KUNTHAVAI NAACHIYAAR GOVT. ARTS OLLEGE FOR WOMEN (AUTONOMOUS), THANJAVUR – 7.

DEPARTMENT OF PHYSICS

M. SC PHYSICS - STUDY MATERIAL

SUB. CODE: 18KP3PELP4

SUB. TITLE: MICROPROCESSOR AND MICROCONTROLLER

UNIT – III

MICROCONTROLLER

Microcontroller

A microcontroller is a small, low-cost and self-contained computer-on-a-chip which is designed to perform the specific tasks of embedded systems like displaying microwave's information, receiving remote signals, etc. Microcontrollers usually have

- An 8 or 16 bit microprocessor.
- A little measure of RAM.
- Programmable ROM and flash memory.
- Parallel and serial I/O.
- Timers and signal generators.
- Analog to Digital and Digital to Analog conversion

Difference between microprocessor and microcontroller

As now you are basically aware of what is a microcontroller and microprocessor, it would be easy to identify the major differences between a microcontroller and microprocessor.

1. Key difference in both of them is presence of external peripheral, where microcontrollers have RAM, ROM, EEPROM embedded in it while we have to use external circuits in case of microprocessors.

2. As all the peripheral of microcontroller are on single chip it is compact while microprocessor is bulky.

3. Microcontrollers are made by using complementary metal oxide semiconductor technology so they are far cheaper than microprocessors. In addition, the applications made with microcontrollers are cheaper because they need lesser external components, while the overall cost of systems made with microprocessors are high because of the high number of external components required for such systems.

4. Processing speed of microcontrollers is about 8 MHz to 50 MHz, but in contrary processing speed of general microprocessors is above 1 GHz so it works much faster than microcontrollers.

5. Generally microcontrollers have power saving system, like idle mode or power saving mode so overall it uses less power and also since external components are low overall consumption of power is less. While in microprocessors generally there is no power saving system and also many external components are used with it, so its power consumption is high in comparison with microcontrollers.

6. Microcontrollers are compact so it makes them favourable and efficient system for small products and applications while microprocessors are bulky so they are preferred for larger applications.

7. Tasks performed by microcontrollers are limited and generally less complex. While task performed by microprocessors are software development, Game development, website, documents making etc. which are generally more complex so require more memory and speed so that's why external ROM, RAM are used with it.

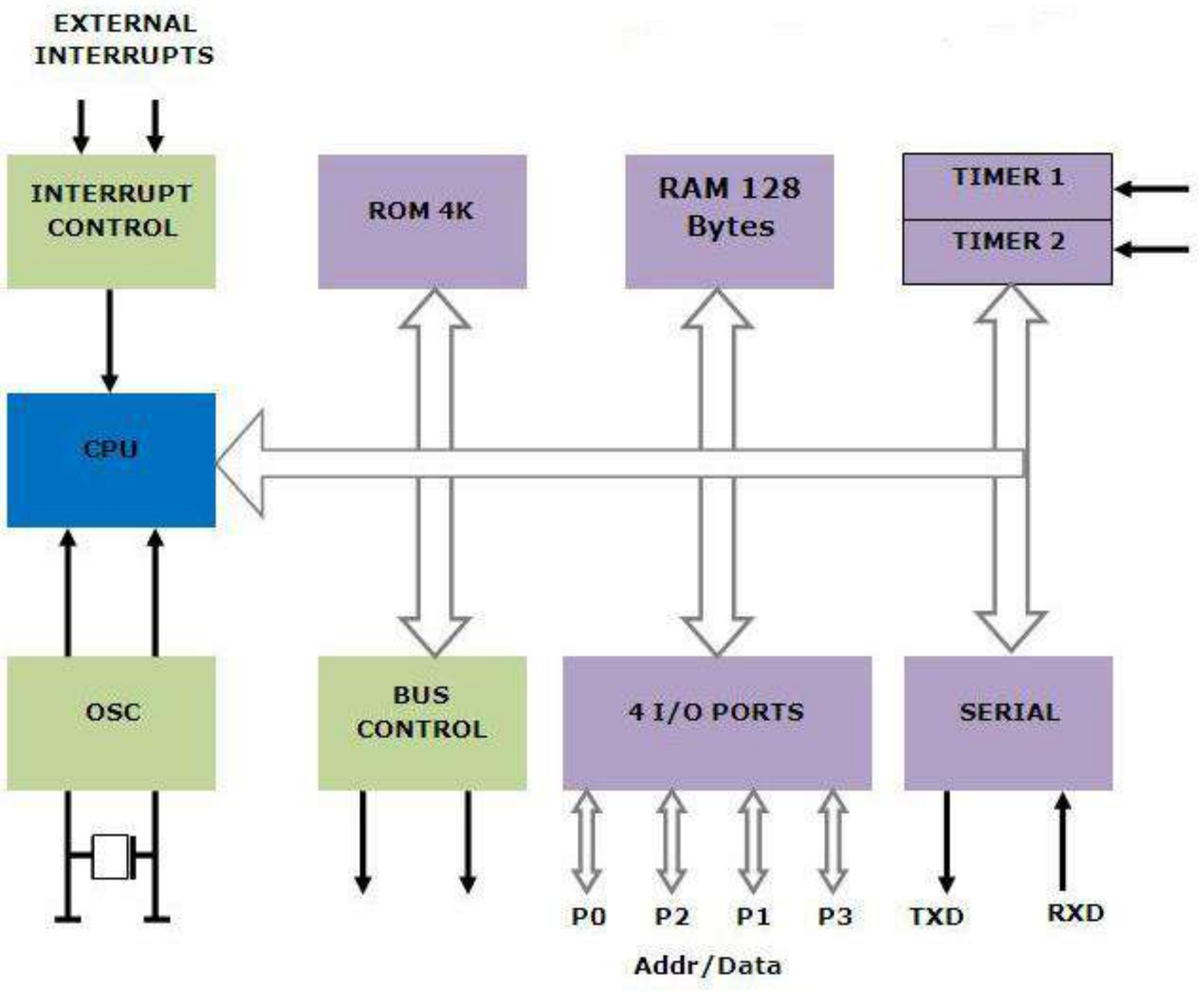
8. Microcontrollers are based on Harvard architecture where program memory and data memory are separate while microprocessors are based on von Neumann model where program and data are stored in same memory module.

Difference between Microprocessor and Microcontroller

The following table highlights the differences between a microprocessor and a microcontroller:

Microcontroller	Microprocessor
Microcontrollers are used to execute a single task within an application.	Microprocessors are used for big applications.
Its designing and hardware cost is low.	Its designing and hardware cost is high.
Easy to replace.	Not so easy to replace.
It is built with CMOS technology, which requires less power to operate.	Its power consumption is high because it has to control the entire system.
It consists of CPU, RAM, ROM, I/O ports.	It doesn't consist of RAM, ROM, I/O ports. It uses its pins to interface to peripheral devices.

Architecture of microcontroller 8051:



CPU (Central Processor Unit):

The Central Processor Unit or CPU is the mind of any processing machine. It scrutinizes and manages all processes that are carried out in the Microcontroller. User has no power over the functioning of CPU. It interprets program printed in storage space (ROM) and carries out all of them and do the projected duty. CPU manages different types of registers in 8051 microcontroller.

Interrupts:

Interrupt is a sub-routine call that reads the Microcontroller’s key function or job and helps it to perform some other program which is extra important at that point of time. The characteristic of 8051 Interrupt is extremely helpful as it aids in emergency cases. Interrupts provides us a method to postpone or delay the current process, carry out a sub-routine task and then all over again restart standard program implementation.

The Micro-controller 8051 can be assembled in such a manner that it momentarily stops or break the core program at the happening of interrupt. When sub-routine task is finished then the implementation of core program initiates automatically as usual. There are 5 interrupt supplies in 8051 Microcontroller, two out of five are peripheral interrupts, two are timer interrupts and one is serial port interrupt.

Memory:

Micro-controller needs a program which is a set of commands. This program enlightens Microcontroller to perform precise tasks. These programs need a storage space on which they can be accumulated and interpret by Microcontroller to act upon any specific process. The memory which is brought into play to accumulate the program of Microcontroller is recognized as Program memory or code memory. In common language it’s also known as Read Only Memory or ROM.

Microcontroller also needs a memory to store data or operands for the short term. The storage space which is employed to momentarily data storage for functioning is acknowledged as Data Memory and we employ Random Access Memory or RAM for this principle reason. Microcontroller 8051 contains code memory or program memory 4K so that is has 4KB Rom and it also comprise of data memory (RAM) of 128 bytes.

Bus:

Fundamentally Bus is a group of wires which functions as a communication canal or mean for the transfer Data. These buses comprise of 8, 16 or more cables. There are two types of buses:

1. Address Bus: Microcontroller 8051 consists of 16 bit address bus. It is brought into play to address memory positions. It is also utilized to transmit the address from Central Processing Unit to Memory.

2. Data Bus: Microcontroller 8051 comprise of 8 bits data bus. It is employed to carry data.

Oscillator:

Microcontroller 8051 consists of an on-chip oscillator which works as a time source for CPU (Central Processing Unit). As the productivity thumps of oscillator are steady as a result, it facilitates harmonized employment of all pieces of 8051 microcontroller.

Input/output Port:

To gather other machinery, gadgets or peripherals with microcontroller, I/O (input/output) interfacing ports are needed. For this function Micro-controller 8051 consists of 4 input/output ports to unite it to other peripherals.

Timers/Counters:

Micro-controller 8051 is incorporated with two 16 bit counters & timers. The counters are separated into 8 bit registers. The timers are utilized for measuring the intervals, to find out pulse width etc.

Pin diagram of 8051 microcontroller :

Pin-40: Vcc is the main power source of +5V DC.

Pin 20: Vss – it represents ground (0 V) connection.

Pins 32-39: Known as Port 0 (P0.0 to P0.7) to serving as I/O ports.

Pin-31: Address Latch Enable (ALE) is used to demultiplex the address-data signal of port 0.

Pin-30: (EA) External Access input is used to enable or disable external memory interfacing. If there is no external memory requirement, this pin is always held high.

Pin- 29: Program Store Enable (PSEN) is used to read signal from external program memory.

