**MODULE-6**

**8051 ADDRESSING MODES**

8051 addressing modes are classified as follows.

1. Immediate addressing.

2. Register addressing.

3. Direct addressing.

4. Indirect addressing.

5. Relative addressing.

6. Indexed addressing.

7. Bit direct addressing.

**1. Immediate addressing.**

In this addressing mode the data is provided as a part of instruction itself. In other words data immediately follows the instruction.

Eg.

MOV A,#30H

ADD A, #83 # Symbol indicates the data is immediate.

**2. Register addressing.**

In this addressing mode the register will hold the data. One of the eight general registers (R0 to R7) can be used and specified as the operand.

Eg.

MOV A,R0

ADD A,R6

R0 – R7 will be selected from the current selection of register bank. The default register bank will be bank 0.

**3. Direct addressing**

There are two ways to access the internal memory. Using direct address and indirect address. Using direct addressing mode we can not only address the internal memory but SFRs also. In direct addressing, an 8 bit internal data memory address is specified as part of the instruction and hence, it can specify the address only in the range of 00H to FFH. In this addressing mode, data is obtained directly from the memory.

Eg.

MOV A,60h

ADD A,30h

**4. Indirect addressing**

The indirect addressing mode uses a register to hold the actual address that will be used in data movement. Registers R0 and R1 and DPTR are the only registers that can be used as data pointers. Indirect addressing cannot be used to refer to SFR registers. Both R0 and R1 can hold 8 bit address and DPTR can hold 16 bit address.

Eg.

MOV A,@R0

ADD A,@R1

MOVX A,@DPTR

**5. Indexed addressing.**

In indexed addressing, either the program counter (PC), or the data pointer (DTPR)—is used to hold the base address, and the A is used to hold the offset address. Adding the value of the base address to the value of the offset address forms the effective address. Indexed addressing is used with JMP or MOVC instructions. Look up tables are easily implemented with the help of index addressing.

Eg.

MOVC A, @A+DPTR // copies the contents of memory location pointed by the sum of the accumulator A and the DPTR into accumulator A.

MOVC A, @A+PC // copies the contents of memory location pointed by the sum of the accumulator A and the program counter into accumulator A.

**6. Relative Addressing.**

Relative addressing is used only with conditional jump instructions. The relative address, (offset), is an 8 bit signed number, which is automatically added to the PC to make the address of the next instruction. The 8 bit signed offset value gives an address range of +127 to —128 locations. The jump destination is usually specified using a label and the assembler calculates the jump offset accordingly. The advantage of relative addressing is that the program code is easy to relocate and the address is relative to position in the memory.

Eg.

SJMP LOOP1

JC BACK

**7.Bit Direct Addressing**

In this addressing mode the direct address of the bit is specified in the instruction. The RAM space 20H to 2FH and most of the special function registers are bit addressable. Bit address values are between 00H to 7FH.

Eg.

CLR 07h ; Clears the bit 7 of 20h RAM space

SETB 07H ; Sets the bit 7 of 20H RAM space.

**8051 INSTRUCTION SET**

**1. Instruction Timings**

The 8051 internal operations and external read/write operations are controlled by the oscillator clock. **T-state, Machine cycle and Instruction cycle** are terms used in instruction timings.

**T-state** is defined as one subdivision of the operation performed in one clock period.

**Machine cycle** is defined as 12 oscillator periods. A machine cycle consists of six states and each state lasts for two oscillator periods. An instruction takes one to four machine cycles to execute an instruction.

**Instruction cycle** is defined as the time required for completing the execution of an instruction. The 8051 instruction cycle consists of one to four machine cycles.

The instructions of 8051 can be broadly classified under the following headings.

1. Data transfer instructions

2. Arithmetic instructions

3. Logical instructions

4. Branch instructions

5. Subroutine instructions

6. Bit manipulation instructions

**DATA TRANSFER INSTRUCTIONS**

In this group, the instructions perform data transfer operations of the following types.

