Ahsanullah University of Science and Technology (AUST)

Department of Computer Science and Engineering

LABORATORY MANUAL

Course No.: CSE 3118

Course Title: Microprocessors and Microcontrollers Lab

For the students of 3rd Year, 1st Semester of

B.Sc. in Computer Science and Engineering program

Table of Contents

COURSE OUTCOMES

After the successful completion of this course, students are expected to be able to:

Sl. No.	COs	POs	Bloom's Taxonomy		
			C	A	P
1	Imitate and modify some programs and techniques to achieve a clear concept of the 8086 microprocessor and I/O elements connected to it	1			1
2	Design of Microcontroller based embedded systems using microcontroller which takes input from sensors and outputs the information using equipment.	3			2
3	Apply modern design tools, open-source hardware and software platforms (Proteus, Keil IDE, Emu, Arduino development board, Arduino IDE, PCB design software) for assessing how various sensors and external peripherals work with microcontrollers.	5			3
4	Build a combination of s/w and h/w system as a project in the multidisciplinary context for sustainable social and economic development to enhance the quality of life in Bangladesh and around the globe.	7			4
5	Practice safety norms, anti-littering behavior, maximizing energy efficiency and minimizing environmental impact during the design and development of computer chips, systems and software.	6			4
6	Present the project to internal and external project examiners utilizing a multimedia system, and produce a comprehensive report.	10			4
7	Participate actively in all project development phases and communicate effectively as a team member.	9			4
8	Estimate the initial budget for required equipment, maintain the estimated budget, and make final budget after project submission.	11			3

PREFERRED TOOLS

Arduino IDE, Proteus 8 Professional (or higher versions), TinkerCad, MDA-8086 Kit.

TEXT/REFERENCE BOOKS

a) Microprocessors and Microcomputer-Based System Design by Mohamed Rafiquzzaman.
b) The 8051 Microcontroller and Embedded systems: using Assembly and C by Muhammad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. Mckinla.
c) The AVR Microcontroller and Embedded Systems Using Assembly and C: Using Arduino Uno and Atmel Studio by Sepehr Naimi, Sarmad Naimi, Muhammad Ali Mazidi (2nd ed.)
d) Microprocessor and Interfacing Programming and Hardware by Douglas V. Hall

ADMINISTRATIVE POLICY OF THE LABORATORY

- Students must perform class assessment tasks individually without help of others.
- Plagiarism is entirely prohibited and will be dealt with strictly.

Familiarization with the MDA-8086 Microprocessor Trainer and EMU8086 Microprocessor Emulator

Session Objective:

- Understand the component of 8086 trainer board.
- Understand the EMU8086 Microprocessor Emulator.
- Learn 8086 16-bit Intel Microprocessor, its register and assembly level programming.
- Learn assembly programming by practicing simple programs including average calculation of 3/4/5/more numbers, calculate area of a rectangle and a triangle, temperature conversion from °C to °F, conversion from °F to °C, conversion from °C to °K, conversion from °K to °C and counting tiles problems.

Experiment 1: Understand the component of the 8086 trainer board.

MDA-8086 Kit Diagram

Figure 1.1: MDA-8086 Kit Diagram

Figure 1.2: MDA-8086 Kit Circuit Diagram

The function of IC's at MDA-8086 System Configuration

1. CPU (Central processing unit): Using Intel 8086, using 14.7456MHz.
2. ROM (Read Only Memory): It has program to control user's key input, LCD display, user's program. 64K Byte, it has data communication program. Range of ROM Address is F0000H~FFFFFH.
3. SRAM (Static Random-Access Memory): Input user's program & data. Address of memory is 00000H~0FFFFH, totally 64K Byte.
4. DISPLAY: Text LCD Module, 16(Characters)×2(Lines)
5. KEYBOARD: It is used to input machine language. There are 16 hexadecimal keys and 8 function keys.
6. SPEAKER: Sound test.
7. RS-232C: Serial communication with IBM compatible PC.
8. DOT MATRIX LED: To understand & test the dot matrix structure and principle of display. It is interfaced to 8255A(PPI).
9. A/D CONVERTER: ADC0804 to convert the analog signal to a digital signal.
10. D/A CONVERTER: DAC0800 (8-bit D/A converter) to convert the digital signal to the analog signal and to control the level meter.
11. STEPPING MOTOR INTERFACE: Stepping motor driver circuit is designed.
12. POWER: AC 110~220V, DC +5V 3A, +12V 1A, -12V 0.5A SMPS.

MDA-8086 Address Map

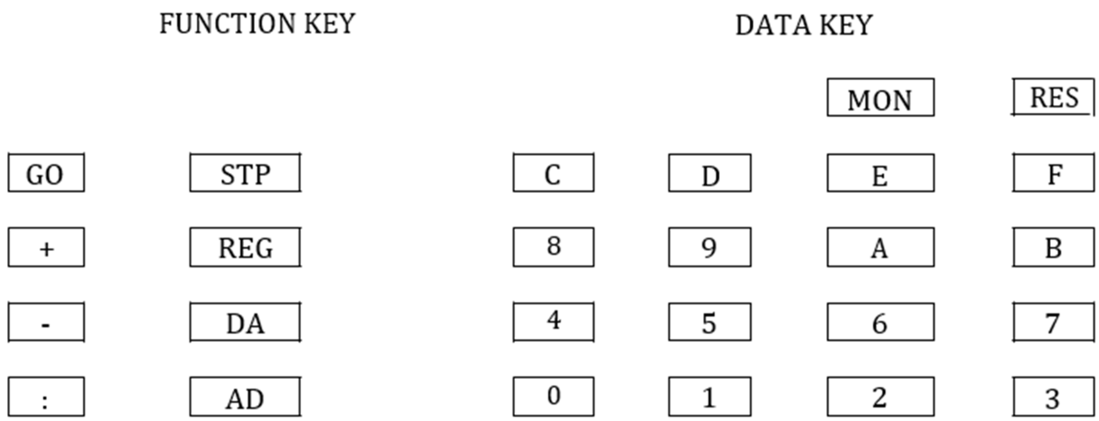
1. Memory Map

ADDRESS	MEMORY	DESCRIPTION
00000H~0FFFFH	RAM	PROGRAM & DATA MEMORY
F0000H~FFFFFH	ROM	MONITOR ROM
10000H~EFFFFH	USER'S RANGE	

ADDRESS	I/O PORT	DESCRIPTION
00H~07H	LCD & KEYBOARD	LCD Display 00H: INSTRUCTION REGISTER 02H: STATUS REGISTER 04H: DATA REGISTER KEYBOARD 01H: KEYBOARD REGISTER (Only read) 01H: KEYBOARD FLAG (Only write)
08H~0FH	8251/8253	8251(Using to data communication) 08H: DATA REGISTER 0AH: INSTRUCTION/STATUS REGISTER 8253 (TIMER/COUNTER) 09H: TIMER 0 REGISTER 0BH: TIMER 1 REGISTER 0DH: TIMER 2 REGISTER 0FH: CONTROL REGISTER
10H~17H	8259/SPEAKER	8259(Interrupt controller) 10H: COMMAND REGISTER 12H: DATA REGISTER SPEAKER 11H: SPEAKER
18H~1FH	8255A-CS1/ 8255A-CS2	8255A-CS1(DOT & ADC INTERFACE) 18H: A PORT DATA REGISTER 1AH: B PORT DATA REGISTER 1CH: C PORT CONTROL REGISTER 8255-CS2(LED & STEPPING MOTOR) 19H: A PORT DATA REGISTER 1BH: B PORT DATA REGISTER 1DH: C PORT CONTROL REGISTER 1FH: CONTROL REGISTER
20H~2FH	I/O EXTEND CONNECTOR	
30H~FFH	USER'S RANGE	

Operation Introduction

MDA-8086 has high performance 64K-byte monitor program. It is designed for easy function. After power is on, the monitor program begins to work. In addition to all the key function the monitor has a memory checking routine.



- RES→ System reset
- STP→ Execute user's program, a single step
- AD→ Set memory address
- GO→ Go to user's program or execute monitor functions
- DA→ Update segment & Offset and input data to memory
- MON→ Immediately break user's program and Non maskable interrupt.
- : → Offset set
- REG→ Register Display.
- +→ Segment & Offset +1 increment. Register display increment.
- -→ Segment & Offset -1 increment. Register display decrement.

8255 Programmable Peripheral Interface Controller

- It has 24-bit input/output pins
- It consists of three ports: port A, port B and port C- all of which are 8 bits
- It also consists of an 8-bit control register (CR)
- The eight bit of port C can be used as individual bits or be grouped in two 4-bit ports: C_{upper}(CU) and C_{lower}(CL)

- The functions of these ports are defined by writing a control word in the control register

Group A	Group B
Port A	Port B
Port C (Upper 4 bit)	Port C (Lower 4 bit)

8086 Instruction Set Summary

Data Registers	AX (Accumulator Register)	AH	AL
	BX (Base Register)	BH	BL
	CX (Count Register)	CH	CL
	DX (Data Register)	DH	DL
Segment Registers	CS (Code Segment)		
	DS (Data Segment)		
	SS (Stack Segment)		
	ES (Extra Segment)		
Index Registers	SI (Source Index)		
	DI (Destination Index)		
Pointer Registers	SP (Stack Pointer)		
	BP (Base Pointer)		
	IP (Instruction Pointer)		
	FLAGS Registers		

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Bit	**Name**	**Symbol**	
0	Carry Flag	CF	**Status Flags**
2	Parity Flag	PF	
4	Auxiliary Carry Flag	AF	
6	Zero Flag	ZF	
7	Sign Flag	SF	
11	Overflow Flag	OF	
8	Trap Flag	TF	**Control Flags**
9	Interrupt Flag	IF	
10	Direction Flag	DF	

Data Transfer Instructions

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCHG
Input	IN
Output	OUT
Push	PUSH
Pop	POP

Arithmetic

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Negate	NEG

Logical and Bit Manipulation

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Disable interrupt	DI

Shift and Rotate

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

Program Control Instructions

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (Subtract)	CMP
Test (AND)	TST

Experiment 2: Understand the EMU8086 Microprocessor Emulator.

Emulation Kit

Emu8086 is a software emulation of Intel's 8086 microprocessor, and I/O Emulation Kit is a software emulation of a group of hardware devices that can be controlled by Emu8086 virtual central processing unit (CPU).

Available hardware devices in I/O Emulation Kit include: Dot Matrix Display, Seven Segment Display, ASCII LCD Display, Group of LEDs, Push Buttons Input, Keyboard Input, Switches Input, Thermometer Input and Pressure Gauge Input.