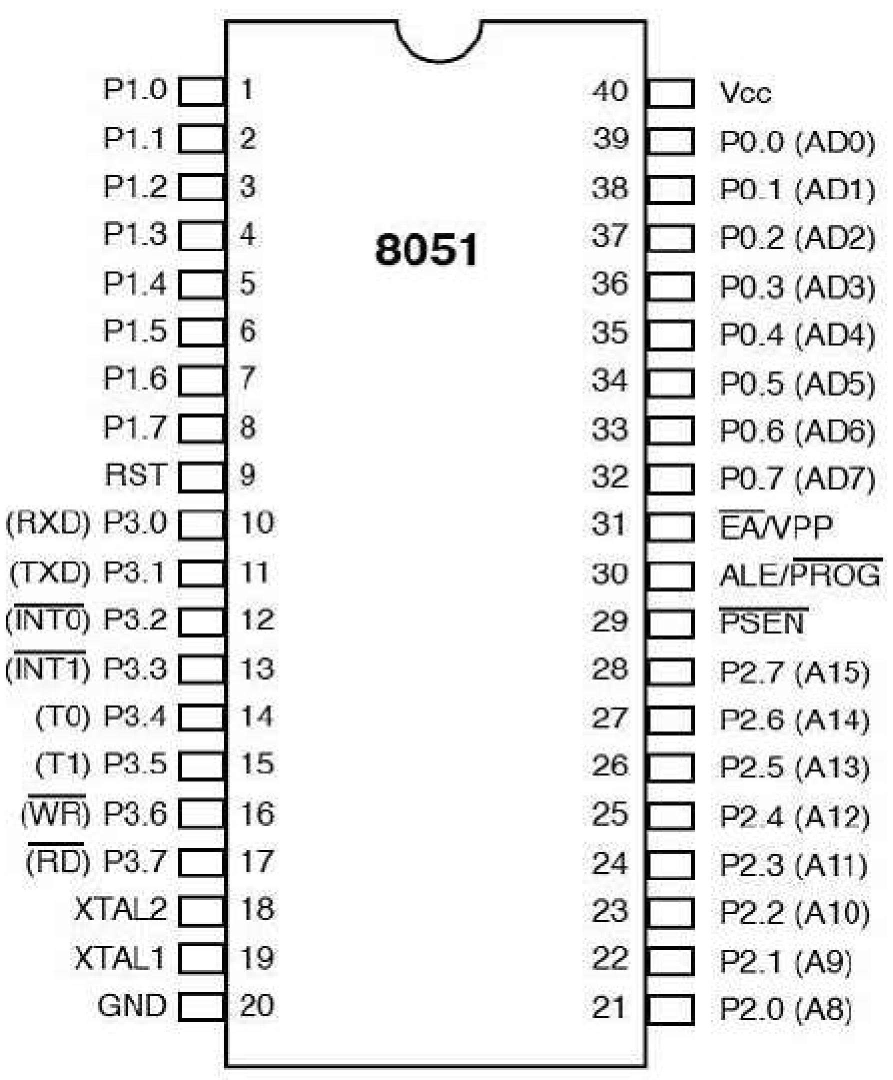
Pins- 21-28: Known as Port 2 (P 2.0 to P 2.7) – in addition to serving as I/O port, higher order address bus signals are multiplexed with this quasi bi directional port.

Pins 18 and 19: Used to interfacing an external crystal to provide system clock.

Pins 10 – 17: This port also serves some other functions like interrupts, timer input, control signals for external memory interfacing Read and Write. This is a quasi bidirectional port with internal pull up.

Pin 9: It is a RESET pin, used to set the 8051 microcontroller to its initial values, while the microcontroller is working or at the initial start of application. A RESET pin is connected to pin9, connected with a capacitor. When the switch is ON, the capacitor starts charging and RST is high. Applying a high to the reset pin resets the microcontroller. If we apply logic zero to this pin, the program starts execution from the beginning.

Pins 1 – 8: This port does not serve any other functions. Port 1 is a quasi bi directional I/O port.



Different Types of Registers in the 8051 Microcontroller

Register

A register is a small place in a CPU that can store small amounts of the data used for performing various operations such as addition and multiplication and loads the resulting data on main memory. Registers contain the address of the memory location where the data is to be stored.

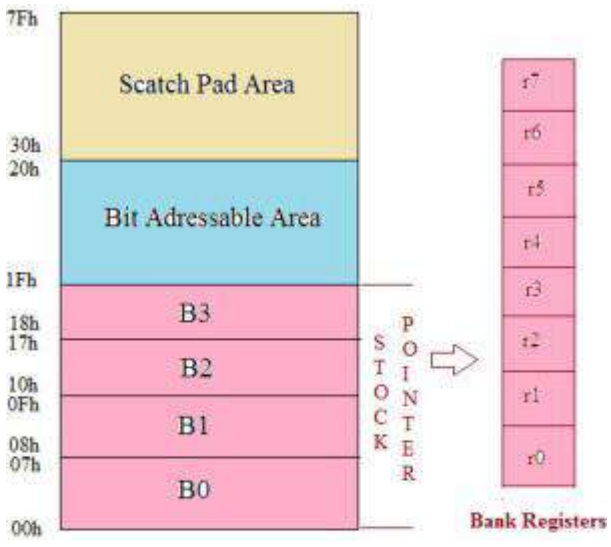
Types of Registers

The 8051 microcontroller contains mainly two types of registers:

- General purpose registers (Byte addressable registers)
- Special function registers (Bit addressable registers)

The 8051 microcontroller consists of 256 bytes of RAM memory, which is divided into two ways, such as 128 bytes for general purpose and 128 bytes for special function registers (SFR) memory. The memory which is used for general purpose is called as RAM memory, and the memory used for SFR contains all the peripheral related registers like Accumulator, 'B' register, Timers or Counters, and interrupt related registers.

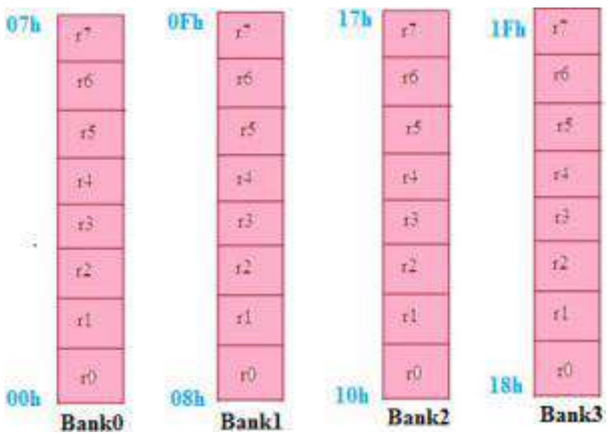
General Purpose Registers



The general purpose memory is called as the RAM memory of the 8051 microcontroller, which is divided into 3 areas such as banks, bit-addressable area, and scratch-pad area. The banks contain different general purpose registers such as R0-R7, and all such registers are byte-addressable registers that store or remove only 1-byte of data.

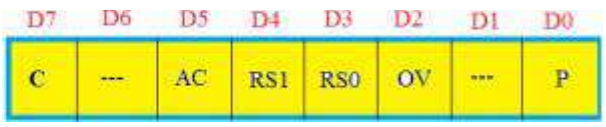
Banks and Registers

The B0, B1, B2, and B3 stand for banks and each bank contains eight general purpose registers ranging from 'R0' to 'R7'. All these registers are byte-addressable registers. Data transfer between general purpose registers to general purpose registers is not possible. These banks are selected by the Program Status Word (PSW) register.



PSW (Program Status Word) Register

The PSW register is a bit and byte-addressable register. This register reflects the status of the operation that is carried out in the controller. The PSW register determines bank selection by a RS1 and RS0, as shown below. The physical address of the PSW starts from D0h and the individual bits are accessed with D0h to D7h.



D7	D6	D5	D4	D3	D2	D1	D0
C	---	AC	RS1	RS0	OV	---	P

Carry Flag (C): The Address of the Carry flag is D7. This carry flag is affected when the bit is the 7th position. When C=0 carry resets; C=1 carry sets.

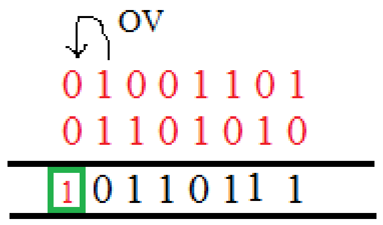
$$\begin{array}{cc} & 1\,0\,0\,0\,0\,0\,0\,0 \\ & 1\,0\,1\,1\,0\,1\,0\,1 \\ \hline \boxed{1} & 0\,0\,1\,1\,0\,1\,0\,1 \\ \hline C & \end{array}$$

Auxillary Flag(AC): The address of the auxiliary carry is D5. This auxiliary carry is affected when a bit is generated from the 3rd position to the 4th position. AC=0 auxiliary is reset; AC=1 auxiliary is set.

$$\begin{array}{c} \text{AC} \\ 0\,0\,0\,0\,1\,0\,1\,0 \\ 0\,0\,1\,0\,1\,1\,0\,1 \\ \hline 0\,0\,1\,1\,0\,1\,1\,1 \\ \hline \end{array}$$

Overflow Flag (OV): The address of the overflow flag is D2. When a bit is generated from the 6th position to the 7th position, then the overflow flag is affected.

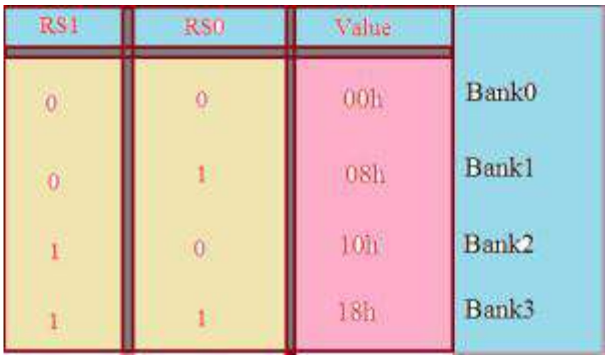
OV=0 overflow flag resets; OV=1 overflow flag sets.



Parity Flag (P): The address of the parity flag is D0. While performing arithmetic operations, if the result is 1, then the parity flag is set – otherwise reset.