**a. Move the contents of a register Rn to A**

i. MOV A,R2

**b. Move the contents o f a register A to Rn**

i. MOV R4,A

**c. Move an immediate 8 bit data to register A or to Rn or to a memory location(direct or indirect)**

i. MOV A, #45H

ii. MOV R6, #51H

iii. MOV 30H, #44H

iv. MOV @R0, #0E8H

v. MOV DPTR, #0F5A2H

**d. Move the contents of a memory location to A or A to a memory location using direct and indirect addressing**

i. MOV A, 65H

ii. MOV A, @R0

iii. MOV 45H, A

iv. MOV @R1, A

**e. Move the contents of a memory location to Rn or Rn to a memory location using direct addressing**

i. MOV R3, 65H

ii. MOV 45H, R2

**f. Move the contents of memory location to another memory location using direct and indirect addressing**

i. MOV 47H, 65H

ii. MOV 45H, @R0

**g. Move the contents of an external memory to A or A to an external memory**

i. MOVX A,@R1

ii. MOVX @R0,A

iii. MOVX A,@DPTR

iv. MOVX@DPTR,A

**h. Move the contents of program memory to A**

i. MOVC A, @A+PC

ii. MOVC A, @A+DPTR

**j. Exchange instructions :-The content of source ie., register, direct memory or indirect memory will be exchanged with the contents of destination ie., accumulator**.

i. XCH A,R3

ii. XCH A,@R1

iii. XCH A,54h

**k. Exchange digit:- Exchange the lower order nibble of Accumulator (A0-A3) with lower order nibble of the internal RAM location which is indirectly addressed by the register.**

i. XCHD A,@R1

ii. XCHD A,@R0

**ARITHMETIC INSTRUCTIONS**

The 8051 can perform addition, subtraction. Multiplication and division operations on 8 bit numbers.

**1.Addition**

In this group, we have instructions to

**i. Add the contents of A with immediate data with or without carry.**

i. ADD A, #45H

ii. ADDC A, #OB4H

**ii. Add the contents of A with register Rn with or without carry.**

i. ADD A, R5

ii. ADDC A, R2

**iii. Add the contents of A with contents of memory with or without carry using direct and indirect addressing**

i. ADD A, 51H

ii. ADDC A, 75H

CY AC and OV flags will be affected by this operation.

**2.Subtraction**

In this group, we have instructions to

**i. Subtract the contents of A with immediate data with or without carry.**

i. SUBB A, #45H

ii. SUBB A, #OB4H

**ii. Subtract the contents of A with register Rn with or without carry.**

i. SUBB A, R5

ii. SUBB A, R2

**iii. Subtract the contents of A with contents of memory with or without carry using direct and indirect addressing**

i. SUBB A, 51H

ii. SUBB A, 75H

iii. SUBB A, @R1

**3.Multiplication**

**MUL AB.**

This instruction multiplies two 8 bit unsigned numbers which are stored in A and B register. After multiplication the lower byte of the result will be stored in accumulator and higher byte of result will be stored in B register.

Eg.

MOV A,#45H ; [A]=45H

MOV B,#0F5H ; [B]=F5H

MUL AB ; [A] x [B] = 45 x F5 = 4209 ;

[A]=09H, [B]=42H

**4.Division**

**DIV AB.**

This instruction divides the 8 bit unsigned number which is stored in A by the 8 bit unsigned number which is stored in B register. After division the result will be stored in accumulator and remainder will be stored in B register.

Eg.

MOV A,#45H ; [A]=0E8H

MOV B,#0F5H ; [B]=1BH

DIV AB ; [A] / [B] = E8 /1B = 08 H with remainder 10H ;

[A] = 08H, [B]=10H

5. **DA A (Decimal Adjust After Addition).**

When two BCD numbers are added, the answer is a non-BCD number. To get the result in BCD, we use DA A instruction after the addition. DA A works as follows. If lower nibble is greater than 9 or auxiliary carry is 1, 6 is added to lower nibble. If upper nibble is greater than 9 or carry is 1, 6 is added to upper nibble.

Eg 1:

MOV A,#23H

MOV R1,#55H

ADD A,R1 // [A]=78

DA A // [A]=78 no changes in the accumulator after da a

Eg 2:

MOV A,#53H

MOV R1,#58H

ADD A,R1 // [A]=AB

DA A // [A]=11, C=1 .

ANSWER IS 111. Accumulator data is changed after DA A

**6. Increment:**

INC increments the value of source by 1. If the initial value of register is FFh, incrementing the value will cause it to reset to 0. The Carry Flag is not set when the value "rolls over" from 255 to 0. In the case of "INC DPTR", the value two-byte unsigned integer value of DPTR is incremented. If the initial value of DPTR is FFFFh, incrementing the value will cause it to reset to 0.

