05H          // Number is 05H i.e. 00000101
                    MOV R1,#00H         // Counter for 1s
                    MOV R2,#00H         // Counter for 0s
                    MOV R3,#08H         // Counter for total number of bits
                    CLR C
UP:                 RRC A               // Rotate right through carry
                    JNC DOWN            // If carry is not present goto label down
                    INC R1              // Increment R1 counter
                    SJMP EXIT
DOWN:               INC R2
EXIT:               DJNZ R3,UP          // Checking for end of 8 bits, if not, again goto up
                                           label
                    INC R0
                    MOV A,R1            // for saving status of R1
                    MOV A,R2
                    MOV @R0,A
                    END
```

rite an assembly language program to compute x to the power n where both x and n are 8-b
umbers given by user and the result should not be more than 16 bits.

olution

he logic of this program is very simple. The x is multiplied to itself $n-1$ times.

```
        ORG 0000H
        MOV A,#02H // This is x
        MOV B,#03H // This is n
        MOV R0,B
        MOV R1,A
        MOV R2,#01H
OOP1:   MOV A,R2
        MOV B,R1
        MUL AB// Multiplication
        DEC R0// Decrementing counter
        MOV R2,A
        CJNE R0,#00H,LOOP1
        MOV A,R2  // Result is stored in accumulator
        END
```

Counters and Timers

- Two 16-Bit up counters T0 & T1
- Can be used as Timers (Internal) or Counters (External)
- Timers are divided into two 8-bit registers (timer low and timer high)

Counters and Timers

- The count value to be loaded in to the

  Timer = 2^n – Required count.

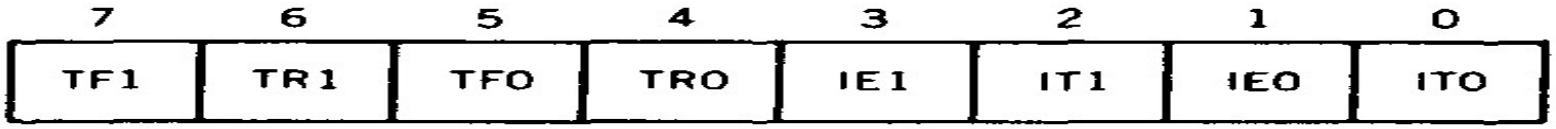
- **Example:**

  To count 10 pulses by using 8-bit Timer is:

  $2^8 – 10 = 256 – 10 = 246$ is the value to be loaded in to the Timer.

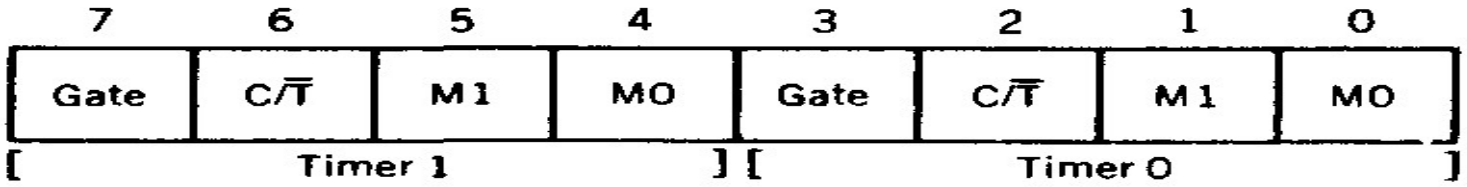
  Timers/Counters are increment in nature, count cannot be decremented.

TCON Register



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	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.
4	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.
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	IEO	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.