RS1 and RS0

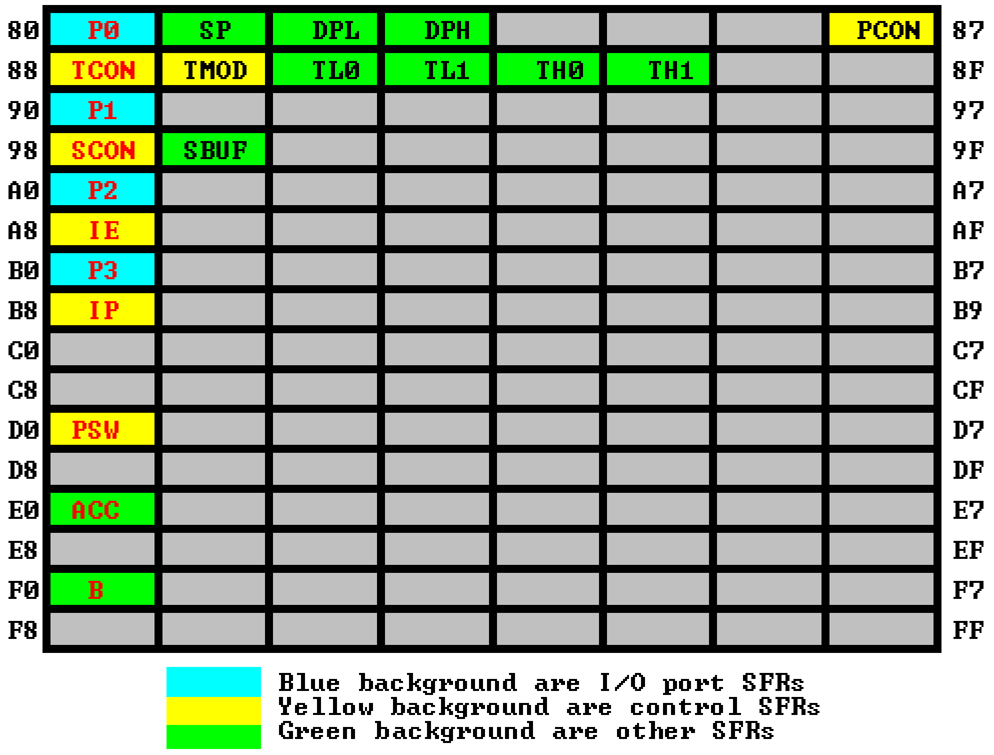
The RS1 and RS0, the bits in PSW register, are used to select different memory location (bank0 to bank4) in the RAM memory.



RS1	RS0	Value	
0	0	00h	Bank0
0	1	08h	Bank1
1	0	10h	Bank2
1	1	18h	Bank3

Special Function Registers (SFR)

Special function registers are upper RAM memory in the 8051 microcontroller. These registers contain all peripheral related registers like P0, P1, P2, P3, timers or counters, serial port and interrupts-related registers. The SFR memory address starts from 80h to FFh. The SFR register is implemented by bit-address registers and byte-address registers.



80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

🟦	Blue background are I/O port SFRs
🟨	Yellow background are control SFRs
🟩	Green background are other SFRs

The accumulator, B register, Po, P1, P2, P3, IE registers are bit-addressable register remaining all are byte-addressable registers.

Accumulator

The accumulator which is also known as ACC or A is a bit as well as a byte-addressable register by an address of the accumulator. If you want to use a bit-addressable register, you can use a single bit (E0) of the register and you can use an 8-bit of the accumulator as a byte-addressable register. The accumulator holds the results of most Arithmetic and logical operations.

B-Register

The B-register is a bit and byte-addressable register. You can access 1-bit or all 8-bits by a physical address F0h. Suppose to access a bit 1, we have to use f1. The B register is only used for multiplication and division operations.

Port Registers

The 8051 microcontroller consists of 4-input and output ports (P0, P1, P2, and P3) or 32-I/O pins. Each pin is designed with a transistor and P registers. The pin configuration is very important for a microcontroller that depends on the logic states of the registers. The pin configuration as input given by 1 or output 0 depends on the logic states. If logic 1 is applied to the bit of the P register, the output transistor switches off the appropriate pin that acts as an input pin.

Counters and registers

Many microcontrollers consist of one or more timers and counters. The timers are used to generate precious time delay and the source for the timers is crystal oscillator. The counters are used to count the number of external events – for instance, the objective counter , and the source for counters are external pulses applied across the counter pin.

The 8051 microcontroller consists of two 16-bit timers and counters such as timer 0 and timer 1. Both the timers consist of a 16-bit register in which the lower byte is stored in the TL and the higher byte is stored in the TH. The Timer can be used as a counter as well as for timing operation that depends on the source of the clock pulses to the counters.

The Counters and Timers in 8051 microcontrollers contain two special function registers: TMOD (Timer Mode Register) and TCON (Timer Control Register), which are used for activating and configuring timers and counters.

Types of Shift Register

Shift registers are a type of sequential logic circuits that are mainly used for storage of digital data. The shift registers are bit-addressable registers that store only one bit of data. The shift registers are constructed with flip-flops – a group of flip-flops connected as a chain so that the output from one flip-flop becomes the input of the next flip-flop.

All the flip-flops are driven by the clock signals that are implemented by the D-flip-flap. The shift registers are mainly used for serial communication.

These are classified into 4- types:

- Serial in Serial out (SISO)
- Serial in Parallel out (SIPO)
- Parallel in Serial out (PISO)
- Parallel in Parallel out (PIPO)

Interrupts of 8051

Interrupts are the events that temporarily suspend the main program, pass the control to the external sources and execute their task. It then passes the control to the main program where it had left off.

8051 has 5 interrupt signals, i.e. INT0, TFO, INT1, TF1, RI/TI. Each interrupt can be enabled or disabled by setting bits of the IE register and the whole interrupt system can be disabled by clearing the EA bit of the same register.

Interrupt Enable Register

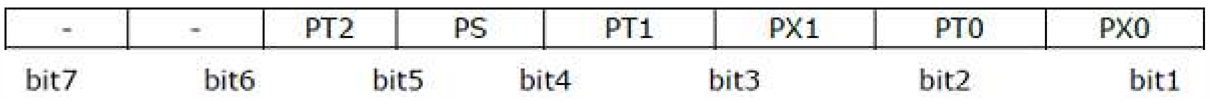
This register is responsible for enabling and disabling the interrupt. EA register is set to one for enabling interrupts and set to 0 for disabling the interrupts. Its bit sequence and their meanings are shown in the following figure.

EA	-	-	ES	ET1	EX1	ET0	EX0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

EA	IE.7	It disables all interrupts. When EA = 0 no interrupt will be acknowledged and EA = 1 enables the interrupt individually.
-	IE.6	Reserved for future use.
-	IE.5	Reserved for future use.
ES	IE.4	Enables/disables serial port interrupt.
ET1	IE.3	Enables/disables timer1 overflow interrupt.
EX1	IE.2	Enables/disables external interrupt1.
ET0	IE.1	Enables/disables timer0 overflow interrupt.
EX0	IE.0	Enables/disables external interrupt0.

Interrupt Priority Register

The priority levels of the interrupts can be changed by changing the corresponding bit in the Interrupt Priority register as shown.



-	-	PT2	PS	PT1	PX1	PT0	PX0
bit7	bit6	bit5	bit4		bit3	bit2	bit1

-	IP.6	Reserved for future use.
-	IP.5	Reserved for future use.
PS	IP.4	It defines the serial port interrupt priority level.
PT1	IP.3	It defines the timer interrupt of 1 priority.
PX1	IP.2	It defines the external interrupt priority level.
PT0	IP.1	It defines the timer0 interrupt priority level.
PX0	IP.0	It defines the external interrupt of 0 priority level.

- A low priority interrupt can only be interrupted by the high priority interrupt, but not interrupted by another low priority interrupt.
- If two interrupts of different priority levels are received simultaneously, the request of higher priority level is served.
- If the requests of the same priority levels are received simultaneously, then the internal polling sequence determines which request is to be serviced.

Serial communication of 8051 microcontroller

When electronic devices communicate with each other, they can transfer data in two different ways. One is serial and other one is parallel. When digital data is transferred serially, it is transmitted bit by bit, whereas in parallel transfer, many bits are transmitted at same time.

Though parallel transfer of data is much faster but requires many wires, while serial transfer is slower as compared to parallel transfer but requires few wires. Serial communication may be synchronous or asynchronous. In synchronous communication, transmitter also transmits a clock along with data. This clock is used for synchronization between transmitter and receiver device. In asynchronous transfer of data, there is no clock.

Serial communication maybe simplex, half-duplex or full duplex. Simplex communication means that data will be transmitted only in one direction while half duplex means data will be transmitted in both directions but at one time, only one device can transmit, whereas full duplex means data may be transmitted in both directions at one time, while one device is transmitting, it can also receive data transmitted from other device at same time.

Transmitter and receiver are configured to communicate at some data transfer rate before communication starts. This data transfer rate or number of bits transmitted per second is called baud rate for handling serial communication.

Parallel communication is fast but it is not applicable for long distances (for printers). Moreover, it is also expensive. Serial is not much fast as parallel communication but it can deal with transmission of data over longer distances (for telephone line, ADC, DAC). It is also cheaper and requires less physical wires, that’s why we use serial communication.

Methods of serial communication

There are two methods of Serial Communication: serial communication types asynchronous and synchronous

Synchronous: Transfer the block of data (characters) between sender and receiver spaced by fixed time interval. This transmission is synchronized by an external clock.

Asynchronous: There is no clock involved here and transmission is synchronized by special signals along the transmission medium. It transfers a single byte at a time between sender and receiver along with inserting a start bit before each data character and a stop bit at its termination so that to inform the receiver where the data begins and ends. An example is the interface between a keyboard and a computer. Keyboard is the transmitter and the computer is the receiver. We use USART and UART for serial communications. USART or UART is a microcontroller peripheral which converts incoming and outgoing bytes of data into a serial bit stream. Both have same work but with different methods.

USART:

USART stands for Universal Synchronous/Asynchronous Receiver-Transmitter. USART uses external clock. So it needs separate line to carry the clock signal. Sending peripheral generates a clock and the receiving peripheral recover from the data stream without knowing the baud rate ahead of time. By use of external clock, USART's data rate can be much higher (up to rates of 4 Mbps) than that of a standard UART.

UART:

It stands for Universal Asynchronous Receiver-Transmitter. A UART generates its internal data clock to the microcontroller. It synchronizes that clock with the data stream by using the start bit transition. The receiver needs the baud rate to know ahead of time to properly receive the data stream.

8051 Serial Communication Programming Registers

8051 microcontroller has a built-in serial port called UART. We can easily read and write values to the serial port. For using serial port we simply need to configure the serial port:

- Operation mode (how many data bits we want)
- Baud rate

There are 21 Special function registers (SFR) in 8051 microcontroller and 21 unique locations are for these 21 SFR. Each of these register is of 1 byte size. The "Serial Control" (SCON) is the SFR which is used to configure serial port.

8051 serial communication SCON register:

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0: Serial port mode

SM1: Serial port mode

SM2: Enable multiprocessor

Enable multiprocessor communication in modes 2 and 3 (for 9 bit UART).

REN: Receiver Enable

Set/clear by software to enable/disable receive operation

TB8: Transmit bit. Set or clear by software. The 9 bits will be transmitted in mode 2 and 3.