INC A

INC Rn

INC DIRECT

INC @Ri

INC DPTR

**7.Decrement:**

DEC decrements the value of source by 1. If the initial value of is 0, decrementing the value will cause it to reset to FFh. The Carry Flag is not set when the value "rolls over" from 0 to FFh.

DEC A

DEC Rn

DEC DIRECT

DEC @Ri

**LOGICAL INSTRUCTIONS**

1. **Logical AND**

**ANL** destination, source

ANL does a bitwise "AND" operation between source and destination, leaving the resulting value in destination. The value in source is not affected. "AND" instruction logically AND the bits of source and destination.

Eg:-

ANL A,#DATA

ANL A, Rn

1. **Logical OR**

**ORL destination, source**

ORL does a bitwise "OR" operation between source and destination, leaving the resulting value in destination. The value in source is not affected. " OR " instruction logically OR the bits of source and destination.

Eg:-

ORL A,#DATA

ORL A, Rn

ORL A,DIRECT

1. **Logical Ex-OR**

**XRL destination, source**

XRL does a bitwise "EX-OR" operation between source and destination, leaving the resulting value in destination. The value in source is not affected. " XRL " instruction logically EX-OR the bits of source and destination.

Eg:-

XRL A,#DATA

XRL A,Rn

XRL A,DIRECT

XRL A,@Ri

1. **Logical NOT**

CPL complements operand, leaving the result in operand. If operand is a single bit then the state of the bit will be reversed. If operand is the Accumulator then all the bits in the Accumulator will be reversed.

Eg:-

CPL A, CPL C, CPL bit address

SWAP A – Swap the upper nibble and lower nibble of A.

**Rotate Instructions**

**RR A**

This instruction is rotate right the accumulator. Its operation is illustrated below. Each bit is shifted one location to the right, with bit 0 going to bit 7.

**RL A**

Rotate left the accumulator. Each bit is shifted one location to the left, with bit 7 going to bit 0

**RRC A**

Rotate right through the carry. Each bit is shifted one location to the right, with bit 0 going into the carry bit in the PSW, while the carry was at goes into bit 7

**RLC A**

Rotate left through the carry. Each bit is shifted one location to the left, with bit 7 going into the carry bit in the PSW, while the carry goes into bit 0.

**BRANCH INSTRUCTIONS**

There are 3 types of jump instructions. They are:-

1. Relative Jump

2. Short Absolute Jump

3. Long Absolute Jump

**Relative Jump**

Jump that replaces the PC (program counter) content with a new address that is greater than (the address following the jump instruction by 127 or less) or less than (the address following the jump by 128 or less) is called a relative jump.

Eg:-

SJMP *<relative address>*

JC *<relative address>*

**Short Absolute Jump**

In this case only 11bits of the absolute jump address are needed.

Example of short absolute jump: -

ACALL *<relative address>*

*AJMP <relative address>*

**Long Absolute Jump/Call**

Applications that need to access the entire program memory from 0000H to FFFFH use long absolute jump. Since the absolute address has to be specified in the op-code, the instruction length is 3 bytes (except for JMP @ A+DPTR). This jump is not re-locatable.

Example: -

LCALL *<16 bit address>*

*LJMP <16 bit address>*

JMP @A+DPTR

**SUBROUTINE INSTRUCTIONS**

Subroutines are handled by CALL and RET instructions There are two types of CALL instructions

1. **LCALL address(16 bit)**

This is long call instruction which unconditionally calls the subroutine located at the indicated 16 bit address. This is a 3 byte instruction. The LCALL instruction works as follows.

a. During execution of LCALL, [PC] = [PC]+3; (if address where LCALL resides is say, 0x3254; during execution of this instruction [PC] = 3254h + 3h = 3257h

b. [SP]=[SP]+1; (if SP contains default value 07, then SP increments and [SP]=08)

c. [[SP]] = [PC7-0]; (lower byte of PC content ie., 57 will be stored in memory location 08.

d. [SP]=[SP]+1; (SP increments again and [SP]=09)

e. [[SP]] = [PC15-8]; (higher byte of PC content ie., 32 will be stored in memory location 09.