0	ITO	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.

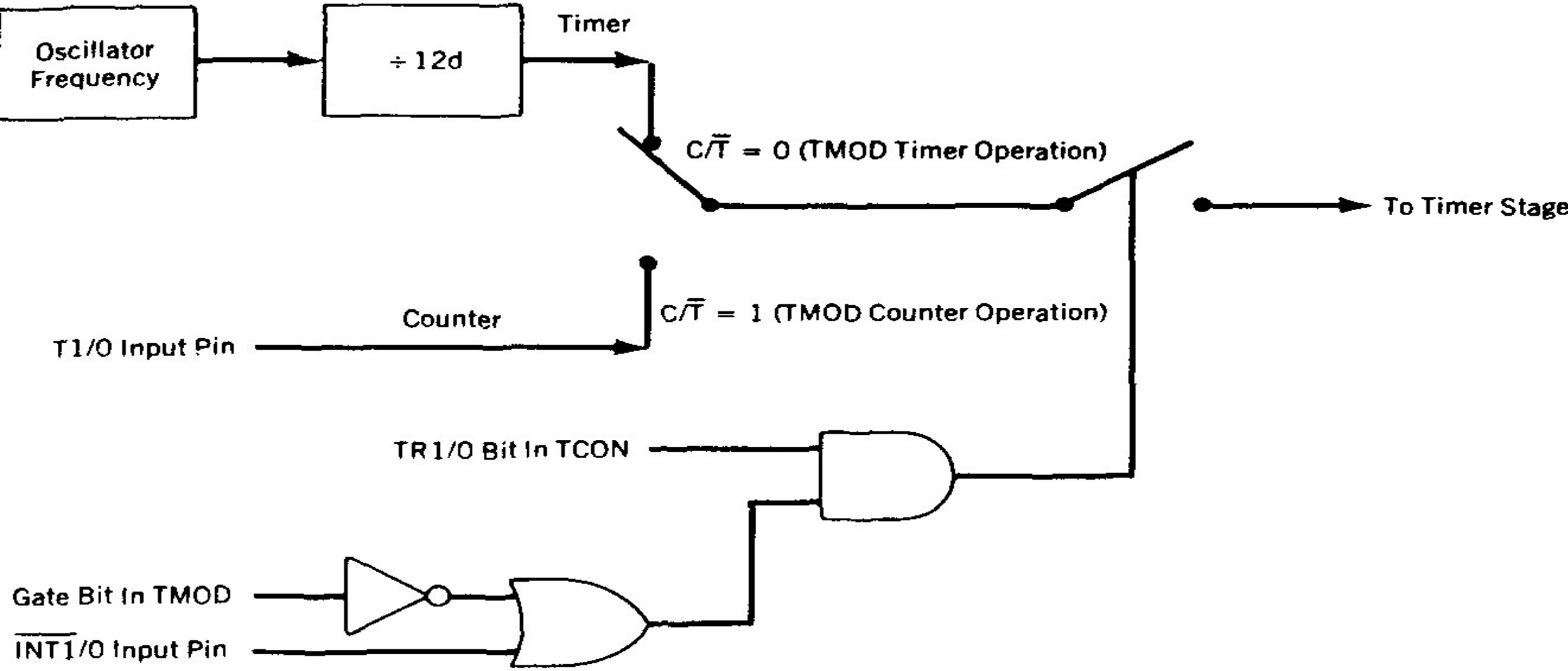
TMOD Register



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

Timer/Counter Control Logic

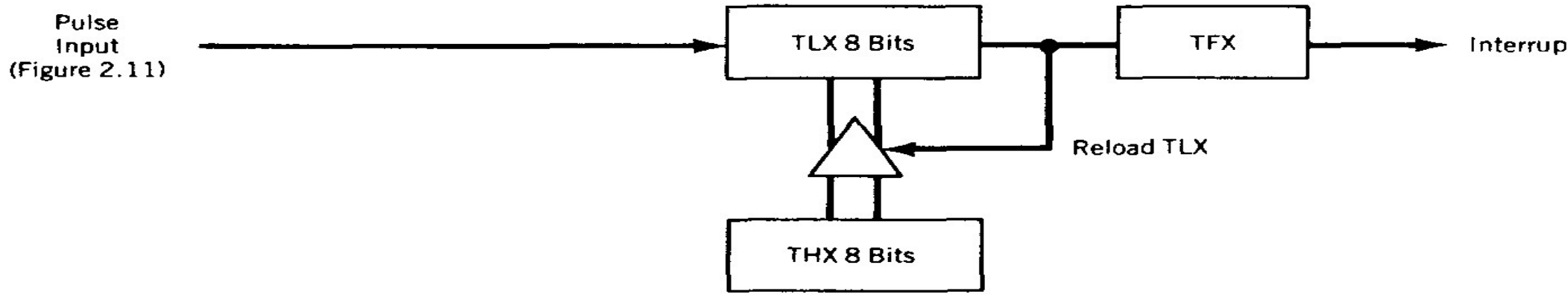


Timer Operation Modes

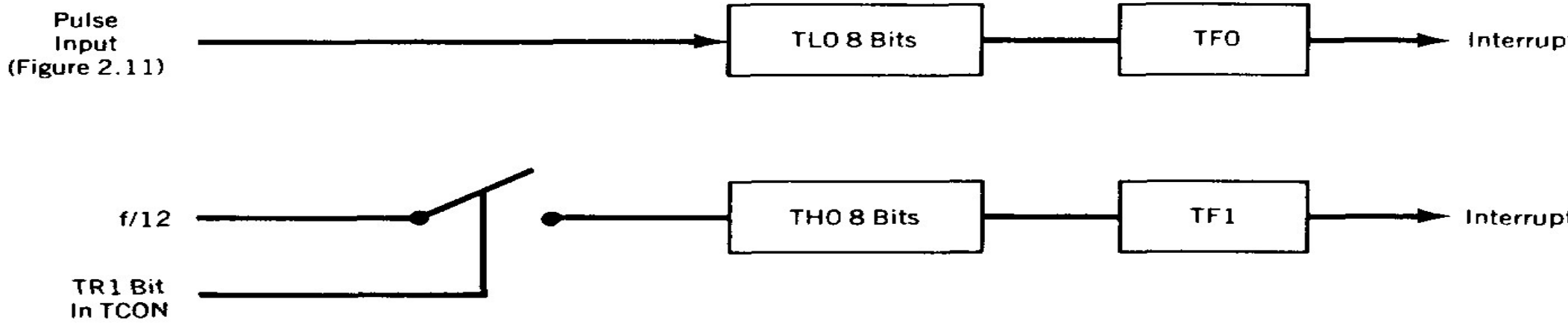
Timer Mode 0 13 - Bit Timer/Counter

Timer Mode 1 16 - Bit Timer/Counter

Timer Operation Modes



Timer Mode 2 Auto - Reload of TL from TH



Timer Mode 3 Two 8 - Bit Timers Using Timer 0

Serial Port Operation

- 4 programmable modes for serial data communication
- Special function registers SBUF, SCON and PCON are used for serial communication
- SBUF is used to hold the data
- SCON controls the data communication
- PCON controls the data rate

SCON 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 the received data is a 1; no interrupt is generated if bit 9 is a 0. If set to 1 for mod no interrupt will be generated unless a valid stop bit is received. Clear to 0 if mode 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 i mode 0.