TB8 = 1, a value is written to the serial port, 9th bit of data = 1.

TB8 = 0, 9th bit of data = 0, RI will not set.

RB8: Receive bit 8. Set or clear by software. The 9 bits will be received in mode 2 and 3. First eight bits are the data received and 9th bit received will be placed in RB8.

TI: Transmit Interrupt flag. Set by hardware when a byte is transmitted completely. Now the port is free and ready to send the next byte. This bit must be cleared by software.

RI: Receive Interrupt flag. Set by hardware when a byte has been completely received. This lets the program to know that it needs to read the value quickly before another byte is read. This bit must be cleared by software.

Timer /Counter

8051 has two 16-bit programmable UP timers/counters. They can be configured to operate either as timers or as event counters. The names of the two counters are T0 and T1 respectively. The timer content is available in four 8-bit special function registers, viz, TL0,TH0,TL1 and TH1 respectively.

In the "timer" function mode, the counter is incremented in every machine cycle. Thus, one can think of it as counting machine cycles. Hence the clock rate is 1/12 th of the oscillator frequency.

In the "counter" function mode, the register is incremented in response to a 1 to 0 transition at its corresponding external input pin (T0 or T1). It requires 2 machine cycles to detect a high to low transition. Hence maximum count rate is 1/24 th of oscillator frequency.

The operation of the timers/counters is controlled by two special function registers, TMOD and TCON respectively.

Timer Mode control (TMOD) Special Function Register:

TMOD register is not bit addressable.

Gate	$C/\overline{T}$	M1	M0	Gate	$C/\overline{T}$	M1	M0
Timer-1				Timer-0			

Gate:

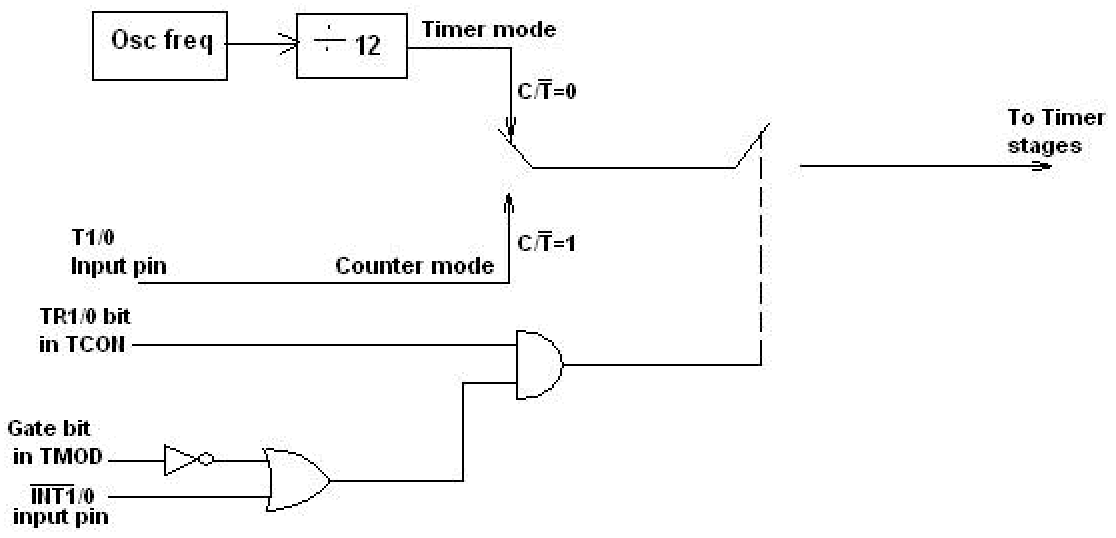
This is an OR Gate enabled bit which controls the effect of $\overline{INT1/0}$ on START/STOP of Timer. It is set to one ('1') by the program to enable the interrupt to start/stop the timer. If TR1/0 in TCON is set and signal on $\overline{INT1/0}$ pin is high then the timer starts counting using either internal clock (timer mode) or external pulses (counter mode).

$C/\overline{T}$:

It is used for the selection of Counter/Timer mode.

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

Timer / Counter Logic:



Timer control (TCON) Special function register:

TCON is bit addressable. The address of TCON is 88H. It is partly related to Timer and partly to interrupt.

TF1	TRI	TF0	TR0	IE1	IT1	IE0	IT0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

---------------- TIMER ----------------------- ------------------ INTERRUPT --------------

TF1: Timer1 overflow flag. It is set when timer rolls from all 1s to 0s. It is cleared when processor vectors to execute ISR located at address 001B.

TR1: Timer1 run control bit. Set to 1 to start the timer / counter.

TF0: Timer0 overflow flag. (Similar to TF1).

TR0: Timer0 run control bit.

IE1: Interrupt1 edge flag. Set by hardware when an external interrupt edge is detected. It is cleared when interrupt is processed.

IE0: Interrupt0 edge flag (Similar to IE1).

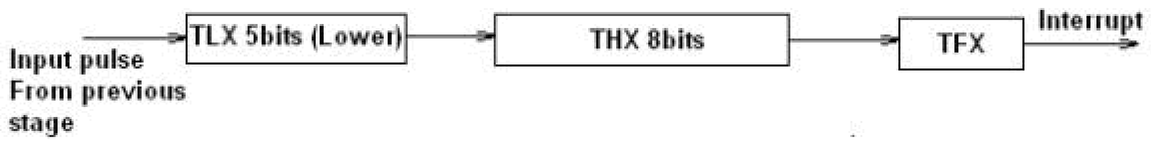
IT1: Interrupt1 type control bit. Set/ cleared by software to specify falling edge / low level triggered external interrupt.

IT0: Interrupt0 type control bit. (Similar to IT1).

Timers operating modes:

Timer mode 0:

In this mode, the timer is used as a 13-bit UP counter.

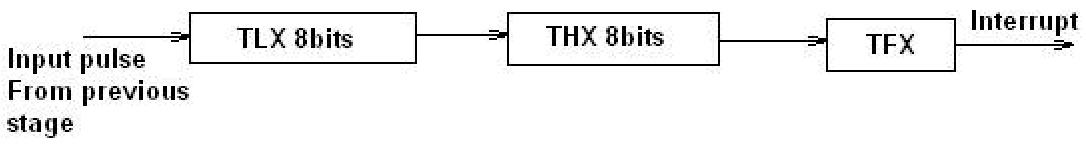


The lower 5 bits of TLX and 8 bits of THX are used for the 13 bit count. Upper 3 bits of TLX are ignored. When the counter rolls over from all 0's to all 1's, TFX flag is set and an interrupt is generated.

The input pulse is obtained from the previous stage. If TR1/0 bit is 1 and Gate bit is 0, the counter continues counting up. If TR1/0 bit is 1 and Gate bit is 1, then the operation of the counter is controlled by   input. This mode is useful to measure the width of a given pulse fed to   input.

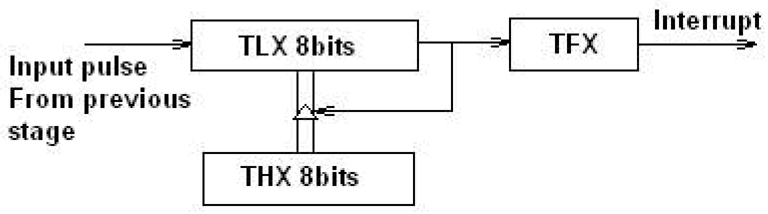
Timer mode 1:

This mode is similar to mode-0 except for the fact that the Timer operates in 16-bit mode.



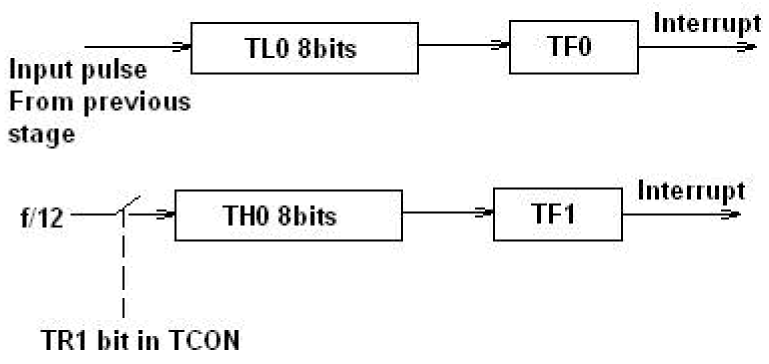
Timer mode 2: (Auto reload mode)

This is a 8 bit counter/timer operation. Counting is performed in TLX while THX stores a constant value. In this mode when the timer overflows (i.e. TLX becomes FF), it is fed with the value stored in THX. For example if we load THX with 50H then the timer in mode 2 will count from 50H to FFH. After that 50H is again reloaded.



Timer mode 3:

Timer 1 in mode-3 simply holds its count. The effect is same as setting TR1=0. Timer0 in mode-3 establishes TL0 and TH0 as two separate counters.



Control bits TR1 and TF1 are used by Timer-0 higher 8 bits (TH0) in Mode-3 while TR0 and TF0 are available to Timer-0 lower 8 bits (TL0).

Review Questions

PART - A

1. What is microcontroller?
2. How many I/O ports are placed in microcontroller 8051?
3. How many SFRs are placed in microcontroller 8051?
4. Define DPTR?
5. Mention the capacity of internal RAM and internal ROM of 8051?
6. What is the purpose of PSW register?
7. What is the purpose of SFRs?
8. What is the use of EA pin?
9. Mentioned the 3 different address spaces of 8051?
10. What is the capacity of internal data memory?
11. What is the capacity of internal RAM and external RAM?
12. How is the internal RAM classified?
13. How many register banks are placed in internal RAM
14. How many bits addressable locations are placed in internal RAM?
15. What is program counter?
16. What is the use of carry flag?
17. Where is the sack memory placed in 8051?
18. Which port is used for alternate input and output functions?
19. What is the use of TMOD register?
20. How many timer/counters are placed in 8051?

21. Which SFR is used for setting the timer?
22. Mention the different types of modes of timer/counter operations?
23. What is interrupt signal?
24. Name the five types of 8051 interrupt signals?
25. Which SFRs are used for interrupts?
26. Which interrupt has highest priority within level?
27. Mention the different modes of serial communication?
28. Mention the baud rate of mode 2?
29. Mention the baud rate of mode 1 ?
30. What are the signals use for accessing external data memory?