Download the I/O Emulation Kit with Help Files and Source Code (Version 1.75b) from the below link:
https://sites.google.com/site/hawawebsite/more/emulation-kit

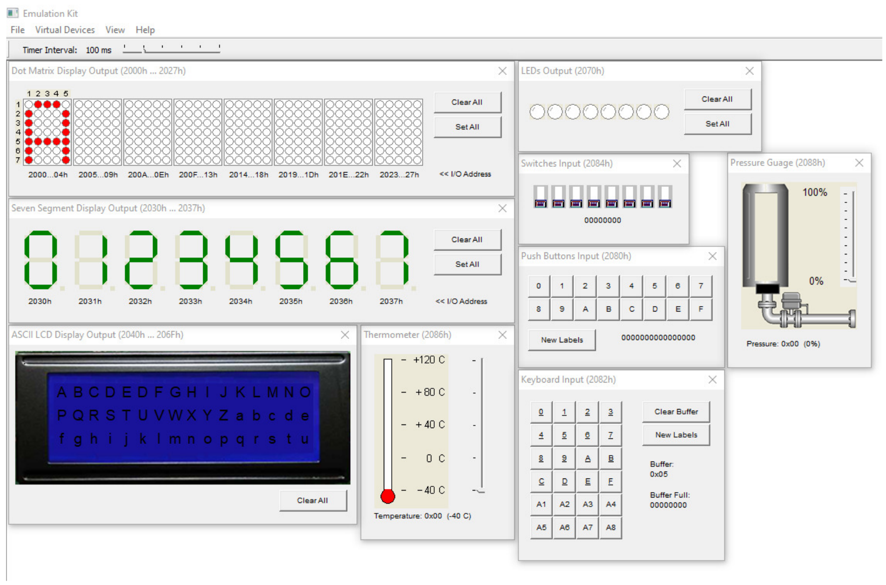


Figure 1.3: EMU8086 Microprocessor Emulator

You can use the following video link to install the software:
https://www.youtube.com/watch?v=nA5GAshhe18

Table 1. Available Devices

Device	I/O Addresses	Number of Addresses	Register Type
Dot Matrix Display Output	2000h … 2027h	40	8 bit
Seven Segment Display Output	2030h … 2037h	8	8 bit
ASCII LCD Display Output	2040h … 206Fh	48	8 bit
LEDs Output	2070h	1	8 bit
Push Buttons Input	2080h	1	16 bit
Keyboard Input	2082h … 2083h	2	8 bit
Switches Input	2084h	1	8 bit
Thermometer Input	2086h	1	8 bit
Pressure Gauge Input	2088h	1	8 bit

Experiment 3: Learn 8086 16-bit Intel Microprocessor, its register and assembly level programming.

Instructions:

RES System Reset

AD Set Memory address

DA Update segment && offset.

STP Execute user's program, a single step.

$\boxed{\text{GO}}$Go to user's program or execute monitor functions.

$\boxed{\text{MON}}$ Immediately break user's program and Non makeable interrupt.

$\boxed{\text{REG}}$ Register display

$\boxed{+}$ Segment & offset +1 increment. Register display increment.

$\boxed{-}$ Segment & offset -1 increment. Register display increment.

From the emulator you will get the **HEX** code for your assembly language program. For execute code and get the result first of all press **RES** button then press **DA** button, now type your **HEX** code here go to next address pressing **+** button. After completing the code typing you have to press **STP** button for executing the code. Now for watch result press **REG** button then you can see the result in the display of MDA-Win8086.

Instruction set:

MOV: This instruction allows copying the value of one register into another register.

ADD: This instruction adds two numbers.

SUB: This instruction subtracts a number from another number.

MUL: This instruction multiplies two numbers.

DIV: The instruction divides a number by another number.

AX= It is called accumulator register.

BX= It is called base register.

CX= It is called count register.

DX=It is called data register

Experiment 4: Learn assembly programming by practicing simple programs including average calculation of 3/4/5/more numbers, calculate area of a rectangle and a triangle, temperature conversion from °C to °F, conversion from °F to °C, conversion from °C to °K, conversion from °K to °C and counting tiles problems.

Problem 1: Temperature conversion from °C to °K let, temperature = 39°C 1°K=1°C + 273

MOV AX, 39
MOV BX, 273
ADD AX, BX
INT 3

Output:	AX=0138 BX=0111 CX=0000 DX=0000
Result Verification:	K= 39+273= 312

Discussion: We know, °K= °C + 273
At first, 27 loaded in AX register and the address is 0404 and 111 replaced in BX register and its address is 0407. Now, AX, BX are added in address 041A. After pressing STP and REG, it shows the result.

INT 3: INT 3 is a special one-byte instruction having op-code is CCH. that is inserted by debuggers at the instruction where the user has set a breakpoint to occur. When it's hit, the interrupt handler breaks into the debugger and then replaces the original instruction so that execution can proceed when the user is ready.

Merge Problem 1 and Problem 2 and show the students about the task done by the INT3

Problem 2: Temperature conversion from °K to °C
let, temperature = 270°K

```
MOV   AX, 270
MOV   BX, 273
SUB   AX, BX
INT   3
```

Output:	AX=FFFD BX=0111 CX=0000 DX=0000
Result Verification:	C= 270-273= -3 = FFFDH

Discussion: We know, °C = °K – 273
At first, 10E replaced in Ax register and the address is 0404 and 111 replaced in Bx register and its address is 0407. Now, Ax, Bx are subtract in address040A.After pressing STP and REG, it shows the result.

Problem 3: Average of 3 numbers: (2+3+5)/3

```
MOV    AX, 2
MOV    BX, 3
ADD    AX, BX
```

MOV BX, 5
ADD AX, BX
MOV BX, 3
DIV BL
INT 3	

Output:	AX=0103 BX=0003 CX=0000 DX=0000
Result Verification:	Avg= (2+3+5)/3= 3 AL = 3, AH = 1
Discussion: At first, load 2 in AX register and the address is 0404 and load 3 in BX register and its address is 0407. Now, AX, BX are added in 040A then load 5 in BX in 040C, and AX, BX are added again in 040F. Now, load 3 in BX and the address is 0411. Then BL is divided in 0413 address. After pressing STP and REG, it shows the result.	

Problem 4: Average of 5 numbers: ((2+3+4+1+5)/5)	

MOV AX, 2
MOV BX, 3
ADD AX, BX
MOV BX, 4
ADD AX, BX
MOV BX, 1
ADD AX, BX
MOV BX, 5
ADD AX, BX
MOV BX, 5
DIV BL
INT 3	

Output:	AX=0003 BX=0005 CX=0000 DX=0000
Result Verification:	Avg= (2+3+4+1+5)/5= 15/5 AL = 3, AH = 0
Discussion:	

At first, 2 replaced in Ax register and the address is 0404 and 3 replaced in Bx register and its address is 0407. Now, Ax, Bx are added in 040A then 4 replaced in Bx in 040C, and Ax, Bx are added again in 040F. Now, 1 replaced in Bx and the address is 0411. Then Bx is added in 0414 address and again 5 replaced in Bx and the address is 0416 then Bx is added in 0419. Now, 5 replaced in Bx and the address is 041B. Then Ax is divided by BL in 041D address. After pressing STP and REG, it shows the result.

Problem 5: Floor size 20*20, Tiles size 2*2. How many tiles are needed to cover up the floor?

```
MOV   AX, 20
MOV   BX, 20
MUL   BL
MOV   CX, AX
MOV   AX, 2
MOV   BX, 2
MUL   BL
MOV   BX, AX
MOV   AX, CX
DIV    BL
INT 3
```

Output:	AX=0064 BX=0004 CX=0190 DX=0000
Result Verification:	Tiles = (20*20)/ (2*2) = 400/4= 100 = 64H

Problem 6: Factorial Operation: 5! – 3!

```
MOV   AX, 1
MOV   CL, 5
L1:       MUL CL
LOOP L1
MOV  DX, AX
MOV  AX, 1
MOV  CL, 3
L2:   MUL CL
LOOP   L2
MOV   BX, AX
```

```
MOV   AX, DX
SUB   AX, BX
INT   3
```

Output:	AX=0072 BX=0006 CX=0000 DX=0078
Result Verification:	5! – 3! = 114 = 72H; AH = 00, AL = 72

Discussion:
At first, we load 1 in AX register and load 5 in CL register then do multiply by giving loop with CL address and move AX value in DX register. Now, again entered value 1 in AX register and 3 replaced in CL register then do multiply by giving loop with CL address and move AX value in BX register. Then move the DX value in AX register and do subtraction of AX and BX. After pressing STP and REG, it produces the result.

Problem 7: (5! / 3!) + 4!

```
MOV   AX, 1
MOV  CL, 5
L1:MUL   CL
LOOP  L1
MOV   DX, AX
MOV   AX, 1
MOV   CL, 3
L2:MUL   CL
LOOP  L2
MOV   BX, AX
MOV   AX, DX
DIV    BL
MOV  DX, AX
MOV  AX, 1
MOV  CL, 4
L3:      MUL   CL
LOOP  L3
ADD   AX,DX
INT    3
```

Output:	AX=002C BX=0006 CX=0000 DX=0014

Result Verification:	(5! / 3!)+4!=(120/6)+24=20+24=44=2C H, AH=00, AL=2C
Discussion: At first, load 1 in AX register and load 5 in CL register then do multiply by giving loop with CL register and move AX value in DX register. Again, load value 1 in AX register and 3 in CL register then do multiply by giving loop with CL register and move AX value in BX register. Then move the DX value in AX register and do division by BL. Now, move AX value in DX and again entered value 1 in AX register and 4 replaced in CL register then do multiply by giving loop with CL address. At last, we do addition of DX and AX. After pressing STP and REG, we get the result.	

Problem 8: (2! *3! *4!) +4!

```
MOV   AX, 1
MOV   CL, 2
L1:MUL   CL
LOOP  L1
MOV   DX, AX
MOV   AX, 1
MOV   CL, 3
L2:MUL   CL
LOOP  L2
MOV   BX, AX
MOV   AX, DX
MUL    BL
MOV   DX, AX
MOV   AX, 1
MOV   CL, 4
L3:MUL   CL
LOOP  L3
ADD   AX, DX
INT     3
```

Output:	AX=0138 BX=0018 CX=0000 DX=0120
Result Verification:	(2! * 3!) + 4! = (2 * 6) + 24 = 12 + 24 = 36 = 24 HDH=00, DL=24

Discussion:
At first, load 1 in AX register and load 2 in CL register then do multiply by giving loop with CL address and move AX value in DX register. Again, enter value 1 in AX register and 3 in CL register then do multiply by giving loop with CL address and move AX value in BX register. Then move the DX value in AX register and do division by BL. Now, move AX value in DX and again load value 1 in AX register and 4 in CL register then do multiply by giving loop with CL address. Now, we do addition of DX and AX in address. At last, we do addition of DX and AX. After pressing STP and REG, we get the result.