1	TI	Transmit interrupt flag. Set to one at the end of bit 7 time in mode 0, and at the beginnir 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 throu 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

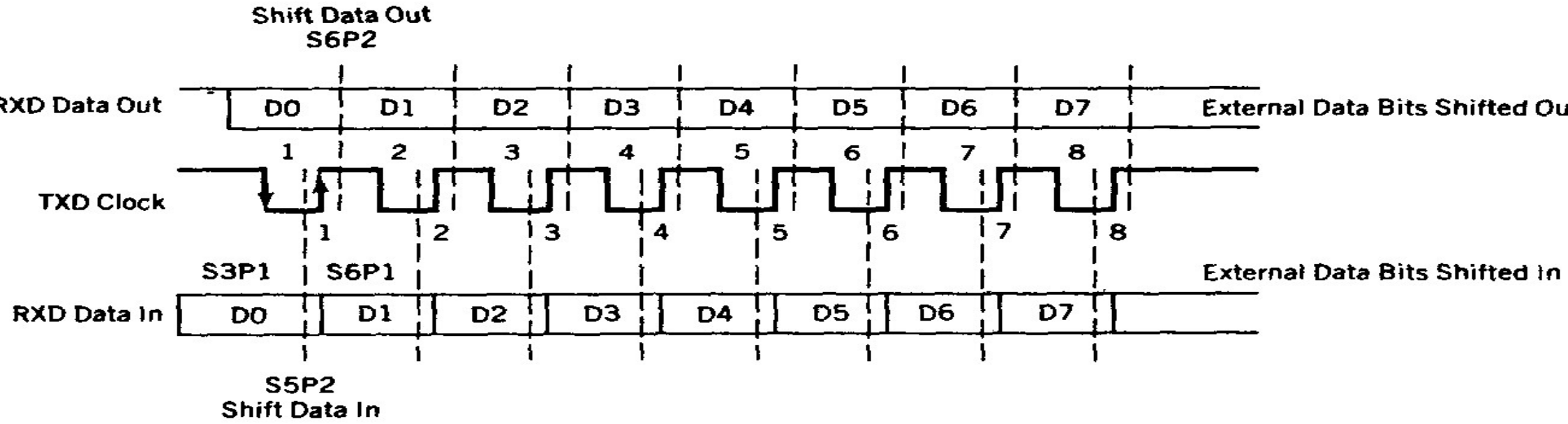
PCON Register

Bit	Symbol	Function
7	SMOD	Serial baud rate modify bit. Set to 1 by program to double baud rate using timer 1 fo 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.

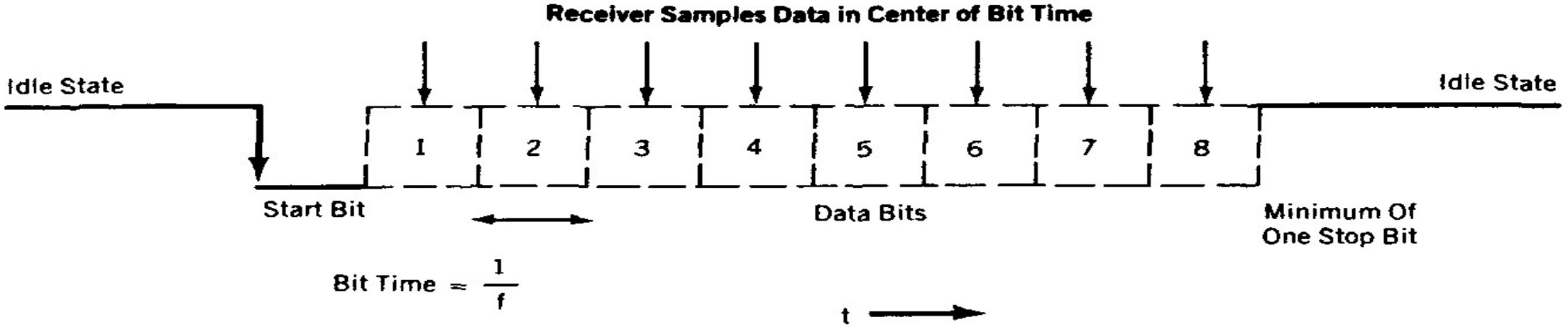
Serial Data Transmission Modes

- Operates in 4 modes
- Mode 0 – Shift Register Mode
- Mode 1 – Standard UART
- Mode 2 – Multiprocessor Mode
- Mode 3 – Similar to Mode 2

Mode 0 – Shift Register Mode



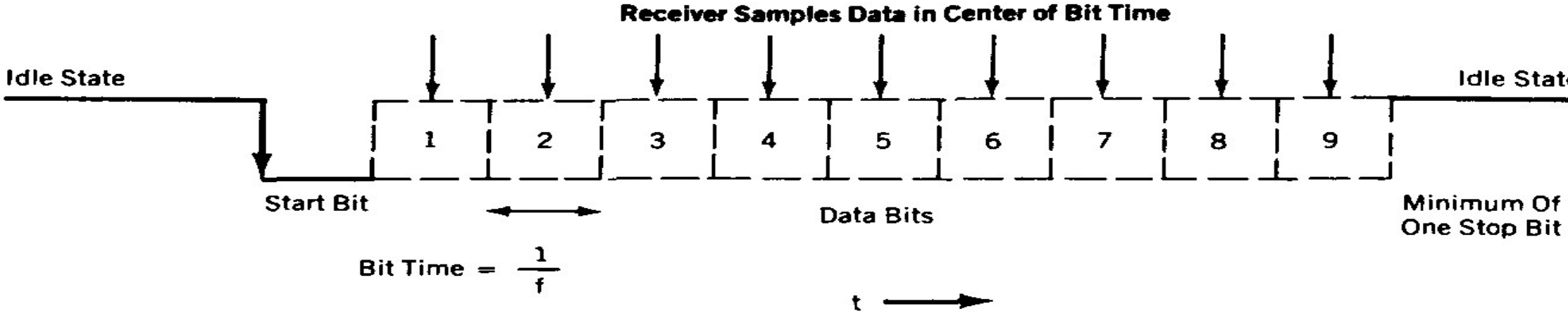
Mode 1 – Standard UART



Mode 1 Baud Rates

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

Mode 2 – Multiprocessor Mode



The baud rate is programmed as follows:

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

Mode 3

- Similar to Mode 2
- Baud rate is same as Mode 1 baud rate

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

ALP to Transfer “LBRCE” (4800, 8-Bit)

```
        MOV   TMOD, #20H    // Timer 1 in Mode 2
        MOV   TH1, FAH      // Baud rate 4800
        MOV   SCON, #50H    // Serial port in Mode 1
        SETB  TR1           // Timer 1 Run bit is set to 1
        MOV   DPTR, #4000H  // Memory Initialization for “LBRCE”
        MOV   R1, #05H      // Count for “LBRCE” 5 characters
START:  MOVX  A, @DPTR      // Get the character in to A