PART – B

1. State the advantagous of microcontroller.
2. State the features of microcontroller.
3. State the disadvantages and applications of microcontroller.
4. Explain program counter and data pointer.
5. Compare microprocessor and microcontroller.
6. Draw the pin diagram of 8051 microcontroller.
7. Explain ALU in 8051.
8. Mention the byte addresses of SFRS.
9. Explain external data memory.
10. Explain external program memory.
11. Draw the structure of internal RAM.
12. Explain PSW register.
13. Explain stack and stacks pointer.
14. Explain TMOD register.
15. Explained TON register.
16. Explain IE register.
17. Explain IP register.
18. How are interrupts handled?
19. Mention the destinations of interrupt.
20. Explain external interrupts.
21. State the mode of operation of serial communication and mentions their baud rates.
22. Explain multiprocessor communication.

23. Write short notes on baud rates of serial communication.
24. Draw the connection diagram of external program memory with 8051.

PART – C

1. Draw and explain the block diagram of 8051 microcontroller.
2. List the special function registers with their byte and bit addresses and explain any one of them.
3. Draw and explain the structure of internal RAM.
4. Draw the pin diagram of microcontroller 8051and explain each bin.
5. Briefly explain the memory organisation of 8051 microcontroller.
6. Explain external data memory and external program memory.
7. With the diagram explain internal data memory.
8. Explain PSW register and SMOD register.
9. Explain stack and stack pointer with example.
10. With the diagram explain the I/O ports of microcontroller 8051.
11. Explain TMOD and TON registers.
12. With the diagram explain mode 0 timer/counter operation.
13. Explain IP and IE register.
14. Explain the priority levels of 8051.
15. Explain external interrupts.
16. Explain timer interrupts.
17. Explain serial port interrupts.
18. Explain SCON and TCON register.
19. Explain the different modes of serial communication.
20. With the explain how external program memory is interfaced with 8051.

UNIT – IV

8051 INSTRUCTION SET AND PROGRAMMING

8051 Instruction set

8051 has about 111 instructions. These can be grouped into the following categories

1. Arithmetic Instructions
2. Logical Instructions
3. Data Transfer instructions
4. Boolean Variable Instructions
5. Program Branching Instructions

The following nomenclatures for register, data, address and variables are used while write instructions.

A: Accumulator

B: "B" register

C: Carry bit

Rn: Register R0 - R7 of the currently selected register bank

Direct: 8-bit internal direct address for data. The data could be in lower 128bytes of RAM (00 - 7FH) or it could be in the special function register (80 - FFH).

@Ri: 8-bit external or internal RAM address available in register R0 or R1. This is used for indirect addressing mode.

#data8: Immediate 8-bit data available in the instruction.

#data16: Immediate 16-bit data available in the instruction.

Addr11: 11-bit destination address for short absolute jump. Used by instructions AJMP & ACALL.

Addr16: 16-bit destination address for long call or long jump.

Rel: 2's complement 8-bit offset (one - byte) used for short jump (SJMP) and all conditional jumps.

bit: Directly addressed bit in internal RAM or SFR

Arithmetic Instructions

Mnemonics	Description	Bytes
ADD A, Rn	A $\longleftarrow$ A + Rn	1

ADD A, direct	A ⟵ A + (direct)	2
ADD A, @Ri	A ⟵ A + @Ri	1
ADD A, #data	A ⟵ A + data	2
ADDC A, Rn	A ⟵ A + Rn + C	1
ADDC A, direct	A ⟵ A + (direct) + C	2
ADDC A, @Ri	A ⟵ A + @Ri + C	1
ADDC A, #data	A ⟵ A + data + C	2
DA A	Decimal adjust accumulator	1
DIV AB	Divide A by B A ⟵ quotient B ⟵ remainder	1
DEC A	A ⟵ A -1	1
DEC Rn	Rn ⟵ Rn - 1	1
DEC direct	(direct) ⟵ (direct) - 1	2
DEC @Ri	@Ri ⟵ @Ri - 1	1
INC A	A ⟵ A+1	1
INC Rn	Rn ⟵ Rn + 1	1
INC direct	(direct) ⟵ (direct) + 1	2
INC @Ri	@Ri ⟵ @Ri +1	1
INC DPTR	DPTR ⟵ DPTR +1	1
MUL AB	Multiply A by B A ⟵ low byte (A*B) B ⟵ high byte (A* B)	1
SUBB A, Rn	A ⟵ A - Rn - C	1
SUBB A, direct	A ⟵ A - (direct) - C	2
SUBB A, @Ri	A ⟵ A - @Ri - C	1
SUBB A, #data	A ⟵ A - data - C	2

Logical Instructions

Mnemonics	Description	Bytes
ANL A, Rn	A ⟵ A AND Rn	1
ANL A, direct	A ⟵ A AND (direct)	2
ANL A, @Ri	A ⟵ A AND @Ri	1
ANL A, #data	A ⟵ A AND data	2
ANL direct, A	(direct) ⟵ (direct) AND A	2
ANL direct, #data	(direct) ⟵ (direct) AND data	3
CLR A	A⟵ 00H	1
CPL A	A⟵A	1
ORL A, Rn	A ⟵ A OR Rn	1
ORL A, direct	A ⟵ A OR (direct)	1
ORL A, @Ri	A ⟵ A OR @Ri	2
ORL A, #data	A ⟵ A OR data	1
ORL direct, A	(direct) ⟵ (direct) OR A	2

ORL direct, #data	(direct) ⟵ (direct) OR data	3
RL A	Rotate accumulator left	1
RLC A	Rotate accumulator left through carry	1
RR A	Rotate accumulator right	1
RRC A	Rotate accumulator right through carry	1
SWAP A	Swap nibbles within Acumulator	1
XRL A, Rn	A ⟵ A EXOR Rn	1
XRL A, direct	A ⟵ A EXOR (direct)	1
XRL A, @Ri	A ⟵ A EXOR @Ri	2
XRL A, #data	A ⟵ A EXOR data	1
XRL direct, A	(direct) ⟵ (direct) EXOR A	2
XRL direct, #data	(direct) ⟵ (direct) EXOR data	3

Data Transfer Instructions

Mnemonics	Description	Bytes
MOV A, Rn	A ⟵ Rn	1
MOV A, direct	A ⟵ (direct)	2
MOV A, @Ri	A ⟵ @Ri	1
MOV A, #data	A ⟵ data	2
MOV Rn, A	Rn ⟵ A	1
MOV Rn, direct	Rn ⟵ (direct)	2
MOV Rn, #data	Rn ⟵ data	2
MOV direct, A	(direct) ⟵ A	2
MOV direct, Rn	(direct) ⟵ Rn	2
MOV direct1, direct2	(direct1) ⟵ (direct2)	3
MOV direct, @Ri	(direct)⟵ @Ri	2
MOV direct, #data	(direct) ⟵ #data	3
MOV @Ri, A	@Ri ⟵ A	1
MOV @Ri, direct	@Ri ⟵ (direct)	2
MOV @Ri, #data	@Ri ⟵ data	2
MOV DPTR, #data16	DPTR ⟵ data16	3
MOVC A, @A+DPTR	A ⟵ Code byte pointed by A + DPTR	1
MOVC A, @A+PC	A ⟵ Code byte pointed by A + PC	1
MOVC A, @Ri	A ⟵ Code byte pointed by Ri 8-bit address)	1
MOVX A, @DPTR	A ⟵ External data pointed by DPTR	1
MOVX @Ri, A	@Ri ⟵ A (External data - 8bit address)	1

MOVX @DPTR, A	@DPTR $\longleftarrow$ A(External data - 16bit address)	1
PUSH direct	(SP) $\longleftarrow$ (direct)	2
POP direct	(direct) $\longleftarrow$ (SP)	2
XCH Rn	Exchange A with Rn	1
XCH direct	Exchange A with direct byte	2
XCH @Ri	Exchange A with indirect RAM	1
XCHD A, @Ri	Exchange least significant nibble of A with that of indirect RAM	1

Boolean Variable Instructions

Mnemonics	Description	Bytes
CLR C	C-bit $\longleftarrow$ 0	1
CLR bit	bit $\longleftarrow$ 0	2
SET C	C $\longleftarrow$ 1	1
SET bit	bit $\longleftarrow$ 1	2
CPL C	C $\longleftarrow$ $\overline{\text{C-bit}}$	1
CPL bit	bit $\longleftarrow$ $\overline{\text{bit}}$	2
ANL C, /bit	C $\longleftarrow$ C . $\overline{\text{bit}}$	2
ANL C, bit	C $\longleftarrow$ C. bit	2
ORL C, /bit	C $\longleftarrow$ C + $\overline{\text{bit}}$	2
ORL C, bit	C $\longleftarrow$ C + bit	2
MOV C, bit	C $\longleftarrow$ bit	2
MOV bit, C	bit $\longleftarrow$ C	2

Program Branching Instructions

Mnemonics	Description	Bytes
ACALL addr11	PC + 2 $\longrightarrow$ (SP) ; addr 11 $\longrightarrow$ PC	2
AJMP addr11	Addr11 $\longrightarrow$ PC	2
CJNE A, direct, rel	Compare with A, jump (PC + rel) if not equal	3
CJNE A, #data, rel	Compare with A, jump (PC + rel) if not equal	3
CJNE Rn, #data, rel	Compare with Rn, jump (PC + rel) if not equal	3
CJNE @Ri, #data, rel	Compare with @Ri A, jump (PC + rel) if not equal	3

DJNZ Rn, rel	Decrement Rn, jump if not zero	2
DJNZ direct, rel	Decrement (direct), jump if not zero	3
JC rel	Jump (PC + rel) if C bit = 1	2
JNC rel	Jump (PC + rel) if C bit = 0	2
JB bit, rel	Jump (PC + rel) if bit = 1	3
JNB bit, rel	Jump (PC + rel) if bit = 0	3
JBC bit, rel	Jump (PC + rel) if bit = 1	3
JMP @A+DPTR	A+DPTR $\longrightarrow$ PC	1
JZ rel	If A=0, jump to PC + rel	2
JNZ rel	If A $\neq$ 0 , jump to PC + rel	2
LCALL addr16	PC + 3 $\longrightarrow$ (SP), addr16 $\longrightarrow$ PC	3
LJMP addr 16	Addr16 $\longrightarrow$ PC	3
NOP	No operation	1
RET	(SP) $\longrightarrow$ PC	1
RETI	(SP) $\longrightarrow$ PC, Enable Interrupt	1
SJMP rel	PC + 2 + rel $\longrightarrow$ PC	2
JMP @A+DPTR	A+DPTR $\longrightarrow$ PC	1
JZ rel	If A = 0. jump PC+ rel	2
JNZ rel	If A $\neq$ 0, jump PC + rel	2
NOP	No operation	1

Programs

1. **Addition of two eight bit numbers; Sum 8 bit**

```
            MOV A, # DATA1
            ADD A, # DATA2
            MOV 50H,A
HERE    SJMP    HERE
```

