MICROCONTROLLER AND PLC LAB-436

SEMESTER-5

Exp:1

STUDY OF MICROCONTROLLER 8051

Aim:-

To study the microcontroller and familiarize the 8051microcontroller kit

Theory:-

A Microcontroller consists of a powerful CPU tightly coupled with memory (RAM, ROM or EPROM), various I/O features such as Serial port(s), Parallel port(s), Timer/Counter(s), Interrupt Controller, Data Acquisition Interfaces - Analog to Digital Converter(ADC), Digital to Analog Converter(DAC), everything integrated onto a single Silicon chip.

It does not mean that any microcontroller should have all the above said features onchip. Depending on the need and area of application for which it is designed, the onchip features present in it may or may not include all the individual sections said above. Any microcomputer system requires memory to store a sequence of instructions making up a program, parallel port or serial port for communicating with an external system, timer/counter for control purposes like generating time delays, baud rate for the serial port, apart from the controlling unit called the Central Processing Unit. If a system is developed with a microprocessor, the designer has to go for external memory such as RAM, ROM or EPROM and peripherals and hence the size of the PCB (Printed Circuit Board) will be large enough to hold all the required peripherals. But, the microcontroller has got all these peripheral facilities on a single chip. So, development of a similar system with a microcontroller reduces PCB size and cost of the design. One of the major differences between a microcontroller and a microprocessor is that a controller often deals with bits, not bytes as in the real world applications. For example, switch contacts can only be open or close, indicators should be lit or dark and motors can be either turned on or off and so forth.

The Major Features of 8051 Are:

* 8-bit CPU optimised for control applications.
* Extensive Boolean processing (single-bit logic) capabilities.
* 64K Program Memory address space.
* 64K Data Memory address space.
* 4K bytes of on-chip Program Memory (in 8051 and 8751 only).
* 128 bytes of on-chip Data RAM.
* 32 bi-directional and individually addressable I/O lines.
* Two 16-bit Timer / Counters.
* Full Duplex UART (Universal Asynchronous Receiver / Transmitter).
* 6-source / 5-vector interrupt structure with two priority levels.
* On-chip Oscillator and Clock circuitry.

THE 8051 MICROCONTROLLER ARCHITECTURE

The architecture of the 8051 family of microcontrollers is referred to as the MCS-51 architecture (Micro Controller Series-51), or sometimes simply as MCS-51.The microcontrollers have an 8-bit data bus. They are capable of addressing 64K of program memory and a separate 64K of data memory. The block diagram of 8051 microcontroller is shown in fig.

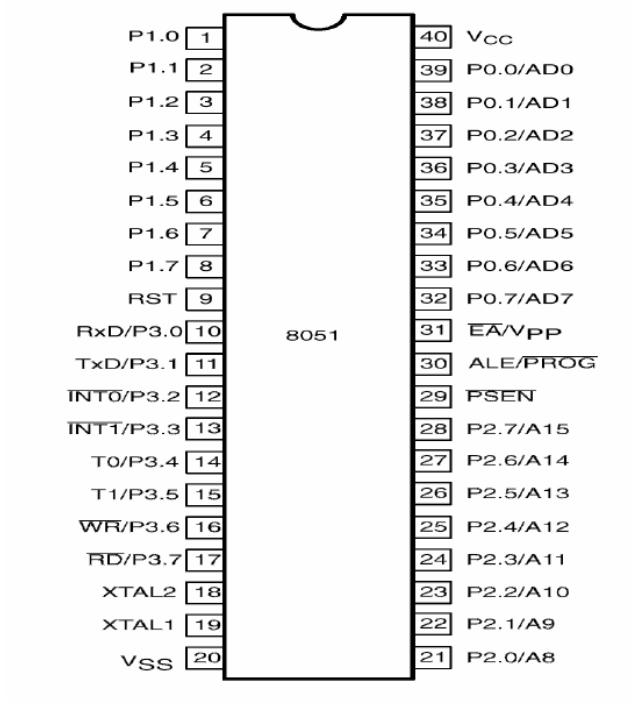
The 8051 have 4K of code memory implemented as on-chip Read Only Memory (ROM). The 8051 have 128 bytes of internal Random Access Memory (RAM). The 8051 has two timer/counters, a serial port, 4 general purpose parallel input/output ports, and interrupt control logic with five sources of interrupts. Besides internal RAM, the 8051

have various Special Function Registers (SFR), which are the control and data registers for on-chip facilities. The SFRs also include the accumulator, the B register, and the Program Status Word (PSW), which contains the CPU flags. Programming the various internal hardware facilities of the 8051 is achieved by placing the appropriate control words into the corresponding SFRs. The 8031 are similar to the 8051, except it lacks the on-chip ROM.