With these the address (0x3254) which was in PC is stored in stack.

f. [PC]= address (16 bit);the new address of subroutine is loaded to PC. No flags are affected

1. **ACALL address(11 bit)**

This is absolute call instruction which unconditionally calls the subroutine located at the indicated 11 bit address. This is a 2 byte instruction. The ACALL instruction works as follows.

a. During execution of SCALL, [PC] = [PC]+2; (if address where LCALL resides is say, 0x8549; during execution of this instruction [PC] = 8549h + 2h = 854Bh

b. [SP]=[SP]+1; (if SP contains default value 07, then SP increments and [SP]=08

c. [[SP]] = [PC7-0]; (lower byte of PC content ie., 4B will be stored in memory location 08.

d. [SP]=[SP]+1; (SP increments again and [SP]=09)

e. [[SP]] = [PC15-8]; (higher byte of PC content ie., 85 will be stored in memory location 09.

With these the address (0x854B) which was in PC is stored in stack.

f. [PC10-0]= address (11 bit); the new address of subroutine is loaded to PC. No flags are affected.

**RET instruction**

RET instruction pops top two contents from the stack and load it to PC.

g. [PC15-8] = [[SP]] ;content of current top of the stack will be moved to higher byte of PC.

h. [SP]=[SP]-1; (SP decrements)

i. [PC7-0] = [[SP]] ;content of bottom of the stack will be moved to lower byte of PC.

j. [SP]=[SP]-1; (SP decrements again)

**BIT MANIPULATION INSTRUCTIONS**

8051 has 128 bit addressable memory. Bit addressable SFRs and bit addressable PORT pins. It is possible to perform following bit wise operations for these bit addressable locations.

**1. LOGICAL AND**

a. ANL C,BIT(BIT ADDRESS) ; ‘LOGICALLY AND’ CARRY AND CONTENT OF BIT ADDRESS, STORE RESULT IN CARRY

b. ANL C, /BIT; ; ‘LOGICALLY AND’ CARRY AND COMPLEMENT OF CONTENT OF BIT ADDRESS, STORE RESULT IN CARRY

**2. LOGICAL OR**

a. ORL C,BIT(BIT ADDRESS) ; ‘LOGICALLY OR’ CARRY AND CONTENT OF BIT ADDRESS, STORE RESULT IN CARRY

b. ORL C, /BIT; ; ‘LOGICALLY OR’ CARRY AND COMPLEMENT OF CONTENT OF BIT ADDRESS, STORE RESULT IN CARRY

**3. CLR bit**

a. CLR bit ; CONTENT OF BIT ADDRESS SPECIFIED WILL BE CLEARED.

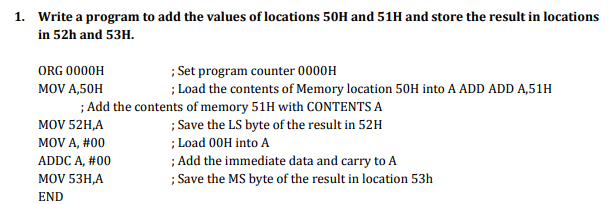
b. CLR C ; CONTENT OF CARRY WILL BE CLEARED.

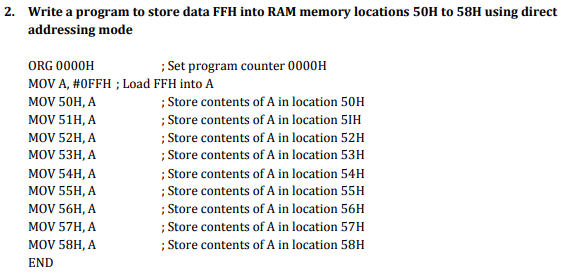
**4. CPL bit**

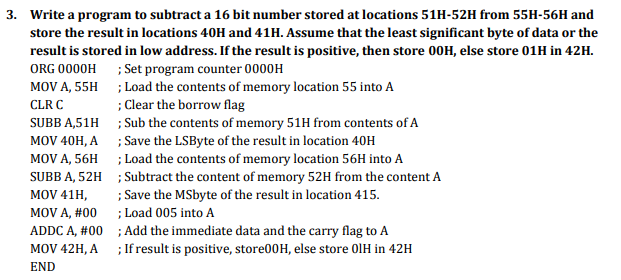
a. CPL bit ; CONTENT OF BIT ADDRESS SPECIFIED WILL BE COMPLEMENTED.

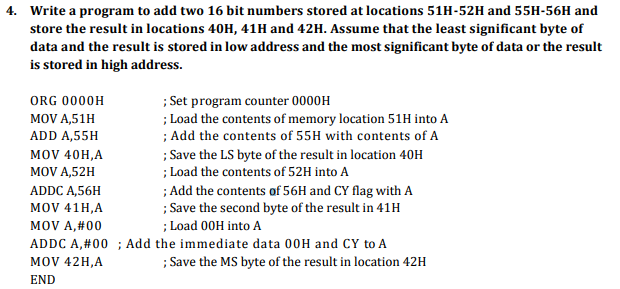
b. CPL C ; CONTENT OF CARRY WILL BE COMPLEMENTED.