2.

```
           MOV A, Direct Address1
           ADD A,  Direct Address 2
           MOV 53H,A
HERE   SJMP    HERE
```

3. **Addition of two eight bit numbers; Sum 16 bit**

```
            MOV A, # DATA1
            ADD A, # DATA2
            MOV 50H, A
            CLR A
            RLC A
            MOV 51H, A
HERE    SJMP    HERE
```

4. Subtraction of two eight bit numbers

```
            CLR C
            MOV A, # DATA1
            SUBB A, # DATA2
            MOV 50H,A
HERE    SJMP    HERE
```

5. Addition of two 16 bit numbers; Sum 32 bit (With carry)

```
            MOV A, Direct Address1
            ADD A,  Direct Address 2
            MOV 54H, A
            MOV A, Direct Address3
            ADDC A,  Direct Address 4
            MOV 55H, A
            CLR A
            RLC A
            MOV 56H, A
HERE    SJMP    HERE
```

6. Sum of series of numbers

```
            MOV R3, #00
            MOV R2, 50H

            MOV R0, #51H
            MOV A, @R0
            DEC R2
LOOP    INC R0
            ADD A, @R0
            JNC    NEXT
            INC R3
NEXT    DEC R2
            JNZ   LOOP
            MOV 61H, A
            CLR A
            RLC A
            MOV 62H, A
HERE    SJMP    HERE
```

Review Questions

PART - A

1. What are the different types of MOVC instructions?
2. What is the difference between MOVX A, @Ri and MOVXA, @DPTR instructions?
3. What are the different types of logical operations performed in 8051?
4. What will happen during the execution of SWAP A instruction?
5. What is the use of compare instructions used in 8051?
6. Mention the different types of compare instructions used in 8051?
7. Mention the different types of rotate instructions used in 8051?
8. Mention the different types of unconditional jump instructions used in 8051?
9. What is the basic difference between jump and call instructions?
10. What is the basic difference between AJMP and LJMP instructions?
11. Why do the call instructions use stack memory?
12. What is the basic difference between RET and RETI instructions?
13. What is the use of time delay routine?

PART – B

1. Explain direct addressing with example.
2. Explain register addressing with example.
3. Explain exchange instructions.
4. Explain general purpose data transfer instructions.
5. Explain MUL AB instruction.
6. Explain ADD and ADDC instructions.
7. Explain the various types of rotate instructions used in in 8051.
8. Explain the various types of compare instructions used in 8051.
9. Explain the various types of DJNZ instructions used in 8051.
10. What are the differences between call and jump instructions used in 8051.
11. Differentiate ACALL and LCALL instructions

PART – C

1. Explain the data transfer instructions used in 8051.
2. Explain the arithmetic instructions used in 8051.
3. Explain the logical instructions used in 8051.
4. Explain the branch control instructions used in 8051.

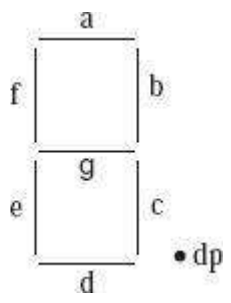
5. Explain bit manipulation instructions used in 8051.
6. Explain MOVX and MOVC instructions with examples.
7. Explain ADD and ADDC instructions used in 8051.
8. Explain MUL A B and DIV A B instructions used in 8051 with examples.
9. Explain the signed addition and subtraction functions performed in 8051 with example.
10. Explain rotate instructions used in 8051 with diagrams.
11. Write a program to add two 16 bit numbers using 8051.
12. Write a program to sum a series of numbers using 8051.

UNIT – V

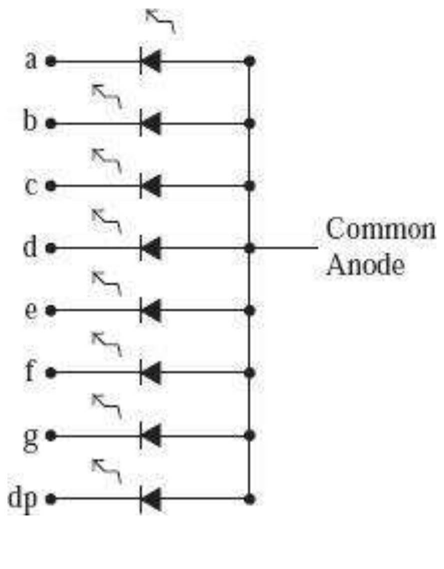
APPLICATIONS

Seven segment LED display

An output device which is very common is seven segment display. Moreover, there are eight segments in a LED display consisting of 7 segments which includes '.', consisting of character 8 and having a decimal point just next to it. The segments are 'a, b, c, d, e, f, g, and dp' where dp signifies '.' which is the decimal point. Moreover, these are LEDs or together a series of Light Emitting Diodes. A 7-segment display is as shown in the following figure

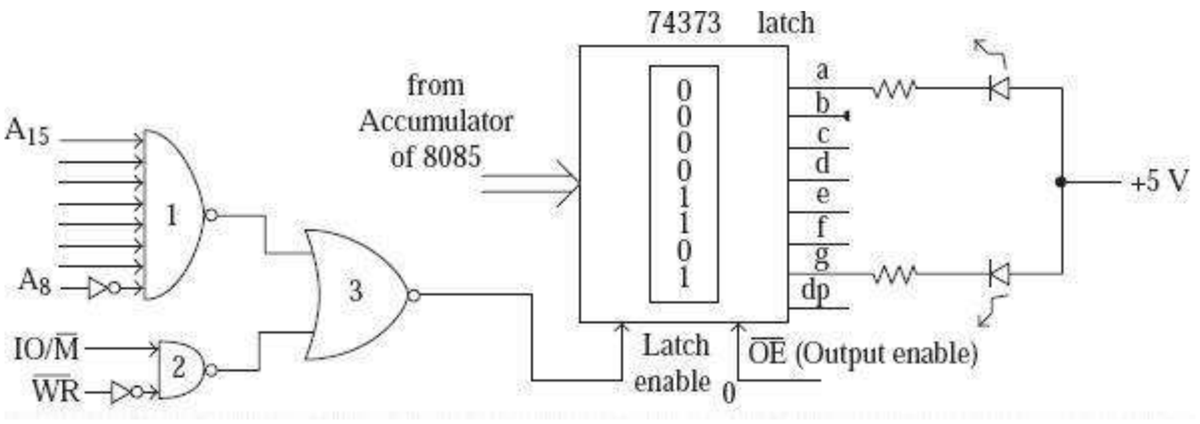


The internal circuit comprising of a display of seven segment is as shown in figure



There are two types of 7-segment LED: They are the common anode type and the common cathode type. We have discussed the common anode-type which is 7 segmented Light Emitting Diode. In the LED which is common anode and is 7-segmented, here we connect all the eight LED anodes together and the eight external pin is brought to display. And this pin gets connected to a DC supply of +5 Volt. The cathode ends of the eight segments are brought out on the pins of the display.

The use of 74373 latch for interfacing a 7-segment display is shown in the following figure



74373 latch is used as an I/O mapped I/O port with the port address as FEH. The program to display 0 to 9 is as follows:

Memory Addresss	**Lable**	**Mnemonics**
2000		MVI A, 98H
2002		OUT 03
2004	ABOVE	LXI H, 2500
2007		MOV E, M
2008	LOOP	INX H
2009		MOV A, M
200A		OUT 01
200C		MVI B, 0F
200E	BEHIND	MVI C, FF
2010	BACK	MVI D, FF
2012	GO	DCR D
2013		JNZ GO
2016		DCR C
2017		JNZ BACK
201A		DCR B
201B		JNZ BEHIND
201E		DCR E
201F		JNZ LOOP
2022		JMP ABOVE

DATA
2500 – 0A 2501 - 00 2502 – 01 2503 – 03 2504 – 04 2505 – 05 2506 – 06 2507 – 07
2508 – 08 2509 - 09

Temperature measurement

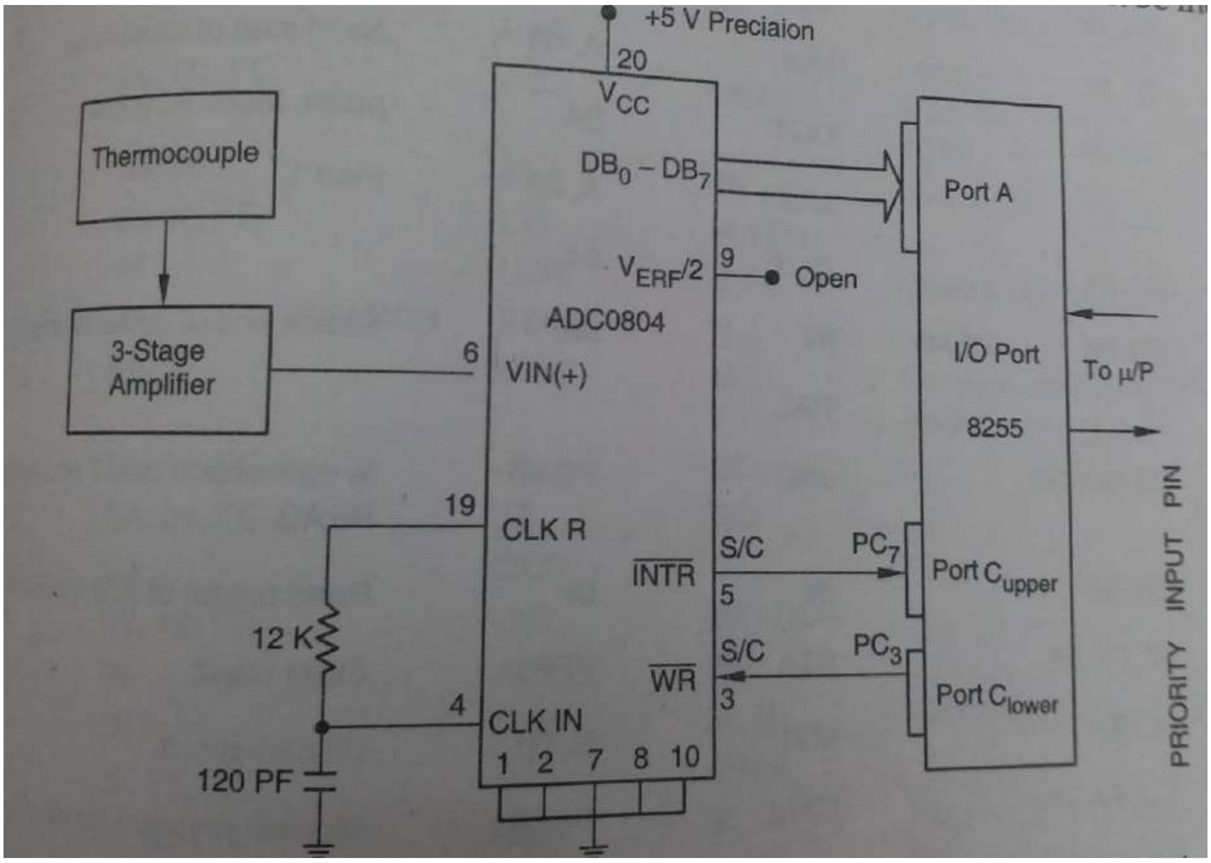


Figure shows a microprocessor based scheme for temperature measurement and control. The output of a Thermocouple proportional to the temperature of the Furnace or oven etc., is in millivolt. It is amplified using multistage amplifier before it is processed by microprocessor. The amplified voltage is applied to A/D converter.