Problem 9: (4!/2!)/3!

```
MOV   AX, 1
MOV   CL, 4
L1:
  MUL   CL
LOOP   L1
MOV   DX, AX
MOV   AX, 1
MOV   CL, 2
L2:
  MUL   CL
LOOP   L2
MOV    BX, AX
MOV   AX, DX
DIV    BL
 MOV   DX, AX
MOV   AX, 1
MOV   CL, 3
L3:
  MUL   CL
LOOP   L3
MOV   BX, AX
MOV   AX, DX
DIV    BL
INT     3
```

Output:	AX=0002 BX=0006 CX=0000 DX=000C
Result Verification:	(4! / 2!)/3! =(24/2)/6=12/6=2=2HAH=00, AL=02

Discussion:

At first, 1 replaced in AX register and 4 replaced in CL register then do multiply by giving loop with CL address and move AX value in DX register. Again, entered value 1 in AX register and 2 replaced in CL register then do multiply by giving loop with CL address and move AX value in BX register. Then move the DX value in AX register and do division by BL. Now, move AX value in DX and again entered value 1 in AX register and 3 replaced in CL register then do multiply by giving loop with CL address. Now, we move the AX value in BX register then move the DX value in register AX and divide the value by BL. After pressing STP and REG, we get the result.

Problem 10: Byte with Byte Division

```
ORG 100h
.MODEL SMALL
.DATA
NUM_1 DB 0F2H
NUM_2 DB 4H
.CODE
MOV BH, NUM_2     ;Load numerator in BH
MOV AL, NUM_1     ;Load denominator in AL
DIV BH            ;Divide BH by AL
RET
```

Output: The DIV instruction divides BH by AL. F2 divided by 04 gives quotient of 3C and give 02 as a remainder. AL stores the quotient and remainder is stored in AH register.	AX=023C

- ORG (abbr. for ORiGin) is an assembly directive (not an instruction). It defines where the machine code (translated assembly program) is to place in memory. As for ORG 100H this deals with 80x86 COM program format (COMMAND) which consist of only one segment of max. 64k bytes. 100H says that the machine code starts from address (offset) 100h in this segment, effective address is CS:100H.
- With .model small you get a program where CS points to a 64k bytes code segment and DS point to 64k bytes data segment. Thus, code and data both use 64k bytes maximum space.

```
.MODEL MEDIUM     ;the data must fit into 64K bytes
                  ;but the code can exceed 64K bytes of memory
.MODEL COMPACT    ;the data can exceed 64K bytes
                  ;but the code cannot exceed 64K bytes
.MODEL LARGE      ;both data and code can exceed 64K
                  ;but no single set of data should exceed 64K
.MODEL HUGE       ;both code and data can exceed 64K
                  ;data items (such as arrays) can exceed 64K
.MODEL TINY       ;used with COM files in which data and code
                  ;must fit into 64K bytes
```

<table>
<tr><td colspan="2">Problem 11: Word with Word Division</td></tr>
<tr><td colspan="2">ORG 100h
.MODEL SMALL
.DATA
NUM_1 DW 0F213H
NUM_2 DW 41A8H
.CODE
MOV AX, NUM_1 ;Load numerator in AX
DIV NUM_2 ;Divide AX by NUM_2
RET</td></tr>
<tr><td><u>Output:</u>

The output window shows that the division of F213H by 41A8 gives the remainder of 2D1B into DX register and 03 as a quotient into AX.</td><td>AX=0003

DX=2D1B</td></tr>
<tr><td colspan="2"></td></tr>
</table>

Conclusion:

In this experiment, we have learnt conversion from °C to °F, conversion from °F to °C, conversion from °C to °K, Conversion from °K to °C, Average of 3 numbers, average of 5 numbers, area of rectangle, area of triangle, find how many tiles. After performing those operation, we use assembly language in 8086 microprocessors which results in getting the correct output.

Example: HomeWorks

1. Temperature conversion from °C to °F (37°C)
2. Temperature conversion from °F to °C (110°F)
3. Temperature conversion from °F to °K (130°F) ; AX=0547H; K=(F-32)*5/9+273;
4. Temperature conversion from °K to °F (300°K); F=9(K-273)/5+32
5. 3! +4!
6. (4! + 3!) - 2!
7. (1! * 2!) * 6!
8. Find out the average of the ten numbers.
9. Factorial Operation: 7! – 4! + 2!
10. Floor size 80*80, Tiles size 4*4. How many tiles will be required to pave up the floor?

Session 2

Session Objective:

- To get familiar with 8051 Microcontroller and its simulation tools including Keil C51 Evaluation Kit and Proteus Kit.
- To simulate a simple example program-LED Blinking with 8051 Microcontroller.
- To understand the timer system (crystal oscillator circuit) and the relationship between clock frequency ND and microprocessor speed.

Experiment 1: To get familiar with 8051 Microcontroller and its simulation tools including Keil C51 Evaluation Kit and Proteus Kit.

Microcontroller (MC) is called a computer on chip science it includes microprocessor with internal ROM, RAM, parallel and serial ports within single chip. MC is broadly used in washing machines, vcd player, microwave oven, robotics and etc. 8051 is an 8-bit microcontroller, means 8-bit data bus, means able to read, write and process 8-bit data. Architecture of 8051 is presented in figure 1. 8051 executes code from an embedded masked ROM. Intel's original MCS-51 family was developed using N-type metal-oxide-semiconductor (NMOS) technology like its predecessor Intel MCS-48, but later versions, identified by a letter C in their name (e.g., 89C51), used complementary metal-oxide-semiconductor (CMOS) technology and consumed less power than their NMOS predecessors. This made them more suitable for battery powered devices.

In this particular experiment you are going to use AT89C52, which is an 8-bit microcontroller and belongs to Atmel's 8051 family. AT89C52 has 8KB of Flash programmable and erasable read only memory (PEROM) and 256 bytes of RAM. AT89C52 has an endurance of 1000 Write/Erase cycles which means that it can be erased and programmed to a maximum of 1000 times.

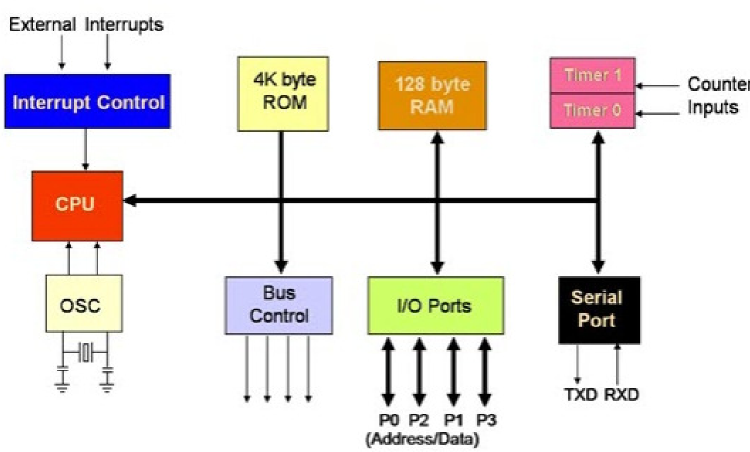


Figure 2.1: 8051 Microcontroller Architecture

You need to understand three (3) parts of 8051 including –Oscillator and I/O ports to understand this experiment.

Oscillator: It is used for providing the clock to 8051 MC (using to input pins XTAL2 and XTAL1) and decides the speed of MC. In this experiment, you are going to use a crystal oscillator and its frequency varies from 4MHz to 30 MHz, but normally it formulates11.0592 MHz frequency.

Input Output Ports: There are four input-output ports available P0, P1, P2, and P3. Each port is 8-bit wide and has special function registers P0, P1, P2, and P3 which are bit addressable which, means each bit can be set or reset by the Bit instructions (SETB for high, CLR for low) independently. The data at any port that is transmitting or receiving is in these registers. The port 0 can perform dual works. It is used as a Lower order address bus (A0 to A7) multiplexed with 8-bit data bus P0.0 to P0.7 is AD0 to AD7 respectively the address bus and data bus are demultiplex by the ALE signal and latch which is further discussed in details. P1 is a true I/O port (P1.0 to P1.7), because it doesn't have any alternative functions as is the case with P0, but can be configured as general I/O only. Port 2 can be used as an I/O port as well as a higher order address bus A8 to A15. Port 3 also has dual functions it can be worked as I/O as well and each pin of P3 has a specific function and you will learn in details of each port later.

Figure 2.2: AT89c51 Architecture

Tools:

1. **KeilμVission5 -** Keil Microcontroller Tool includes C/C++ compilers, integrated development environments, RTOS, middleware, as well as debug adapters and evaluation boards for Arm Cortex®-M based devices.
2. **Proteus 8 Professionals-** Proteus is a complete software solution for circuit simulation and PCB design. It comprises several modules for schematic capture, firmware IDE, and PCB layout that appear as tabs inside a single, integrated application. Proteus virtual system modeling (VSM) bridges the gap in the design life cycle between schematic capture and PCB layout. It enables you to write and apply your firmware to a microcontroller component on the schematic (PIC, AVR, ARM, 8051, etc.) and then co-simulate the program within a mixed-mode SPICE circuit simulation.