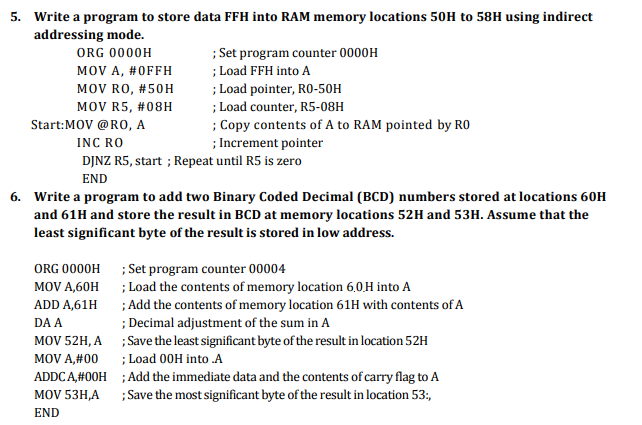
**8051 SIMPLE PROGRAMS**

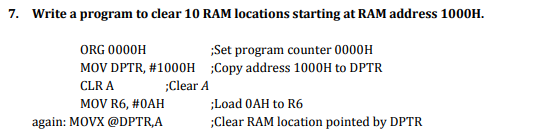


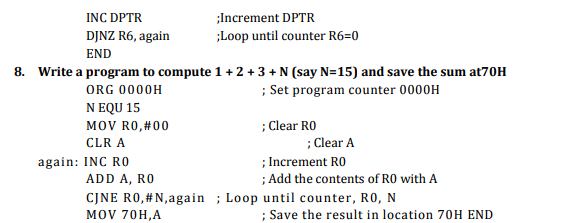


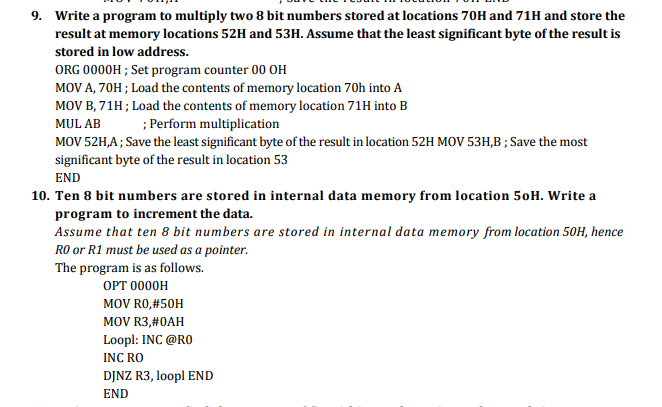


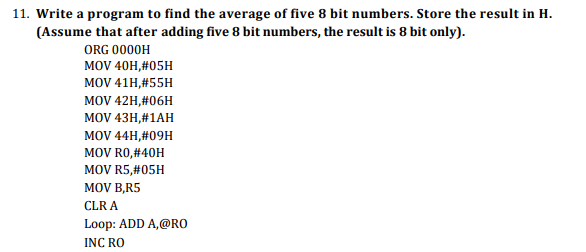


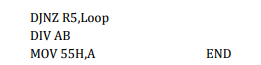


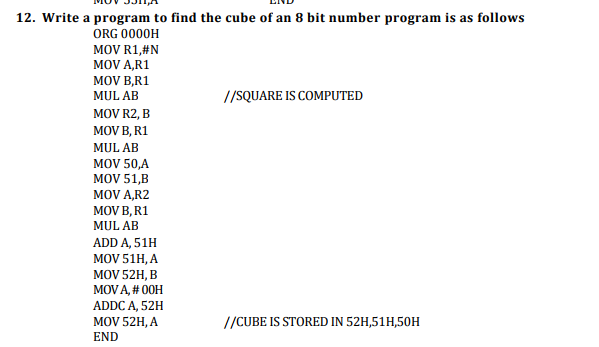


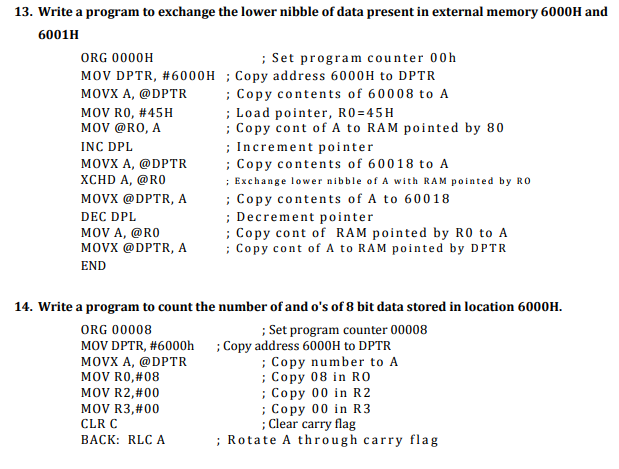


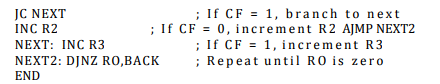


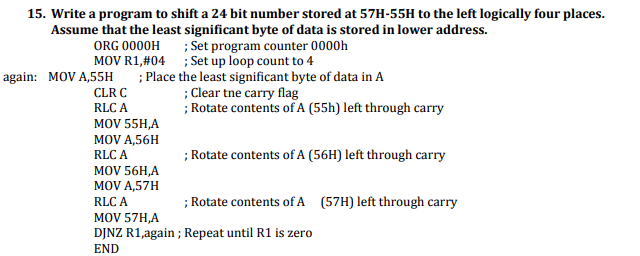












**8254/8253 PROGRAMMABLE INTERVAL TIMER**

The Intel 8253 and 8254 are Programmable Interval Timers (PTIs) designed for microprocessors to perform timing and counting functions using three 16-bit registers. Each counter has 2 input pins, i.e. Clock & Gate, and 1 pin for “OUT” output. To operate a counter, a 16-bit count is loaded in its register. On command, it begins to decrement the count until it reaches 0, then it generates a pulse that can be used to interrupt the CPU.

## Difference between 8253 and 8254

The following table differentiates the features of 8253 and 8254 −

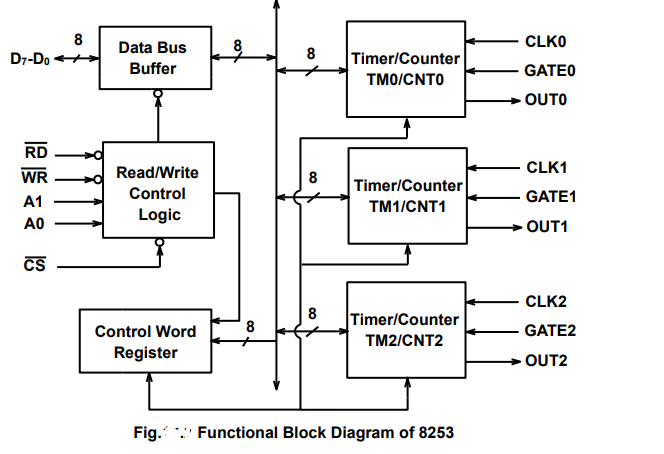
|  |  |
| --- | --- |
| **8253** | **8254** |
| Its operating frequency is 0 - 2.6 MHz | Its operating frequency is 0 - 10 MHz |
| It uses N-MOS technology | It uses H-MOS technology |
| Read-Back command is not available | Read-Back command is available |
| Reads and writes of the same counter cannot be interleaved. | Reads and writes of the same counter can be interleaved. |

## Features of 8253 / 54

The most prominent features of 8253/54 are as follows −

* It has three independent 16-bit down counters.
* It can handle inputs from DC to 10 MHz.
* These three counters can be programmed for either binary or BCD count.
* It is compatible with almost all microprocessors.
* 8254 has a powerful command called READ BACK command, which allows the user to check the count value, the programmed mode, the current mode, and the current status of the counter.

**8253/54 ARCHITECTURE**

****

In the above figure, there are three counters, a data bus buffer, Read/Write control logic, and a control register. Each counter has two input signals - CLOCK & GATE, and one output signal - OUT.

### Data Bus Buffer

It is a tri-state, bi-directional, 8-bit buffer, which is used to interface the 8253/54 to the system data bus. The operation of this buffer is controlled by the chip select line (CS ) which tells the timer chip that the microprocessor is communicating with it, i.e., trying to transfer information to or from it even though CS is part of the READ/WRITE control logic. Data is transmitted or received by the buffer upon execution of IN PORT or OUR PORT instruction from CPU.It has three basic functions −

* Programming the modes of 8253/54.
* Loading the count registers.
* Reading the count values.