PIN DESCRIPTION OF THE 8051

1–8: Port 1: Each of these pins can be used as either input or output according to your needs. Also, pins 1 and 2 (P1.0 and P1.1) have special functions associated with Timer 2.

9: Reset Signal: High logical state on this input halts the MCU and clears all the registers. Bringing this pin back to logical state zero starts the program a new as if the power had just been turned on. In another words, positive voltage impulse on this pin resets the MCU. Depending on the device's purpose and environs, this pin is usually connected to the push-button, reset-upon-start circuit or a brown out reset circuit.



10-17: Port 3:as with Port 1, each of these pins can be used as universal input or output. However, each pin of Port 3 has an alternative function:

a. Pin 10: **RXD** - serial input for asynchronous communication or serial output for synchronous communication.
b. Pin 11: **TXD** - serial output for asynchronous communication or clock output for synchronous communication
c. Pin 12: **INT0*** - input for interrupt 0
d. Pin 13: **INT1*** - input for interrupt 1
e. Pin 14: **T0** - clock input of counter 0
f. Pin 15: **T1** - clock input of counter 1
g. Pin 16: **WR*** - signal for writing to external (add-on) RAM memory.
h. Pin 17: **RD*** - signal for reading from external RAM memory

18-19: X2 and X1: Input and output of internal oscillator. Quartz crystal controlling the frequency commonly connects to these pins. Capacitances within the oscillator mechanism optimal voltage

20: GND: Ground

21- 28: Port 2: if external memory is not present, pins of Port 2 act as universal input/output. If external memory is present, this is the location of the higher address byte, i.e. addresses A8 – A15. It is important to note that in cases when not all the 8 bits are used for addressing the memory (i.e. memory is smaller than 64kB), the rest of the unused bits are not available as input/output.

29: PSEN*: MCU activates this bit (brings to low state) upon each reading of byte instruction) from program memory. If external ROM is used for storing the program, PSEN- is directly connected to its control pins.
30: ALE: before each reading of the external memory, MCU sends the lower byte of the address register (addresses A0 – A7) to port P0 and activates the output ALE. External Chip (eg: 74HC373), memorizes the state of port P0 upon receiving a signal from ALE pin, and uses it as part of the address for memory chip. During the second part of the MCU cycle, signal on ALE is off, and port P0 is used as *Data Bus*. In this
way, by adding only one integrated circuit, data from port can be multiplexed and the port simultaneously used for transferring both addresses and data.
31: EA*: Bringing this pin to the logical state zero designates the ports P2 and P3 for transferring addresses regardless of the presence of the internal memory. This means that even if there is a program loaded in the MCU it will not be executed, but the one from the external ROM will be used instead. Conversely, bringing the pin to the high logical state causes the controller to use both memories, first the internal, and then the external (if present).
32-39: Port 0: Similar to Port 2, pins of Port 0 can be used as universal input/output, if external memory is not used. If external memory is used, P0 behaves as address output (A0 – A7) when ALE pin is at high logical level, or as data output (Data Bus) when ALE pin is at low logical level.
40: VCC: Power +5V

Result:-

Studied and familiarized the microcontroller 8051

Exp:-2

FIND SUM OF TWO 8-BIT NUMBERS

Aim:-

Write a program to find sum of two 8- bit numbers stored at 8200h, 8201h and store the result at 8202h

Program:-

Memory address	Hex code	Label	Mnemonics	Comments
	90		MOV DPTR,#address	Load DPTR with external address ……
	……..			
	……..			
	E0		MOVX A,@DPTR	Load the value from external memory to Accumulator
	F9		MOV R1,A	Copy the value from accumulator to register R1 (R1=A)
	90		MOV DPTR,# address	Load DPTR with external address ……
	….			
	….			
	ED		MOVX A,@DPTR	Load the value from external memory to Accumulator
	29		ADD A,R1	A=A+R1
	90		MOV DPTR,# address	Load DPTR with external address ……
	……			
	……			
	F0		MOVX @DPTR,A	Copy the result to external memory
	80		SJMP, $	Here the program ends
	FE			