Experiment 2: Complete the following task according to the given instructions:

1. Open: KeilμVision 5

2. Project (From Menu Bar)

3. New Vision Project 4. Create a project folder in Desktop (e. g. CSE3118lab) and

5. Open a file with a name (e.g. cse3118)

6. Save

7. A window appears: Select device for Target 1 ‘Target 1’

8. Click on ATML (+)

9. Select AT89C51 (8051-based Fully Static 24 MHz CMOS controller with 32 I/O lines)

10. OK

11. Yes

12. Click File (From menu bar)

13. Select new

14. File (from menu bar)

15. Click on Save as

16. Give a file name with extension .c (e.g. cse3118.c)

17. Write the following code on the text file.

```
#include void delay(unsigned int);
void main(void) {
   P1_1=0; P1_2=0;
   delay(300);
   P1_1=1; P1_2=1;
   delay(300);
}
void delay(unsigned int itime) {
    int i,j;
    for(i=0;i<itime;i++);
    for(j=0, j<5000; j++);
}
```

18. Click on +Target 1 (From the left side of the window)

19. Click on Rt. Mouse button on Source Group 1

20. Select Add existing files to Group ‘Source Group 1’

21. Select: file c3118.c

22. Click on Add

23. Click on Close

24. Create an environment for creating a Hex file by clicking Target 1 (Rt. Mouse button)

25. Select Option for Target 'Target 1'

26. Click on Target

27. Set frequency Xtal (MHz): 12 MHz

28. Select Output and

29. Click on Create Hex File (Hex: 80)

30. OK

31. Click on Translate (or Press CTRL + F7: To see the errors on the code Menu bar)

32. Click on Build (or F7)

33. Go to the Project folder (CSE442lab)

34. Go to the Objects folder in the Project folder (CSE442lab)

35. Find Hex file (cse442.hex)

Go to the software PROTEUS Professionals

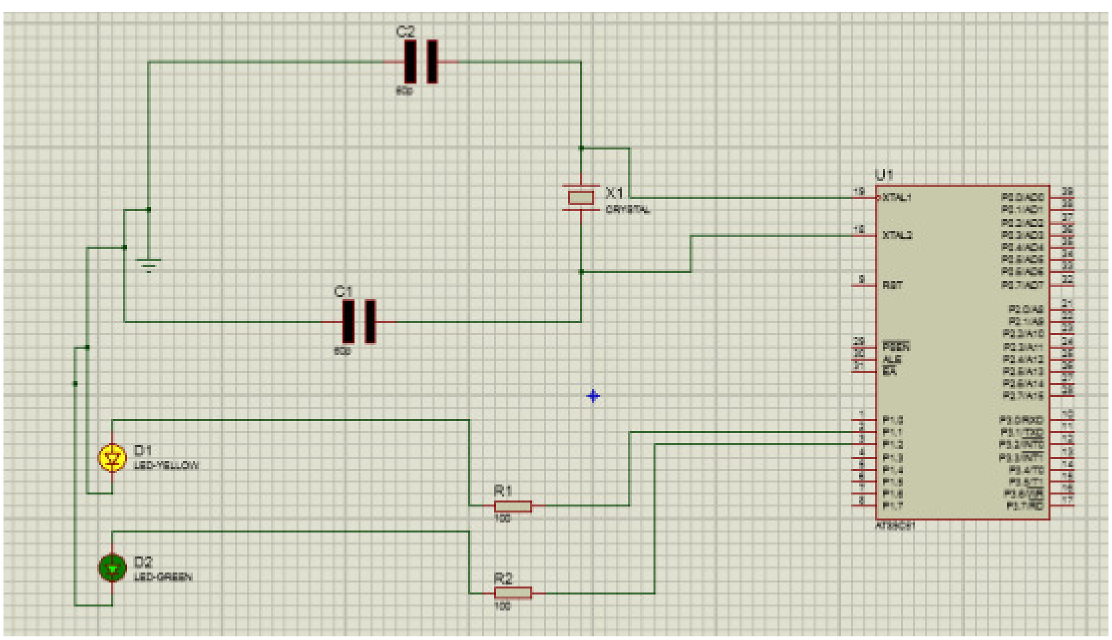


Figure 2.3: Circuit Diagram of the Experiment

Draw the above circuit diagram by selecting the appropriate tools from the PROTEUS.

a. Open: Proteus 8 Professionals
b. Select: ISIS
c. Select: P (Pick Devices) for Search
d. Components: AT89C51, Resistor, LED, Capacitors, Crystal to design the above diagram, (by typing in the keyword area) and select a terminal mood from the right side for ground.
e. Set the value of all resistors to 100 ohms (right mouse click on resistor)
f. Set the value of all capacitors to 60 picos
g. Set value crystal 12MHz
h. Right click on AT89C51(circuit) and select program from project folder -> object folder->cse3118.hex file->open-> ok
i. Click Run (On the bottom left corner) of the screen.

Experiment 3: Complete each of the following tasks and write the effect in your system.

• Change the code to increase delay (3000)

• Change the clock speed from 12MHz to 24MHz

Experiment -4: Simulate 8086 interfacing with 7 segments using the masm32 compiler.

Session 3

Session Objective:

This session will help the student to get introduced to the basics of Arduino, its various models, its basic programming structure, and how to get started with a few simple experiments. They will also form teams for their upcoming group projects and discussion will be held about the social responsibility, environment and sustainability issues.

Experiment 1: Interfacing basic LED blink with Arduino.

Objective: Interfacing Arduino analog and digital ports.

Description: The pins on the Arduino can be configured as either inputs or outputs. Pins configured as OUTPUT with pinMode() are said to be in a low-impedance state. This means that they can provide a substantial amount of current to other circuits. Atmega pins can source (provide positive current) or sink (provide negative current) up to 40 mA (milliamps) of current to other devices/circuits. This is enough current to brightly light up an LED (don't forget the series resistor), or run many sensors, for example, but not enough current to run most relays, solenoids, or motors. Short circuits on Arduino pins, or attempting to run high current devices from them, can damage or destroy the output transistors in the pin, or damage the entire Atmega chip. Often this will result in a "dead" pin in the microcontroller but the remaining chip will still function adequately. For this reason, it is a good idea to connect OUTPUT pins to other devices with 470Ω or 1k resistors, unless maximum current draw from the pins is required for a particular application.

Design:

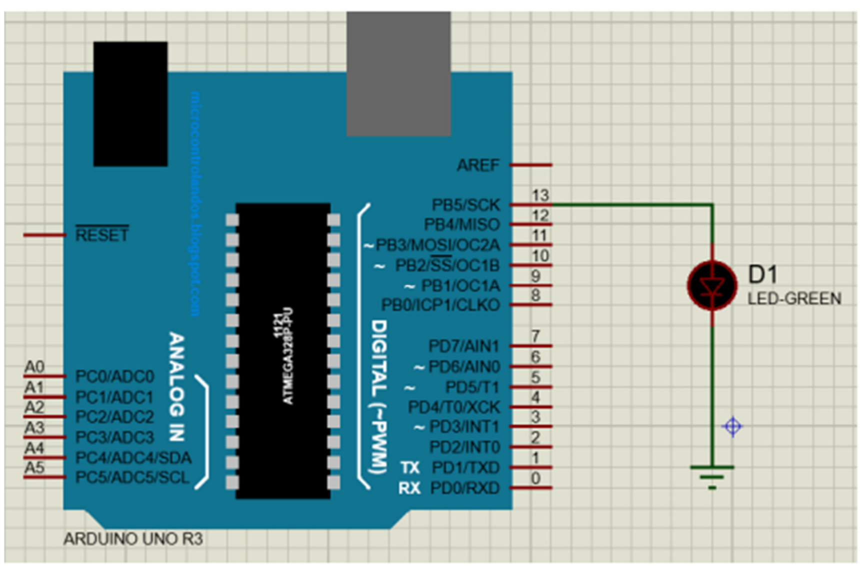


Figure 3.1: Implementation of simple Arduino based circuits with LED.

Exercises:

- Design Arduino based switch controlled LED system.

Experiment 2: Interfacing basic 4x4 keypad with Arduino.

Objective: Interfacing Arduino digital ports with 4x4 keypad and use of serial monitor for value cross checking.

Description: The pins on the Arduino can be configured as either inputs or outputs. Pins configured as OUTPUT with pinMode() are said to be in a low-impedance state. This means that they can provide a substantial amount of current to other circuits. Atmega pins can source (provide positive current) or sink (provide negative current) up to 40 mA (milliamps) of current to other devices/circuits. This is enough current to brightly light up an LED (don't forget the series resistor), or run many sensors, for example, but not enough current to run most relays, solenoids, or motors. Short circuits on Arduino pins, or attempting to run high current devices from them, can damage or destroy the output transistors in the pin, or damage the entire Atmega chip. Often this will result in a "dead" pin in the microcontroller but the remaining chip will still function adequately. For this reason, it is a good idea to connect OUTPUT pins to other devices with 470Ω or 1k resistors, unless maximum current draw from the pins is required for a particular application.

Design:

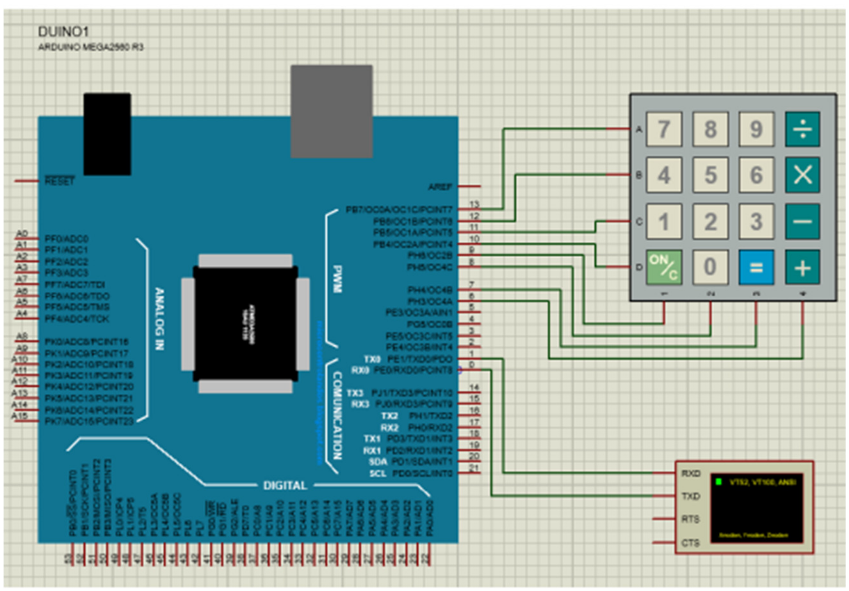


Figure 3.2: Interfacing basic 4x4 keypad with Arduino.

Exercises:

- Design a simple calculator using Arduino and available keypad.

Experiment 3: Printing Potentiometer value to Serial Monitor.

Objective: Interfacing Arduino analog port with potentiometer and use of serial monitor for observation of value for cross checking.

Description: The analog pins on the Arduino can be configured as either inputs or outputs. Pins configured as INPUT with pinMode() can be used to take input from devices that generate an analog value. Potentiometers are devices that can produce variable resistance. In this experiment we shall connect the middle wire (wiper) of the potentiometer to an analog port of an Arduino, the other two terminals are connected to power and ground ports of Arduino respectively. Thus when the wiper is rotated, variable resistance is provided as input to the analog pin of Arduino, which is read and output is shown correspondingly to serial monitor.