The data bus buffer is connected to processor using D7-D0 pins which are bidirectional. The data transfer takes place through these pins.

### Read/Write Logic

It accepts inputs for the system control bus and in turn generates the control signals for overall device operation. It is enabled or disabled by CS so that no operation can occur to change the function unless the device has been selected by the system logic

It includes 5 signals, i.e. RD, WR, CS, and the address lines A0 & A1. In the peripheral I/O mode, the RD and WR signals are connected to IOR and IOW, respectively. In the memorymapped I/O mode, these are connected to MEMR and MEMW.

Address lines A0 & A1 of the CPU are connected to lines A0 and A1 of the 8253/54, and CS is tied to a decoded address. The control word register and counters are selected according to the signals on lines A0 & A1.

|  |  |  |
| --- | --- | --- |
| **A1** | **A0** | **Result** |
| 0 | 0 | Counter 0 |
| 0 | 1 | Counter 1 |
| 1 | 0 | Counter 2 |
| 1 | 1 | Control Word Register |
| X | X | No Selection |

### CS :

### The chip select input is used to enable the communication between 8253 chip and the microprocessor by means of data bus. A low on CS enables the data bus buffers, while a high disables the buffer. The CS input does not have any affect on the operation of three timers once they have been initialized.

**Counters:**

Each of the timers has three pins associated with it. These are clock (CLK) input, the gate (GATE) control input and the output (OUT).

The clock input pin provides 16-bit timer with the signal that causes the timer to decrement. The maximum clock frequency input is 2.6MHz.

The gate input pin is used to initiate or enable counting. The exact effect of the gate signal depends on which of the six modes of operation is chosen.

The output pin provides an output from the timer. Its actual use depends on the mode of operation of the timer.

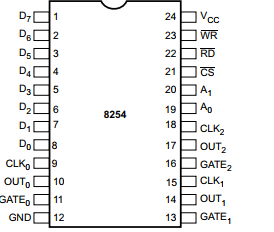
### Control Word Register

This register is accessed when lines A0 & A1 are at logic 1. It is used to write a command word, which specifies the counter to be used, its mode, and either a read or write operation.

**Counters**

Each counter consists of a single, 16 bit-down counter, which can be operated in either binary or BCD. Its input and output is configured by the selection of modes stored in the control word register. The programmer can read the contents of any of the three counters without disturbing the actual count in process.

**Pin Diagram**

****

**Clock** This is the clock input for the counter. The counter is 16 bits. The maximum clock frequency is 1 / 380 nanoseconds or 2.6 megahertz.

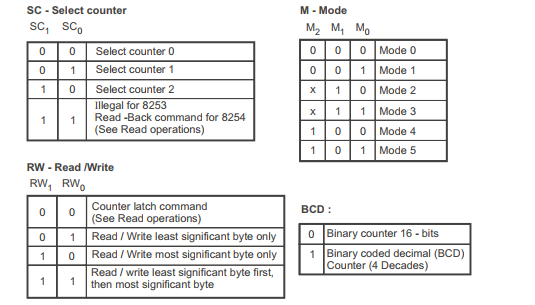
**Out** This single output line is the signal that is the final programmed output of the device. Actual operation of the out line depends on how the device has been programmed.

**Gate** This input can act as a gate for the clock input line, or it can act as a start pulse, depending on the programmed mode of the counter.

**Control Word of 8253/54**

Each counter of the 8253/54 is individually programmed by writing a control word into the control word register





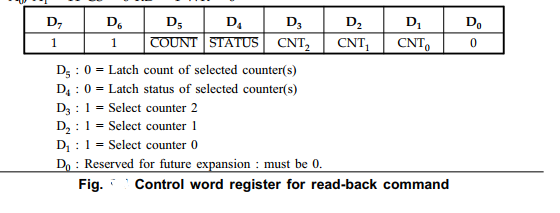
**READ Operation :**

In some applications, especially in event counters, it is necessary to read the value of the count in process. This can be done by three possible methods:

1. Simple Read : It involves reading a count after inhibiting the counter by controlling the gate input or the clock input of the selected counter, and two I/O read operations are performed by the CPU. The first I/O operation reads the low-order byte, and the second I/O operation reads the high order byte.