The microprocessor sensor start of conversion signal to the A/D converter through the port of 8255 PPI. When A/D converter complete conversion it sends and end of conversion signal to the microprocessor.

Having received an end of conversion signal from A/D converter, the microprocessor reads the output of the A/D converter which is a digital quantity proportional to the temperature to be measured.

The microprocessor displayed the measured temperature. If the temperature of a furnace oven or water bath is to be controlled the microprocessor first measures its temperature and then compass the measured temperature with the reference temperature at which the temperature is to be maintained. If the measured temperature is higher than the reference temperature, microprocessor sends control signal to reduce temperature.

If the measured temperature is less than the reference temperature the microprocessor sends a control signal to increase temperature. The temperature of a furnace or over can be increased or decreased by increasing or decreasing the fuel input to the Furnace. If heating is done by electric heaters current in heating element is controlled. DC level detector is for initial adjustment.

Memory Addresss	**Lable**	**Mnemonics**
2000		MVI A, 98H
2002		OUT 0B
2004	LOOP	MVI A, 00
2006		OUT 0A
2007		MVI A, 08
2009		OUT 0A
200A	READ	IN 0A
200D		RAL
2010		JNC REAB
2013		IN 08
2014		STA 2050
2016		MVI B, 00
2017		CALL 06FA
201A		JUM LOOP

Readings will depend upon the transducer and interfacing circuits. If temperature is to be displayed in degree Celsius, a lookup table can be used.

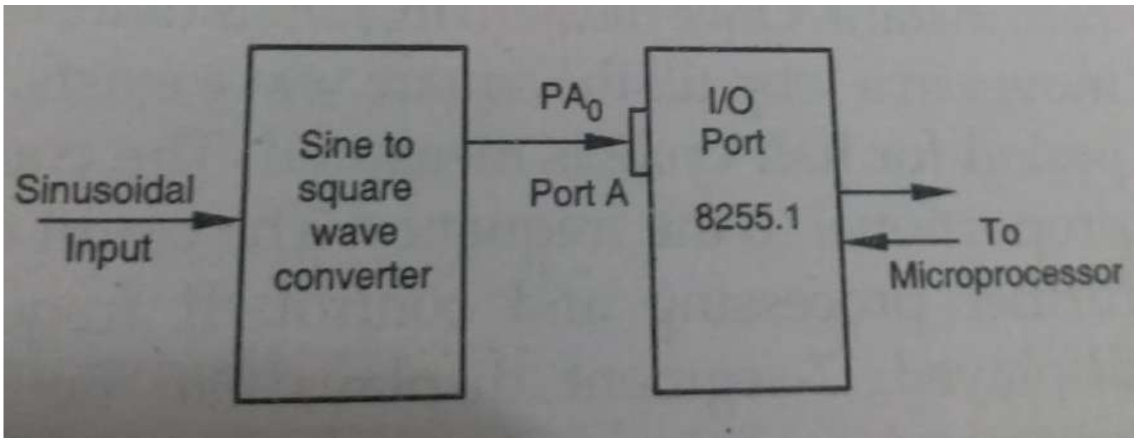
Frequency measurement

To measure the frequency of a signal the time period for half cycle is measured which is inversely proportional to the frequency. Sinusoidal signal is converted to square wave using a voltage comparator or operational amplifier. A diode is used to rectify the output signal. A potential divider is used to reduce the magnitude to five volts.

A program has been developed to sense the zero instant of the rectified square wave. The microprocessor measures the magnitude of the square wave at two consecutive points. The two magnitudes are compared and decision is taken on the basis of carry and zero status flags whether the point is at zero instant.

As soon as the zero instant point it is detected the microprocessor initiates a register pair to count the number how many times the loop is executed. The microprocessor reads the magnitude of the square wave again and again and moves in the loop. It crosses the loop when the magnitude of the square wave becomes zero. Thus the time for half cycle is measured.

The count can be compared with the stored numbers in a lookup table and the frequency can be displayed. The count which is inversely proportional to the frequency of the input signal can be used for further processing and control as desired.
An interfacing circuitry is shown. The port A is input. Control word is 98H.



Program

Memory Addresss	**Lable**	**Mnemonics**
2000		MVI A, 98H
2002		OUT 03
2004	BACK	IN 00
2006		MOV B,A
2007		IN 00
2009		CMP B
200A		JZ BACK
200D		JC BACK
2010		LXI B, 00 00
2013	LOOP	INX B
2014		IN 00
2016		RAR
2017		JC LOOP
201A		HLT

Measurement of voltage and current

An A/D converter is used for the measurement of voltage. The voltage is applied to the analog input terminal of A/D converter. The output of an ADC is a digital quantity equivalent to the input voltage. The output terminals of the A/D converter are connected to the I/O port.

The microprocessor reads the I/O port and the voltage in digital form is transferred to the accumulator. ADC gives an output of 80H for zero volt input and FF for 5 volt Input.

An unknown input voltage is converted to digital quantity by the A/D converter. The microcomputer reads this digital voltage. The digital voltage obtained can be calibrated either by computation or using lookup table.

This can be displayed in decimal numbers. ADC gives an output 00 for 0 volt input and FF for 5 volt input. On this basis any voltage measured in the digital form can be calibrated and may be displayed.

For the measurement of current, the current signal is converted to voltage signal. The voltage signal is applied to A/D converter.

Memory Addresss	**Lable**	**Mnemonics**
2000		MVI A, 98H
2002		OUT 03
2004		MVI A, 9A
2006		OUT 0B
2008	START	IN 09
200A		MOV B, A
200B		IN 09
200D		CMP B
200E		JZ START
2011		JC START
2014		MVI A, 03
2016		OUT 02
2018		MVI A, 01
201A		OUT 0A
201C		MVI D, 03
201E	BACK	DCR D
201F		JNZ BACK
2022		MVI A, 00
2024		OUT 0A
2026		MVI A, 0B
2028		OUT 02
202A		MVI A, 03
202C		OUT 02
202E	READ	IN 02
2030		RAL
2031		JNC READ
2034		IN 00
2036		CMA
2037		SUI 80H
2039		STA 2450
203C		MVI A, 06
203E		OUT 02
2040		MVI A, 01
2042		OUT 0A
2044		MVI D, 03
2046	GO	DCR D
2047		JNZ GO
204A		MVI A, 00
204C		OUT 0A
204E		MVI A, 0B
2050		OUT 02
2052		MVI A, 03
2054		OUT 02
2056	SEE	IN 02

```
2058        RAL
2059        JNC   SEE
205C         IN      00
205E         CMA
205F         SUI    80H
2061        STA    2502
2064        CALL  2200
2067        CALL  2400
206A        HLT
```

Traffic Light Controller

The 8085 Microprocessor is a popular Microprocessor used in Industries for various applications. Such as traffic light control, temperature control, stepper motor control, etc. The traffic lights are interfaced to Microprocessor system through buffer and ports of programmable peripheral Interface 8255, so the traffic lights can be automatically switched ON/OFF in desired sequence.

The hardware of the system consists of two parts. The first part is Microprocessor based system with 8085. Microprocessor as CPU and the peripheral devices like EPROM, RAM, Keyboard & Display Controller 8279, Programmable as Peripheral Interface 8255, 26 pin parallel port connector, 21 keys Hexa key pad and six number of seven segment LED's.

The second part is the traffic light controller interface board, which consist of 36 LED's in which 20 LED's are used for vehicle traffic and they are connected to 20 port lines of 8255 through Buffer. Remaining LED's are used for pedestrian traffic. The traffic light interface board is connected to Main board using 26 core flat cables to 26-pin Port connector. The LED's can be switched ON/OFF in the specified sequence by the Microprocessor.

The normal function of traffic lights requires sophisticated control and coordination to ensure that traffic moves as smoothly and safely as possible and that pedestrians are protected when they cross the roads. A variety of different control systems are used to accomplish this, ranging from simple clockwork mechanisms to sophisticated computerized control and coordination systems that self-adjust to minimize delay to people using the road.

Traffic Controller Systems

A traffic signal is typically controlled by a controller inside a cabinet mounted on a concrete pad. Although some electro-mechanical controllers are still in use (New York City still has 4,800), modern traffic controllers are solid state. The cabinet typically contains a power panel, to distribute electrical power in the cabinet; a detector interface panel, to connect to loop detectors and other detectors; detector amplifiers; the controller itself; a conflict monitor unit; flash transfer relays; a police panel, to allow the police to disable the signal; and other components.

The following program controls light of one square. By changing the delay between two signals one can change the speed of traffic. 8255 Port Address are Port A- 00H Port B - 01H Port C- 02H; Control Word 03H

Address	**Lable**	**Mnemonics**
2000		MVI A,80H
2002		OUT 03H
2004		MVI A,11H
2006		OUT 00H
2008		OUT 02H
200A		CALL DELAY1
200D	LOOP	MVI A, 44H
200F		OUT 00H
2011		CALL DELAY1
2014		MVI A,22H
2016		OUT 00H
2018		CALL DELAY2
201B		MVI A, 99H
201D		OUT 00H
201F		CALL DELAY1
2022		MVI A,22H
2024		OUT 00H
2026		CALL DELAY2
2029		MVI A,11H
202B		OUT 00H
202D		MVI A,44H
202F		OUT 02H
2031		CALL DELAY1
2034		MVI A,22H
2036		OUT 02H
2038		CALL DELAY2
203B		MVI A, 99H
203D		OUT 02H
203F		CALL DELAY1
2042		MVI A, 22H
2044		OUT 02H
2046		CALL DELAY2
2049		MVI A, 11H
204B		OUT 02H
204D		JMP LOOP
2050	DELAY1:	MVI B, 25H
2052	LP3	MVI C, 0FFH
2054	LP2	MVI D, 0FFH
2056	LP1	DCR D
2057		JNZ LP1
205A		DCR C
205B		JNZ LP2
205E		DCR B
205F		JNZ LP3

```
2062                          RET
2063    DELAY2                MVI    B, 05H
2065            LP6           MVI    C, 0FFH
2067            LP5           MVI    D, 0FFH
2069            LP4           DCR    D
206A                          JNZ    LP4
206D                          DCR    C
206E                          JNZ    LP5
2071                          DCR    B
2072                          JNZ    LP6
2075                          RET
```

Microcontroller Interfacing And Applications

Stepper Motor Interface

Stepper motors are used to translate electrical pulses into mechanical movements. In some disk drives, dot matrix printers, and some other different places the stepper motors are used. The main advantage of using the stepper motor is the position control. Stepper motors generally have a permanent magnet shaft (rotor), and it is surrounded by a stator.