Result:-

Executed the program and verified the result

Exp:-3

FIND DIFFERENCE OF TWO 8-BIT NUMBERS

Aim:-

Write a program to find Difference of two 8- bit numbers stored at 8200h, 8201h and store the result at 8202h

Program:-

Memory address	Hex code	Label	Mnemonics	Comments
	90		MOV DPTR,#address	Load DPTR with external address ……
	……..			
	……..			
	F9		MOV R1,A	Copy the value from accumulator to register R1 (R1=A)
	A3		INC DPTR	Increment DPTR (DPTR=DPTR+1)
	E0		MOVX A,@DPTR	Load the value from external memory to Accumulator
	29		SUBB A,R1	A=A-R1
	A3		INC DPTR	Increment DPTR (DPTR=DPTR+1)
	F0		MOVX @DPTR,A	Copy the result to external memory
	80		SJMP, $	Here the program ends
	FE			

Result:-

Executed the program and verified the result

Exp:-4

FIND MULTIPLICATION OF TWO 8-BIT NUMBERS

Aim:-

Write a program to find Multiplication of two 8- bit numbers stored at 8200h, 8201h and store the result at 8202h

Program:-

Memory address	Hex code	Label	Mnemonics	Comments
	90		MOV DPTR,#address	Load DPTR with external address ……
	……..			
	……..			
	E0		MOVX A,@DPTR	Load the value from external memory to Accumulator
	F5		MOV B,A	Copy the value from accumulator to B register
	ED		MOVX A,@DPTR	Load the value from external memory to Accumulator
	29		MUL AB	A=AxB
	A3		INC DPTR	Increment DPTR (DPTR=DPTR+1)
	F0		MOVX @DPTR,A	Copy the result to external memory
	80		SJMP, $	Here the program ends
	FE			

Result:-

Executed the program and verified the result

Exp:-5

FIND DIVISION OF TWO 8-BIT NUMBERS

Aim:-

Write a program to find Division of two 8- bit numbers stored at 8200h, 8201h and store the result at 8202h

Program:-

Memory address	Hex code	Label	Mnemonics	Comments
	90		MOV DPTR,#address	Load DPTR with external address ……
	……..			
	……..			
	E0		MOVX A,@DPTR	Load the value from external memory to Accumulator
	F5		MOV B,A	Copy the value from accumulator to B register
	ED		MOVX A,@DPTR	Load the value from external memory to Accumulator
	29		DIV AB	A=A/B
	A3		INC DPTR	Increment DPTR (DPTR=DPTR+1)
	F0		MOVX @DPTR,A	Copy the result to external memory
	80		SJMP, $	Here the program ends
	FE			

Result:-

Executed the program and verified the result

Exp:-6

FIND SUM OF N NUMBERS

Aim:-

Write a program to find sum of N numbers

Program:-

Memory address	Hex code	Label	Mnemonics	Comments
	90		MOV DPTR,#address	Load DPTR with external address ……
	……..			
	……..			
	E0		MOVX A,@DPTR	Load the value from external memory to Accumulator
	FE		MOV R6,A	Copy the value from accumulator to register R6 (R6=A)
	FD		MOV R5,A	Copy the value from accumulator to register R5 (R5=A)
	79		MOV R1,#30	Load register R1 with internal memory location 30
	30			
	A3	Rpt.1	INC DPTR	Increment DPTR (DPTR=DPTR+1)
	E0		MOVX A,@DPTR	Load the value from external memory to Accumulator
	F1		MOV @R1,A	Store the value from Accumulator to internal memory location
	09		INC R1	Increment R1 (R1=R1+1)
	DD		DJNZ R5,rpt.1	Decrement the register R5 then jump to rpt if non zero (R5=R5-1)
	FA			
	1E		DEC R6	Decrement R6 (R6=R6+1)
	79		MOV R1,#30	Load register R1 with internal memory location 30
	30			
	E7		MOV A,@R1	Load the content of R1 to A
	09	Rpt.2	INC R1	Increment R1 (R1=R1+1)
	27		ADD A,@R1	Add the values
	DE		DJNZ R6,rpt.2	Decrement the register R6 then jump to rpt if non zero (R6=R6-1)
	FC			
	90		MOV DPTR,# address	Load DPTR with external address ……
	….			
	….			
	F0		MOVX @DPTR,A	Copy the result to external memory
	80		SJMP, $	Here the program ends
	FE			