        MOV   SBUF, A               // Send the character to Serial Port
HERE:   JNB   TI, HERE              // Check the Transmitter Interrupt
        CLR   TI            // Clear the Transmitter Interrupt
        INC   DPTR          // Point to the next character
        DJNZ  R1, START     // Decrement the count & jump if not “0”
        END
```

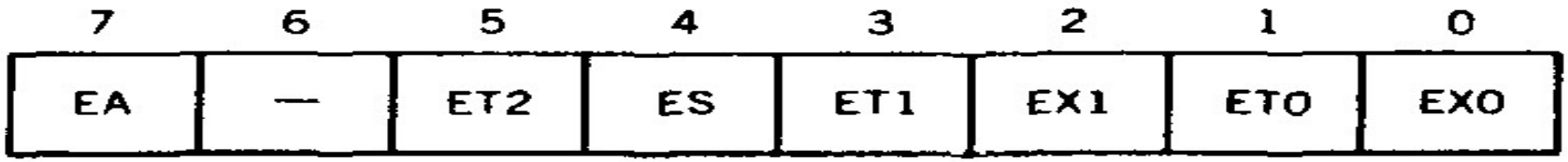
ALP to Receive 100 bytes & Sent to P2 (9600, 8-Bit)

```
        MOV   TMOD, #20H   // Timer 1 in Mode 2
        MOV   TH1, FDH     // Baud rate 9600
        MOV   SCON, #50H   // Serial port in Mode 1
        MOV   R2, #64H     // Count for 100 bytes
        SETB  TR1          // Timer 1 Run bit is set to 1
HERE:   JNB   RI, HERE     // Check the Receiver Interrupt
        MOV   A, SBUF              // Receive Data from Serial Port
        MOV   P2, A        // Send the Data to Port 2
        CLR   RI           // Clear the Receiver Interrupt
        DJNZ  R2, HERE     // Decrement the count & jump if not "0"
        END
```

Interrupts

- 8051 supports 5 interrupts
- Two external interrupts
  - INT0 and
  - INT1
- Three internal interrupts
  - Timer 0
  - Timer 1
  - Serial Port
- IE: To enable or disable the interrupts
- IP: To set the interrupt priorities

IE 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.

IP 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)

Interrupt Vector Addresses

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

Reset

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

All the Best