Design:

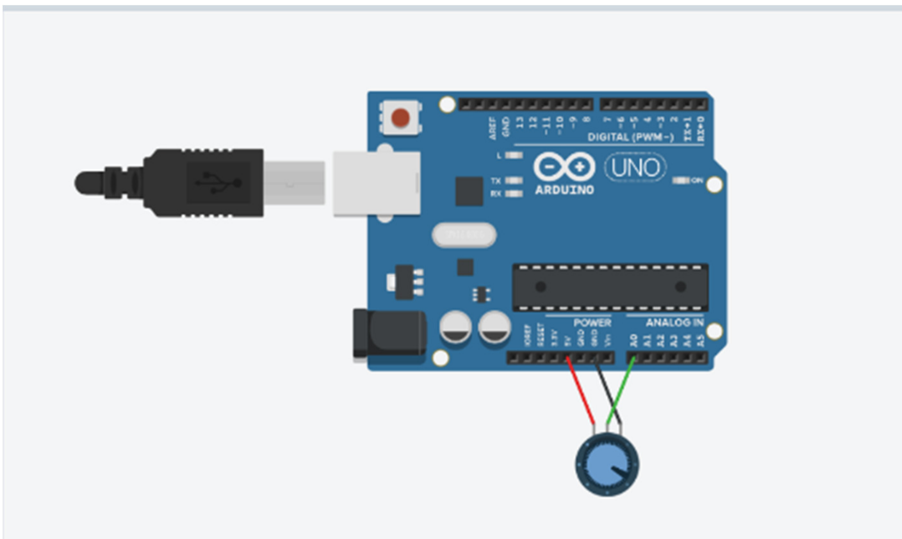


Figure 3.3: Interfacing 4x4 Keypad with Arduino.

Exercises:

- Design a light with dimming/brightening effect using Potentiometer and LED.

Experiment 4: Interfacing Piezo Buzzer with Arduino.

Objective: Interfacing Arduino digital port with a simple buzzer.

Description: Piezo buzzer is an electronic device commonly used to produce sound. Piezo buzzer is based on the inverse principle of piezo electricity discovered in 1880 by Jacques and Pierre Curie. It is the phenomena of generating electricity when mechanical pressure is applied to certain materials and the vice versa is also true. Such materials are called piezo electric materials. Piezoceramic is class of manmade piezoelectric material, which poses piezo electric effect and is widely used to make "disc", the heart of piezo buzzer. When subjected to an alternating electric field they stretch or compress, in accordance with the frequency of the signal thereby producing sound. Built-in functions in the Arduino library delay(), tone() and noTone() may be used.

Design:

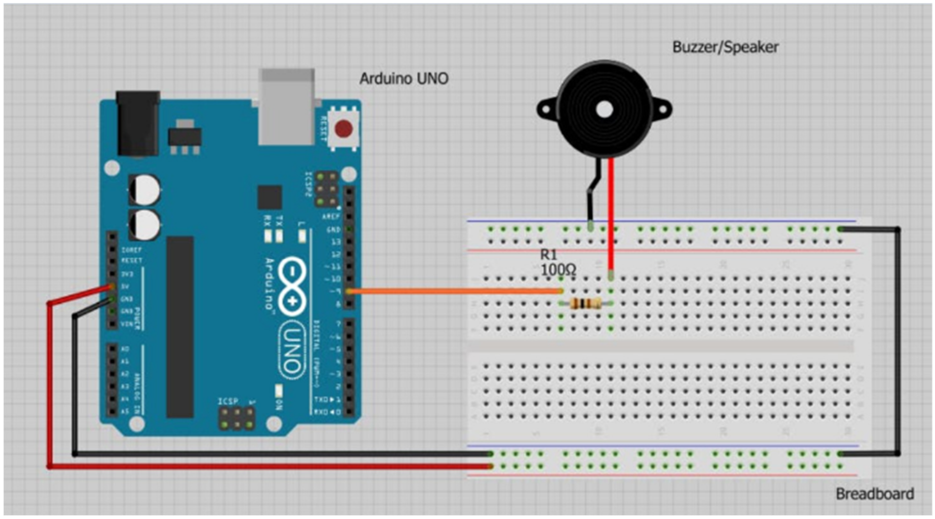


Figure 3.4: Interfacing Piezo Buzzer with Arduino.

Exercises:

- Students can test out how various frequency changes in the tone() function changes the output of the buzzer.

Experiment 5: Interfacing Servo motor with Arduino.

Objective: Interfacing Arduino digital port with a servo motor.

Description: A servo motor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity, and acceleration in a mechanical system. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servo motors. The motors used in the lab has a rotational range of 180°. It consists of DC motor, gear system, Position sensor and Control circuit. Degree of rotation can be controlled by applying the Electrical Pulse of proper width, to its Control pin. Servo library and its associated functions such as attach(), write() may be used.

Design:

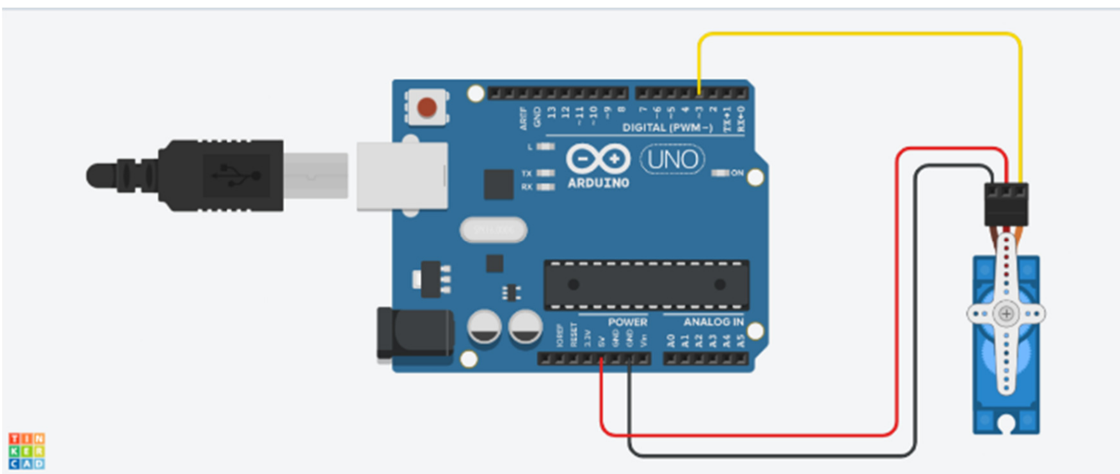


Figure 3.5: Interfacing Servo motor with Arduino.

Exercises:

- Students can test out how various delay changes in code changes the output of the motor.
- Students may experiment how to connect both Servo motor and Buzzer to the same Arduino and combine the separate coding functions to make a cohesive program.

Experiment 6: Interfacing Ultrasonic Sensor with Arduino.

Objective: Interfacing Arduino digital port with an Ultrasonic Sensor.

Description: An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. High-frequency sound waves reflect from boundaries to produce distinct echo patterns. Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. The ultrasonic sensors used in the lab, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse. The working principle is simple. It sends an ultrasonic pulse out at 40kHz which travels through the air and if there is an obstacle or object, it will bounce back to the sensor. By calculating the travel time and the speed of sound, the distance can be calculated. Built-in functions in the Arduino library digitalWrite() and delay() may be used.

Design:

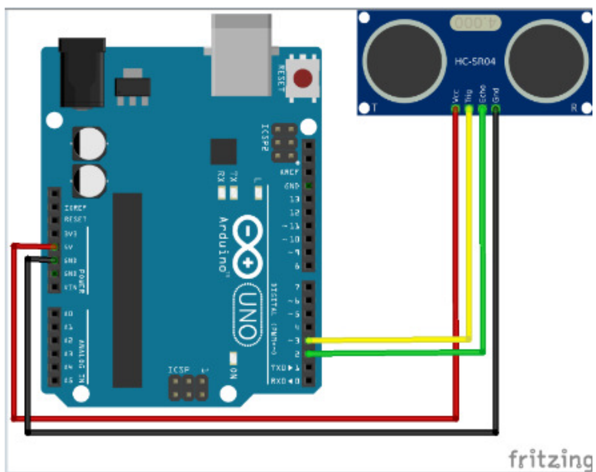


Figure 3.6: Interfacing Ultrasonic Sensor with Arduino.

Exercises:

- Students can test out how obstacles change the output of the sensor.
- Students may experiment how to connect Servo motor, buzzer and the ultrasonic sensor to the same Arduino and combine the separate coding functions to make a cohesive program.

Experiment 7: Interfacing basic LCD display with Arduino.

Objective: Interfacing 2x20 alphanumeric LCD with Arduino digital ports.

Description: The LiquidCrystal library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface. The example sketch provided prints "Hello World!" to the LCD and shows the time in seconds since the Arduino was reset.

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

- A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
- A Read/Write (R/W) pin that selects reading mode or writing mode
- An Enable pin that enables writing to the registers
- 8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.
- There's also a display constrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The LiquidCrystal Library simplifies this for you so you don't need to know the low-level instructions. The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 2x16 LCD in 4-bit mode.

Design:

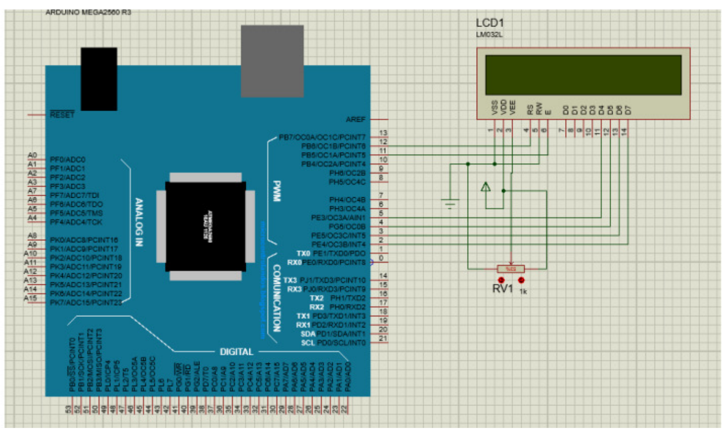


Figure 3.7: Interfacing basic LCD display with Arduino.

Exercises:

- Students can test out how to design a simple calculator using Arduino, available keypad and 2x20 LCD display.
- Students can test out how to design a system to show values in seven segment display

Experiment 8: Interfacing Temperature Sensor with Arduino.

Objective: Interfacing Arduino digital port with an LM-35 temperature sensor.

Description: LM35 is a temperature sensor that outputs an analog signal which is proportional to the instantaneous temperature. The output voltage can easily be interpreted to obtain a temperature reading in Celsius. The advantage of lm35 over thermistor is it does not require any external calibration. The coating also protects it from self-heating. LM35 can measure from -55 degrees centigrade to 150-degree centigrade. The accuracy level is very high if operated at optimal temperature and humidity levels. The conversion of the output voltage to centigrade is also easy and straight forward. The input voltage to LM35 can be from +4 volts to 30 volts. It consumes about 60 microamperes of current. In order to understand the working principle of LM35 temperature sensor we have to understand the linear scale factor. In the features of LM35 it is given to be +10 mills volt per degree centigrade. It means that with increase in output of 10 mills volt by the sensor v_{out} pin the temperature value increases by one. For example, if the sensor is outputting 100 mills volt at v_{out} pin the temperature in centigrade will be 10-degree centigrade. The same goes for the negative temperature reading. If the sensor is outputting -100 mills volt the temperature will be -10 degrees Celsius. Built-in functions in the Arduino library analogRead() and delay() may be used.