Result:-

Executed the program and verified the result

Exp:-7

DATA TRANSFER

Aim:-

Write a program to transfer a block of data from one memory location to another memory location

Program:-

Memory address	Hex code	Label	Mnemonics	Comments
	90		MOV DPTR,# address	Load DPTR with external address ……
	……..			
	……..			
	E0		MOVX A,@DPTR	Load the value from external memory to Accumulator
	FE		MOV R6,A	Copy the value from accumulator to register R6 (R6=A)
	FD		MOV R5,A	Copy the value from accumulator to register R5 (R5=A)
	79		MOV R1,#30	Load register R1 with internal memory location 30
	30			
	A3	Rpt.1	INC DPTR	Increment DPTR (DPTR=DPTR+1)
	E0		MOVX A,@DPTR	Load the value from external memory to Accumulator
	F1		MOV @R1,A	Store the value from Accumulator to internal memory location
	09		INC R1	Increment R1 (R1=R1+1)
	DD		DJNZ R5,rpt.1	Decrement the register R5 then jump to rpt if non zero (R5=R5-1)
	FA			
	90		MOV DPTR,# address	Load DPTR with external address ……
	…..			
	…..			
	79		MOV R1,#30	Load register R1 with internal memory location 30
	30			
	E7	Rpt.2	MOV A,@R1	Load the content of R1 to A
	F0		MOVX @DPTR,A	Copy the values to external memory
	09		INC R1	Increment R1 (R1=R1+1)
	A3		INC DPTR	Increment DPTR (DPTR=DPTR+1)
	DE		DJNZ R6,rpt.2	Decrement the register R6 then jump to rpt if non zero (R6=R6-1)
	FA			
	80		SJMP, $	Here the program ends
	FE			

Result:-

Executed the program and verified the result

Exp:-8

LARGEST FROM N NUMBRS

Aim:

Write a program to find the largest number of an array and display the result in memory location after the last byte of data.

Program:-

Memory address	Hex code	Label	Mnemonics	Comments
	90		MOV DPTR,# address	Load DPTR with external address ……
	……..			
	……..			
	E0		MOVX A,@DPTR	Load the value from external memory to Accumulator
	FE		MOV R6,A	Copy the value from accumulator to register R6 (R6=A)
	FD		MOV R5,A	Copy the value from accumulator to register R5 (R5=A)
	79		MOV R1,#30	Load register R1 with internal memory location 30
	30			
	A3	Rpt.1	INC DPTR	Increment DPTR (DPTR=DPTR+1)
	E0		MOVX A,@DPTR	Load the value from external memory to Accumulator
	F1		MOV @R1,A	Store the value from Accumulator to internal memory location
	09		INC R1	Increment R1 (R1=R1+1)
	DD		DJNZ R5,rpt.1	Decrement the register R5 then jump to rpt if non zero (R5=R5-1)
	FA			
	79		MOV R1,#30	Load register R1 with internal memory location 30
	30			
	1E		DEC R6	Decrement R6 (R6=R6-1)
	E7		MOV A,@R1	Copy the value to accumulator
	09	Rpt.2	INC R1	Increment R1
	FF		MOV R7,A	Copy the value from accumulator to R7
	96		SUBB R7,@R1	Subtracting with borrow
	EF		MOV A,R7	Copy the value from R7 to accumulator
	50		JNC skip (*address)*	Jump to *address* if no carry
	…….			
	E6		MOV A,@R1	Move data from R1 to accumulator
	DE	Skip	DJNZ R6,rpt.2(*address*)	Decrement R6 and jump to *address* if non zero
	……..			
	A3		INC DPTR	Increment DPTR
	F0		MOVX @DPTR,A	Load data with external memory through data pointer
	80		SJMP, $	Here the program ends
	FE			

Result:-

Executed the program and verified the result