Normal motor shafts can move freely but the stepper motor shafts move in fixed repeatable increments.

Some parameters of stepper motors −

- Step Angle − The step angle is the angle in which the rotor moves when one pulse is applied as an input of the stator. This parameter is used to determine the positioning of a stepper motor.
- Steps per Revolution − This is the number of step angles required for a complete revolution. So the formula is 360° /Step Angle.
- Steps per Second − This parameter is used to measure a number of steps covered in each second.
- RPM − The RPM is the Revolution Per Minute. It measures the frequency of rotation. By this parameter, we can measure the number of rotations in one minute.

The relation between RPM, steps per revolution, and steps per second is like below:

Steps per Second = rpm x steps per revolution / 60

Interfacing Stepper Motor with 8051 Microcontroller

The Unipolar stepper motor works in three modes.

- **Wave Drive Mode** − In this mode, one coil is energized at a time. So all four coils are energized one after another. This mode produces less torque than full step drive mode.

The following table is showing the sequence of input states in different windings.

Steps	Winding A	Winding B	Winding C	Winding D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

- **Full Drive Mode** − In this mode, two coils are energized at the same time. This mode produces more torque. Here the power consumption is also high

The following table is showing the sequence of input states in different windings.

Steps	Winding A	Winding B	Winding C	Winding D
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

- **Half Drive Mode** − In this mode, one and two coils are energized alternately. At first, one coil is energized then two coils are energized. This is basically a combination of wave and full drive mode. It increases the angular rotation of the motor

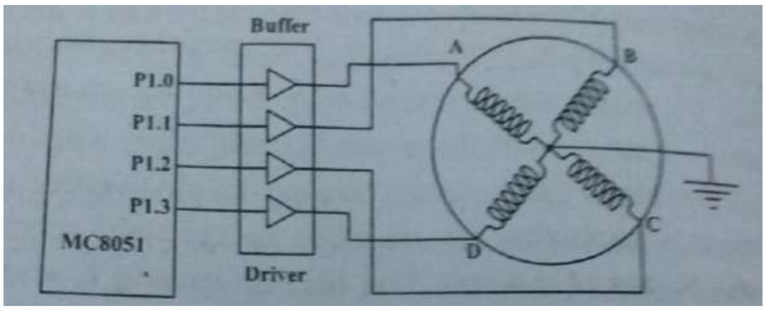
The following table is showing the sequence of input states in different windings.

Steps	A	B	C	D
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

The excitation sequence of a stepper motor is shown below

Step no.	Winding A	Winding B	Winding C	Winding D	Hexa code	Direction	
1	1	0	0	1	09	Forward $\downarrow$	Reverse $\uparrow$
2	0	0	1	1	03		
3	0	1	1	0	06		
4	1	1	0	0	0C		

The connection diagram of stepper motor with microcontroller 8051 is shown below:



Program for moving a stepper motor in forward direction

```
          ORG     4000H
          MOV  A, 99H
REPEAT    MOV  P1, A
          LCALL  DELAY
          RLA
          SJMP  REPEAT
          END
```

DELAY SUBROUTINE

```
DELAY      MOV  R0 , # FF
FIRST      MOV   R1 , # FF
NEXT      DJNZ   R1 ,  NEXT
          DJNZ   R0 , FIRST
          RET
```

For reverse direction, the instruction RRA is used for RLA.

Key board interfacing

The key board considered for interfacing is a matrix keyboard. This key board is designed with a particular rows and columns. These rows and columns are connected to the microcontroller through its ports of the micro controller 8051. We normally use 8*8 matrix key board. So only two ports of 8051 can be easily connected to the rows and columns of the key board.

When ever a key is pressed, a row and a column gets shorted through that pressed key and all the other keys are left open. When a key is pressed only a bit in the port goes high. Which indicates microcontroller that the key is pressed. By this high on the bit key in the corresponding column is identified.

Once we are sure that one of key in the key board is pressed next our aim is to identify that key. To do this we firstly check for particular row and then we check the corresponding column the key board.

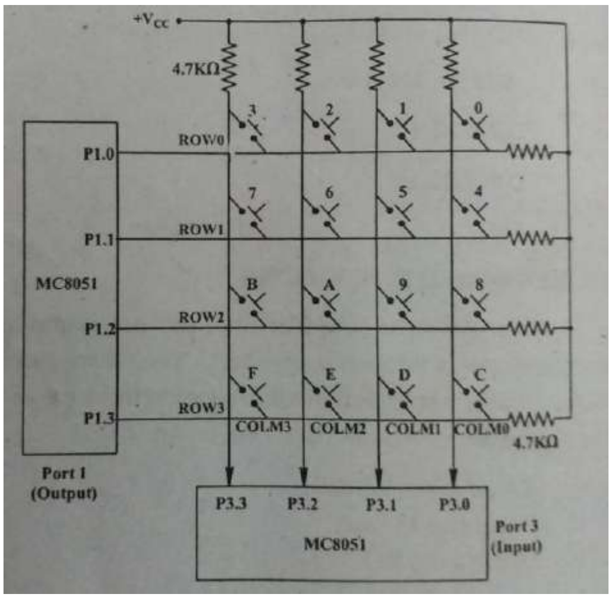
To check the row of the pressed key in the keyboard, one of the row is made high by making one of bit in the output port of 8051 high . This is done until the row is found out. Once we get the row next out job is to find out the column of the pressed key. The column is detected by contents in the input ports with the help of a counter. The content of the input port is rotated with carry until the carry bit is set.

The contents of the counter is then compared and displayed in the display. This display is designed using a seven segment display and a BCD to seven segment decoder IC 7447.

The BCD equivalent number of counter is sent through output part of 8051 displays the number of pressed key. The programming algorithm, the circuit diagram and program is as follows.

1. The 8051 has 4 I/O ports P0 to P3 each with 8 I/O pins, P0.0 to P0.7,P1.0 to P1.7, P2.0 to P2.7, P3.0 to P3.7. The one of the port P1 (it understood that P1 means P1.0 to P1.7) as an I/P port for microcontroller 8051, port P0 as an O/P port of microcontroller 8051 and port P2 is used for displaying the number of pressed key.
2. Make all rows of port P0 high so that it gives high signal when key is pressed.
3. See if any key is pressed by scanning the port P1 by checking all columns for non zero condition.
4. If any key is pressed, to identify which key is pressed make one row high at a time.
5. Initiate a counter to hold the count so that each key is counted.
6. Check port P1 for nonzero condition. If any nonzero number is there in [accumulator], start column scanning by following step 9.
7. Otherwise make next row high in port P1.
8. Add a count of 08h to the counter to move to the next row by repeating steps from step 6.
9. If any key pressed is found, the [accumulator] content is rotated right through the carry until carry bit sets, while doing this increment the count in the counter till carry is found.
10. Move the content in the counter to display in data field or to memory location
11. To repeat the procedures go to step 2.

Circuit diagram:



Program:

```
         START:   MOV A, #00H
                  MOV P1, A
                  MOV A, #0FH
                  MOV P1, A
         PRESS:   MOV A, P2
                  JZ PRESS
                  MOV A, #01H
                  MOV R4, A
                  MOV R3, #00H
          NEXT:   MOV A, R4
                  MOV P1, A
                  MOV A, P2
                  JNZ COLSCAN
                  MOV A, R4
                  RL A
                  MOV R4, A
                  MOV A, R3
                  ADD A, #08H
                  MOV R3, A
                  SJMP NEXT
     COLSCAN      MOV R5, #00H
            IN:   RRC A
                  JC OUT
                  INC R3
                  JMP IN
           OUT:   MOV A, R3
                  DA A
                  MOV P2, A
                  JMP START
```

Review Questions

PART - A

1. What is is interfacing?
2. What is the use of 8255?
3. Mention the ports placed in 8255.
4. Mention the ports placed in group A and group B of 8255.
5. State the modes of operation of 8255.
6. Define ADC.
7. Which technique is used in ADC 0808.
8. Mention the three major task of keyboard to get meaningful data.
9. What is a seven segment display?
10. What are common cathode type and common anode type seven segment display?

11. Mention the different types of seven segment LED.
12. What is the use of stepper motor?

PART – B

1. Draw the functional block diagram of 8255.
2. Draw the control word format for I/O mode of 8255.
3. Explain the input configuration of 8255 in mode 1.
4. Explain the output configuration of 8255 in mode 1.
5. Explain mode 2 of 8255.
6. Draw the interfacing diagram of 8255 with 8051.
7. Draw the interfacing diagram of matrix keyboard with Microcontroller 8051.
8. Draw the interfacing diagram of 4 digit seven segment LED display with Microcontroller 8051.
9. Draw the interfacing diagram of stepper motor with 8051.
10. Discuss the applications of seven segment display.

PART – C

1. Draw the functional diagram of 8255 and explain each block.
2. Explain the control word format for 8255.
3. With the diagram, explain how 8255 is interfaced with 8051.
4. Draw the interfacing diagram of 4 digit 7 segment LED display with Microcontroller 8051 and explain its operation.
5. Discuss a microprocessor based scheme to measure and display frequency.
6. Show interface connections to measure and display current and voltages at several points of a circuit employing a microprocessor based scheme.
7. Show interface connections to measure and control temperature of a furnace employing a microprocessor based scheme.
8. Show interface connections for a microprocessor based scheme for traffic control.
9. Draw the interfacing diagram of stepper motor with 8051 and explain its operation.
10. Draw the interfacing diagram of matrix keyboard with Microcontroller 8051 and explain its operation.

Books for Study & Reference:

1. **Fundamentals of Microprocessors and Microcontrollers by B. RAM**
   DHANPAT RAI Publications (P) Ltd., New Delhi
2. **Microprocessor and microcontroller by M. Parasuram**
   N. V. Publications, Pollachi.