Design:

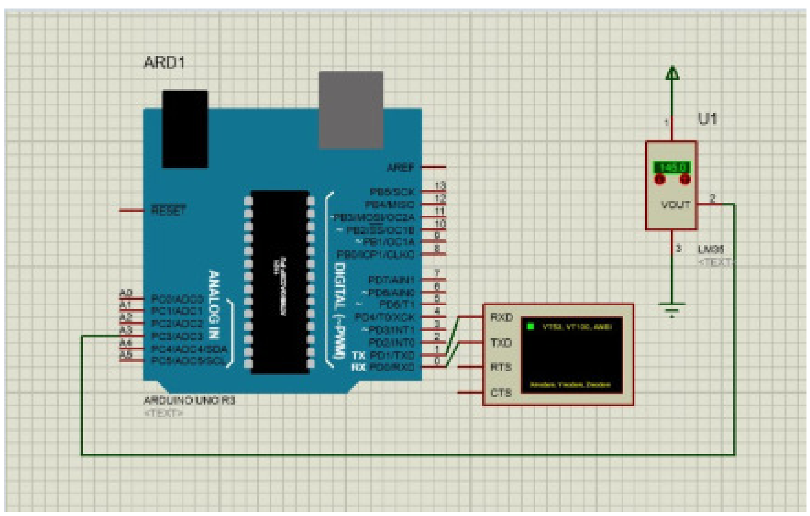


Figure 3.8: Interfacing Temperature Sensor with Arduino.

Exercises:

- Students can test out how to show temperature readings on LCD display.

Experiment 9: Interfacing DC motors with motor driver and Arduino.

Objective: Interfacing simple DC motors with motor driver L293D and Arduino; Creating the working procedures of two-wheeler car.

Description: A direct current, or DC, motor is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction. The L293D has two +V pins (8 and 16). The pin '+Vmotor (8) provides the power for the motors, and +V (16) for the chip's logic. We have connected both of these to the Arduino 5V pin. However, if you were using a more powerful motor, or a higher voltage motor, you would provide the motor with a separate power supply using pin 8 connected to the positive power supply and the ground of the second power supply is connected to the ground of the Arduino. Motor drivers acts as an interface between the motors and the control circuits. Motors require high amount of current whereas the controller circuit works on low current signals. So the function of motor drivers is to take a low-current control signal and then turn it into a higher-current signal that can drive a motor. We should not drive the motor directly from Arduino board pins. This may damage the board. Built-in functions in the Arduino library digitalWrite() and delay() may be used.

Design:

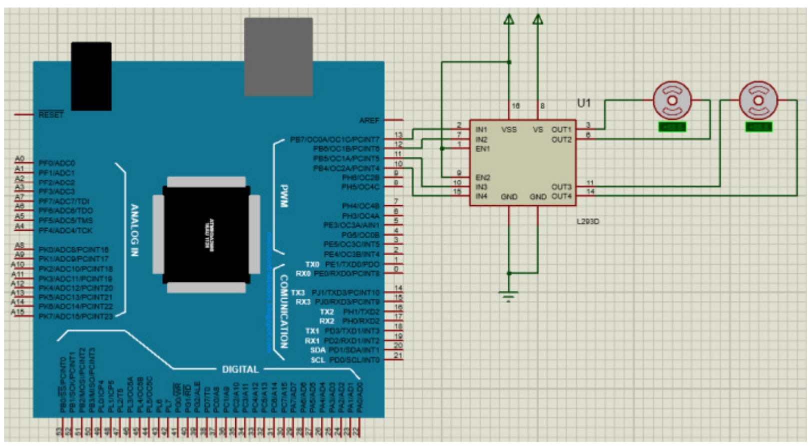


Figure 3.9: Interfacing DC motors with motor driver and Arduino.

Exercises:

- Students can test out how to design a working scenario of four-wheeler car using Arduino.

Experiment 10: Interfacing Light Dependent Resistor with Arduino.

Objective: Interfacing Arduino digital port Light Dependent Resistor with Arduino.

Description: An LDR is a component that has a (variable) resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits. Light Dependent Resistors (LDR) are also called photoresistors. They are made of high resistance semiconductor material. The resistance values of LDR in darkness are several megaohms whereas in bright light it will be dropped to hundred ohms. Built-in functions in the Arduino library analogRead() and delay() may be used.

Design:

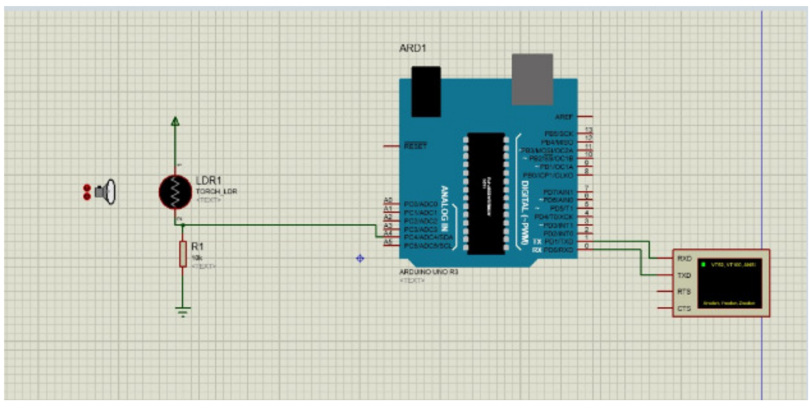


Figure 3.10: Interfacing Light Dependent Resistor with Arduino.

Exercises:

- Students can test out how to combine LDR with other components to create a cohesive system.

Experiment 11: Interfacing Infrared Sensor (Active) with Arduino.

Objective: Interfacing Arduino digital port Infrared Sensor (Active) with Arduino.

Description: An infrared sensor is an electronic device, that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detect the motion. Usually, in the infrared spectrum, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, which can be detected by an infrared sensor. The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode that is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, the resistances and the output voltages will change in proportion to the magnitude of the IR light received. This active infrared sensor includes both the transmitter as well as the receiver. In most of the applications, the light-emitting diode is used as a source. LED is used as a non-imaging infrared sensor whereas the laser diode is used as an imaging infrared sensor. Built-in functions in the Arduino library digitalWrite() and delay() may be used.

Design:

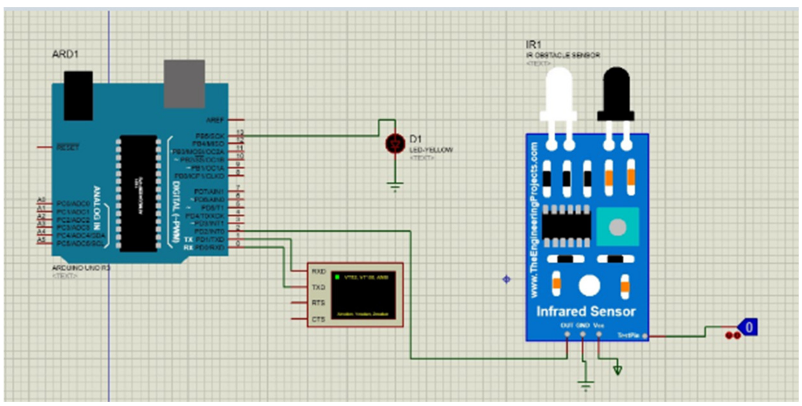


Figure 3.11: Interfacing Infrared Sensor (Active) with Arduino.

Exercises:

- Students can test out how to combine IR Sensor with other components to create a cohesive system.

Experiment 12: Interfacing Infrared Sensor (Passive) with Arduino.

Objective: Interfacing Arduino digital port Infrared Sensor (Passive) with Arduino.

Description: An infrared sensor is an electronic device, that emits Infrared Light in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detect the motion. Usually, in the infrared spectrum, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, which can be detected by an infrared sensor. The passive infrared sensor includes detectors only but they don't include a transmitter. It consists of 1 pair of pyroelectric sensors, to detect heat energy in the surrounding environment. The sensors are housed within a series of lenses. The purpose of lens is widening the device's sensing area. After that, a signal processor is used to understand the signal to obtain the required information. When the sensor is idle, both pyroelectric sensors detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected. Built-in functions in the Arduino library digitalWrite() and delay() may be used.

Design:

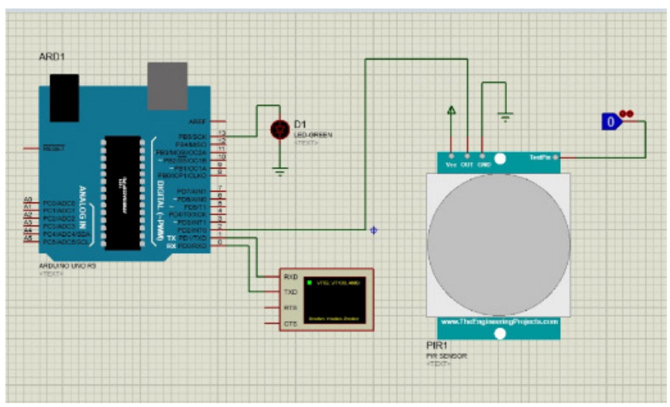


Figure 3.12: Interfacing Infrared Sensor (Passive) with Arduino.

Exercises:

- Students can test out how to combine PIR Sensor with other components to create a cohesive system.

Session 4

Session Objective:

The students will give a formal presentation where they present their plans for a group project to be built based on Arduino and various sensors. They will also submit a project proposal report at this time.

Guidelines of the Presentation:

- The students must keep the following points in their report as well as their presentations, in addition they may keep other points they deem necessary:
    - objectives
    - social values
    - required components
    - working procedure
    - estimated budget
    - conclusion
- Each group will get 15 minutes to present.
- Every group member must provide a part of the presentation. If someone does not present, they will not receive marks.
- The presentation should follow the same points as the project proposal (such as: - objectives, social values, required components etc.) but students must not just copy-paste everything word-for-word from the report. They should make it more concise. Teachers expect not to see lengthy descriptions in the presentation slides.
- There will be a Q&A portion at the end of each presentation, where students will be asked questions regarding their project, so they must be prepared to defend their choice of project.
- Presentation language: English

MID TERM EXAMINATION

There will be a 40 or 50 minutes' mid-term examination after the first half of the semester.

Different types of questions will be included in the exam, including coding, designing, theory etc.

Session 5

Session Objective:

- To get familiar with 8255A Interface, PIN and port configuration.
- To understand the connectivity between 8255 with I/O ports (P3 and P4), LEDs, 7-segments display and DOT Matrix units.
- To simulate an example program to display 0-9 digits in 7-segment display units.
- To simulate an example program to display characters in DOT MATRIX units.