2. Counter Latch Command : In the second method, an appropriate control word is written into the control register to latch a count in the output latch, and two I/O read operations are performed by the CPU. The first I/O operation reads the low-order byte, and the second I/O operation reads the high order byte.

3. Read-Back Command (Available only for 8254) : The third method uses the Read-Back command. This command allows the user to check the count value, programmed Mode, and current status of the OUT pin and Null count flag of the selected counter.



**MODES OF OPERATION**

8253/54 can be operated in 6 different modes.

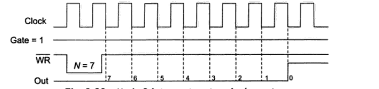
**MODE 0: INTERRUPT ON TERMINAL COUNT**

In this mode OUT is low. Once a count is loaded the counter is decremented after every cycle,

and when count reaches zero, the OUT goes high.

This can be used as an interrupt. The OUT remains high until a new count or command word is

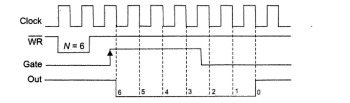
loaded.



**MODE 1: HARDWARE RETRIGGERABLE ONE SHOT**

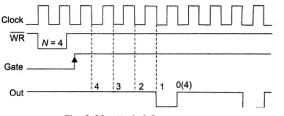
In this mode OUT is intially high. When gate is triggered, the OUT goes low, and at the end of

count it goes high again, thus generating a one shot pulse.



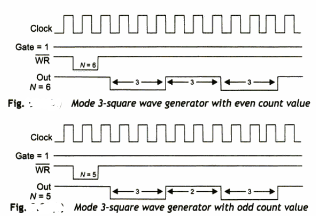
**MODE 2: RATE GENERATOR**

The mode is used to generate a pulse equal to given clock period at a given interval. When a count is loaded, the OUT stays high until count reaches 1 and then OUT goes low for 1 clock period then gets reloaded automatically and this is how pulse gets generated  continuously.



**MODE 3: SQUARE WAVE GENERATOR**

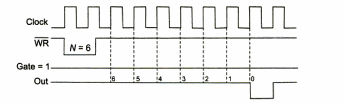
In this a continuous square wave with period equal to count is generated.  The frequency of square wave = frequency of clock divide by count. if count (N) is odd pulse stay high for (N + 1)/2 and low for (N – 1)/2. if count (N) is even pulse stay high for (N )/2 and low for (N)/2.



**MODE 4: SOFTWARE TRIGGERED STROBE**

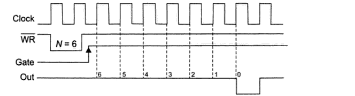
In this mode OUT is initially high, it goes low for one clock period at the end of count.

The count must be reloaded for subsequent outputs.



**MODE 5: HARDWARE TRIGGERED STROBE**

Same as MODE4 except that it is triggered by rising pulse at gate.

****

**USE OF 8253/54 PROGRAMMABLE TIMER**

Intel 8253 programmable Timer/ counter is a specially designed chip for Intel microcomputer applications which require timing and counting operations. These timing and counting functions can be implemented through software.

For example, let in an application, microprocessor is required to execute N different tasks and these tasks are to be executed at an interval of T seconds. The software solution would be to call a delay routine of T seconds after a task (say for example, „i‟) is completed and then do next task (say „j‟). In a software delay subroutine, either a register or a register pair is to initialized and decremented continuously till it become zero. In order to maintain the precision of the delay, it will not be possible for the microprocessor to execute any other task during this interval. If there are more such tasks, then microprocessor will be busy most of the time to execute the delay routines. If microprocessor has to perform some other useful task during (calculation), which is common in control applications, then it is very difficult.

The other possible solution is use of external timer. The µC may start this timer with a programmable value after executing the task „i‟, then µC is free to do something else. This external timer may be down counter or up counter. The external timer after a delay of T seconds interrupts the µP. The µC executes task „j‟ once it gets this interrupt. Such external device is called a programmable timer.

The Intel 8253 is a programmable counter/timer chip designed for use as an Intel µC peripheral. INTEL 8253 chip consists of three identical 16-bit timers TM0, TM1 and TM2. The timers are basically 16- bit down counters and counts HIGH to LOW transition at CLK input. Each timer may be programmed to operator in one of the six modes, independent of the mode of operation of the other two timers. The timers are software programmable.The maximum clock input to the timer is 2.6 MHz.