Experiment 1: Introduction to 8255A Programmable Peripheral Interface and Experiment with Seven (7)-Segments Display and LED Connection Program

8255A is a general purpose programmable I/O device used in microprocessors. It consists of three 8-bit bidirectional I/O ports with 24 I/O pins (figure 1) which may be individually programmed in 2 groups of 12 pins and used in 3 major modes of operation.

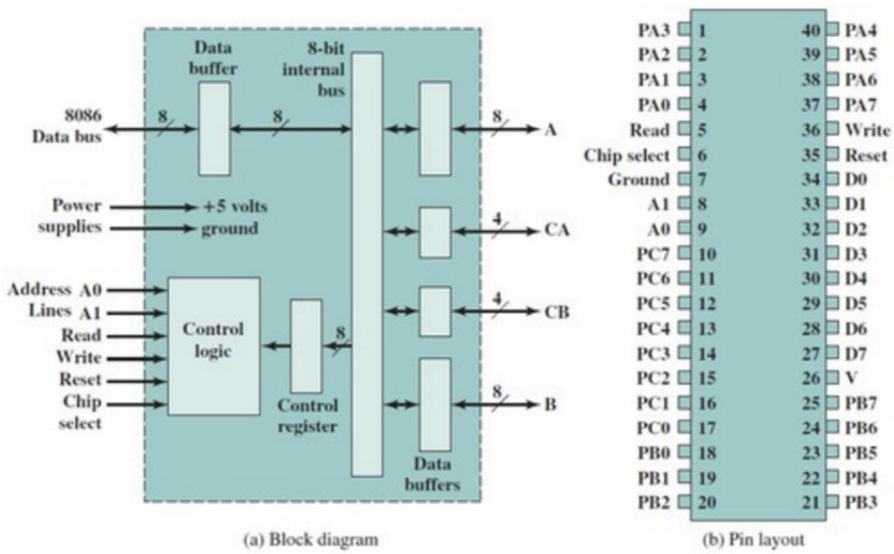


Figure 5.1: 8255 Block Diagram and PIN OUT Diagram

Carry control and status signal

A1	A0	Function
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Command Register

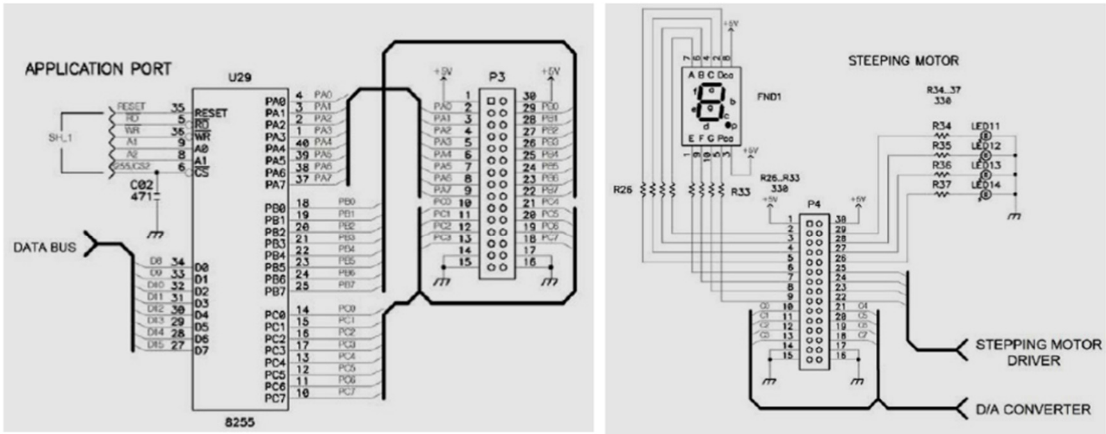


Figure 5.2: Interfacing 8255 with Seven Segment and LEDs

82C55 has three mode of operation including Mode 0, 1, 2.

Mode 0- Basic Input/Output Mode

Causes 82C55 to function either as a buffered input device the pins of Group B/Group A to be programmed as simple I/O ports.

Mode 1- Strobe Input/Output Mode

Causes operation port A and/or port B to function as latching input devices. Similar to mode 0 but data are transferred through port A/port B and handshaking (DATA READY, ACKNOWLEDGE) and interrupt request signals are provided by port C. Strobe inputs signal to microprocessor retrieve data that are stored into the port registers

The address of the control register, port A, port B and port C are given below:

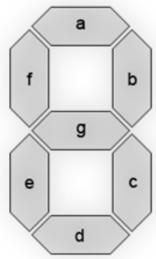
PPIC_C EQU 1FH

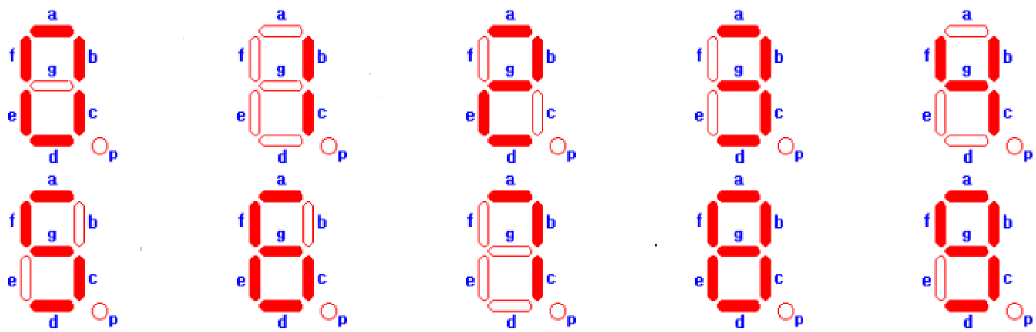
PPIC EQU 1DH

PPIB EQU 1BH

PPIA EQU 19H

Experiment 2: Write an assembly code to display 0-9 in Seven Segment Display (SSD)





- For seven segments display we use 0 for ON and 1 for OFF.
- Control register values will be the column headings of the following table:

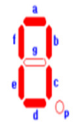
D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0
Control Register 0- BSR mode 1- I/O mode	Mode selection for group A 00- I/O 01- Handshaking		Port A 0- Output 1- Input	Upper 4 bit of port C	Mode selection for group B 0- I/O 1- Handshaking	For port B	For lower 4 bit of port C

Assembly Code:

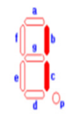
```
S SEGMENT PARA PUBLIC 'CODE'
ASSUME CS: S
ORG 1000H

START:
   ;control register turn on
    MOV AL,80H OUT 1FH,AL
SSD:
     ;display 0
     MOV AL,0C0H
     OUT 19H,AL
     ;for delay
     MOV CX,0FFFFH
     L0:LOOP L0
     ;display 1
     MOV AL,0F9H
     OUT 19H,AL
     ;for delay
     MOV CX,0FFFFH
     L1:LOOP L1
     ;display 2
     MOV AL,0A4H
     OUT 19H,AL
     ;for delay
     MOV CX,0FFFFH
     L2:LOOP L2
     ;display 3
     MOV AL,0B0H
     OUT 19H,AL
     ;for delay
     MOV CX,0FFFFH
     L3:LOOP L3
     ;display 4
     MOV AL,099H
     OUT 19H,AL
     ;for delay
     MOV CX,0FFFFH
     L4:LOOP L4
     ;display 5
     MOV AL,092H
     OUT 19H,AL
     ;for delay
```

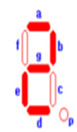
	g	f	e	d	c	b	a
1	1	0	0	0	0	0	0



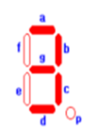
	g	f	e	d	c	b	a
1	1	1	1	1	0	0	1



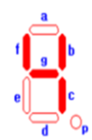
	g	f	e	d	c	b	a
1	0	1	0	0	1	0	0



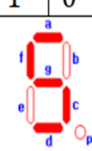
	g	f	e	d	c	b	a
1	0	1	1	0	0	0	0



	g	f	e	d	c	b	a
1	0	0	1	1	0	0	1

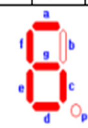


	g	f	e	d	c	b	a
1	0	0	1	0	0	1	0

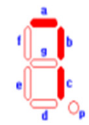


```
    MOV CX,0FFFFH
    L5:LOOP L5
    ;display 6
    MOV AL,082H
    OUT 19H,AL
    ;for delay
    MOV CX,0FFFFH
    L6:LOOP L6
    ;display 7
    MOV AL,0F8H
    OUT 19H,AL
    ;for delay
    MOV CX,0FFFFH
    L7:LOOP L7
    ;display 8
    MOV AL,080H
    OUT 19H,AL
    ;for delay
    MOV CX,0FFFFH
    L8:LOOP L8
    ;display 9
    MOV AL,090H
    OUT 19H,AL
    ;for delay
    MOV CX,0FFFFH
    L9:LOOP L9
    JMP SSD
S ENDS
END START
```

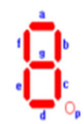
	g	f	e	d	c	b	a
1	0	0	0	0	0	1	0



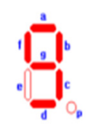
	g	f	e	d	c	b	a
1	1	1	1	1	0	0	0



	g	f	e	d	c	b	a
1	0	0	0	0	0	0	0



	g	f	e	d	c	b	a
1	0	0	1	0	0	0	0



Steps to Run code in MDA-8086 through PC:

- At first copy paste the .ASM file in the mda folder of computer
- Then open cmd and write cd\ and press enter
- Then type cd mda and press enter
- Then type MASM and press enter
- Then write the file_name.ASM and press enter. For our example we will write S.ASM
- Then write the file_name.OBJ and press enter. For our example we will write S.OBJ
- Then write the file_name.LST and press enter. This step is used for error checking. For our example we will write S.LST
- Then when it wants .CRF file simply press enter
- If there is any error in the file, then after this line we can see the number of errors.
- If any error is found, then type EDIT file_name.LST and press enter.
- If no error is found, then type LOD186 and press enter
- Then type file_name.OBJ and press enter. For our example we will write S.OBJ
- Then type file_name.ABS and press enter. For our example we will write S.ABS
- Then type COMM and press enter.
- Then a blue window will occur
- We will now turn on the kit and we will select PC mode from kit mode

- Then press RESET
- If your kit is ok, then it will show up in the blue screen
- Then type L from keyboard and press enter
- If L does not show up, then it means your PC is not connected and you have to try in different PC
- Otherwise press F3 and in the pop-up screen write filename.ABS and press enter. For our example we will write S.ABS
- Then in the kit select kit mode from PC mode
- Then press RESET
- After that press AD
- Then Press GO
- Then you can see the output in the seven segments display

Example 1: Display digits 0–9 and some characters on a 7-segment display (Kit mode).
To display the digits 0 – 9 the bit-patterns are given below.

```
P G F E D C B A
1 1 0 0  0  0  0 0 B
1 1 1 1  1  0  0 1 B
1 0 1 0  0  1  0 0 B
1 0 1 1  0  0  0 0 B
1 0 0 1  1  0  0 1 B
1 0 0 1  0  0  1 0 B
1 0 0 0  0  0  1 0 B
1 1 1 1  1  0  0 0 B
1 0 0 0  0  0  0 0 B
1 0 0 1  0  0  0 0 B
```

Task 1: Write and run the following program to display 0-9

```
B0 80        MOV AL, 10000000B
E6 1F        OUT 1F, AL          ;GOES TO CONTROL REG
B0 F0        MOV AL,11110000B
E6 1B        OUT 1B, AL          ;GOES TO PORT B
B0 00        MOV AL,00000000B
E6 1D        OUT 1D,AL           ; GOES TO PORT C

;DISPLAY STARTS HERE
LEVEL:  ;should be 100CH if the offset address of the code is 1000H
B0 C0        MOV AL, 11000000B
E6 19        OUT 19,AL           ; '0' GOES TO PORT A
B0 F9        MOV AL, 11111001B
E6 19        OUT 19,AL           ;'1'  GOES TO PORT A
B0 A4        MOV AL, 10100100B
E6 19        OUT 19,AL           ; '2' GOES TO PORT A
B0 B0        MOV AL, 10110000B
E6 19        OUT 19,AL           ; '3' GOES TO PORT A
B0 99        MOV AL, 10011001B
E6 19        OUT 19,AL           ; '4' GOES TO PORT A
B0 92        MOV AL, 10010010B
E6 19        OUT 19,AL           ; '5' GOES TO PORT A
B0 82        MOV AL, 10000010B
E6 19        OUT 19,AL           ; '6' GOES TO PORT A
```

```
B0 F8        MOV AL, 11111000B
E6 19        OUT 19,AL           ; '7' GOES TO PORT A
B0 80        MOV AL, 10000000B
E6 19        OUT 19,AL           ; '8' GOES TO PORT A
B0 90        MOV AL, 10010000B
E6 19        OUT 19,AL           ;  '9' GOES TO PORT A

EA  0C                                      ; GIVE THE ADDRESS OF THE INSTRUCTION
MOV AL, 11000000B
10 00        JMP [LEVEL]
00
```

Assignment 1: Write a program to display the hexadecimal digits.
Assignment 2: Write a program to glow the four LEDs of Figure 2 one by one.
Hints: You need to change command register to get the LED on.

```
B0 0F        MOV AL, 00001111B  ; B NEED TO BE WORK AS OUTPUT
E6 1B        OUT 1B, AL          ;GOES TO CONTROL REG
```

Experiment 3: Write an assembly code to glow R1, G, Y and R2 in LED Display respectively.

- For LED display we use 1 for ON and 0 for OFF
- Control register value will be the column headings of the following table:

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0
Control Register 0- BSR mode 1- I/O mode	Mode selection for group A 00- I/O 01- Handshaking		Port A 0- Output 1- Input	Upper 4 bit of port C	Mode selection for group B 0- I/O 1- Handshaking	For port B	For lower 4 bit of port C

Assembly Code:

```
L SEGMENT PARA PUBLIC 'CODE'
ASSUME CS: L
ORG 1000H
START:
;control register turn on
MOV AL,80H
OUT 1FH,AL
;segment address forcefully off
MOV AL,0FFH
OUT 19H,AL
LED:
;R1 LED turn on
MOV AL,01H
OUT 1BH,AL
;for delay
MOV CX,0FFFFH
LR1:LOOP LR1
;G LED turn on
MOV AL,02H
OUT 1BH,AL
;for delay
MOV CX,0FFFFH
LG:LOOP LG
;Y LED turn on
MOV AL,04H
OUT 1BH,AL
;for delay
MOV CX,0FFFFH
LY:LOOP LY
;R2 LED turn on
MOV AL,08H
OUT 1BH,AL
;for delay
MOV CX,0FFFFH
LR2:LOOP LR2
JMP LED
L ENDS
END START
```

				R2	Y	G	R1
0	0	0	0	0	0	0	1

				R2	Y	G	R1
0	0	0	0	0	0	1	0

				R2	Y	G	R1
0	0	0	0	0	1	0	0

				R2	Y	G	R1
0	0	0	0	1	0	0	0

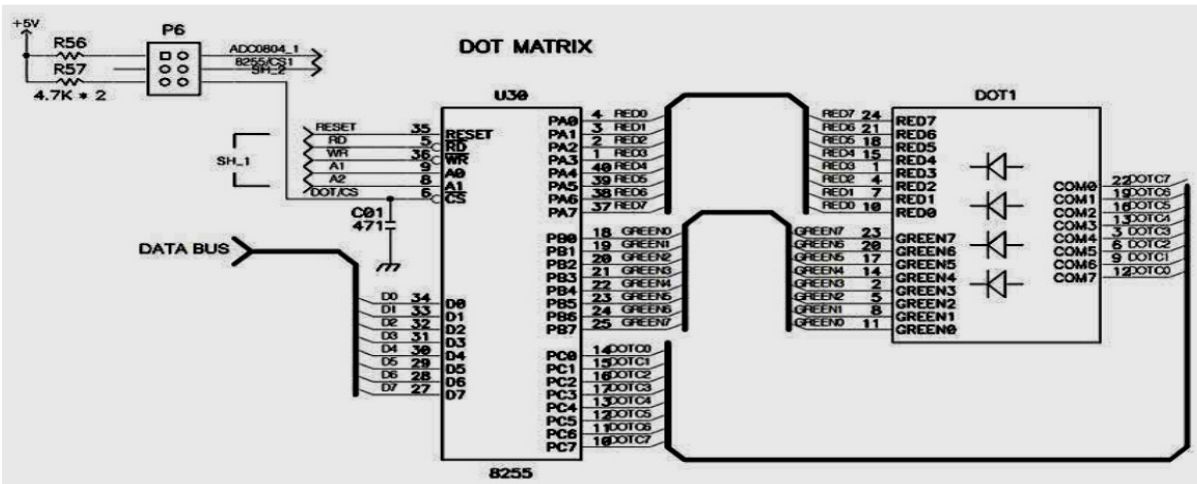
Steps to run code in MDA-8086 through PC:

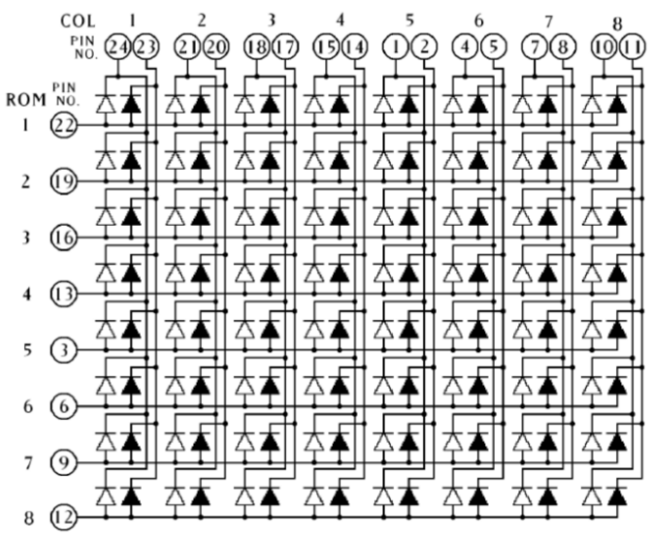
- At first copy paste the .ASM file in the mda folder of computer
- Then open cmd and write cd\ and press enter
- Then type cd mda and press enter
- Then type MASM and press enter
- Then write the file_name.ASM and press enter. For our example we will write L.ASM
- Then write the file_name.OBJ and press enter. For our example we will write L.OBJ

- Then write the file_name. LST and press enter. This step is used for error checking. For our example we will write L.LST
- Then when it wants. CRF file simply press enter
- If there is any error in the file, then after this line we can see the number of errors.
- If any error is found, then type EDIT file_name.LST and press enter.
- If no error is found, then type LOD186 and press enter
- Then type file_name.OBJ and press enter. For our example we will write L.OBJ
- Then type file_name. ABS and press enter. For our example we will write L.ABS
- Then type COMM and press enter.
- Then a blue window will occur
- We will now turn on the kit and we will select PC mode from kit mode
- Then press RESET
- If your kit is ok, then it will show up in the blue screen
- Then type L from keyboard and press enter
- If L does not show up, then it means your PC is not connected and you have to try in different PC
- Otherwise press F3 and in the pop-up screen write filename.ABS and press enter. For our example we will write L.ABS
- Then in the kit select kit mode from PC mode
- Then press RESET
- After that press AD
- Then Press GO
- Then you can see the output in the LED display

Experiment 4: Dot Matrix Display with The Microprocessor Through Peripheral Programmable Interface 82555a

8255A is a general purpose programmable I/O device used in microprocessors. It consists of three 8-bit bidirectional I/O ports with 24 I/O pins (figure 1) which may be individually programmed in 2 groups of 12 pins and used in 3 major modes of operation.





To formulate (8x8) DOTT MATRIX – two (2) color LEDs – including RED and GREEN. Port C HIGH (1) and Port A is LOW (0) glows corresponding RED LED, Port C HIGH (1) and Port B is LOW (0) glows corresponding GRREEN LED.

The address of the control register, port A, port B and port C of the 8255 IC are 1E, 18, 1A and 1C respectively.

Figure 5.3: Circuit diagram of a DOT MATRIX

Task : Understand the basic configuration of 8255 and the DOT matrix.

LEDs are a particular type of diode that converts electrical energy into light. In fact, LED stands for "Light Emitting Diode". The following figure is taken from TechTerms.com and shows the basic forward bias electricity flow from Anode (Positive +) to Cathode (Negative -). If both side of diode have same voltage value (1/0) then no conduction, means no current. However, if potential difference is equal to or greater than threshold (0.7 for germanium) then there will be conduction.

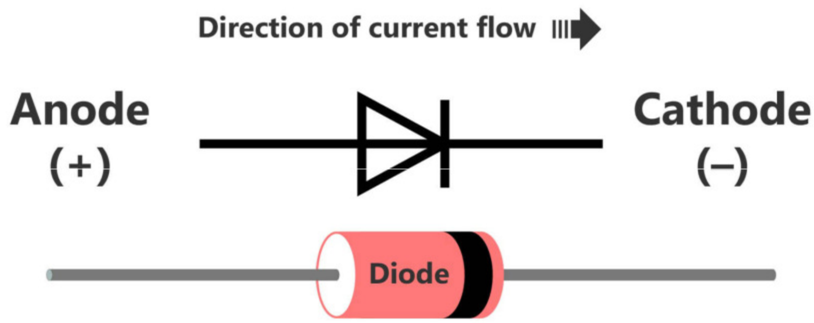


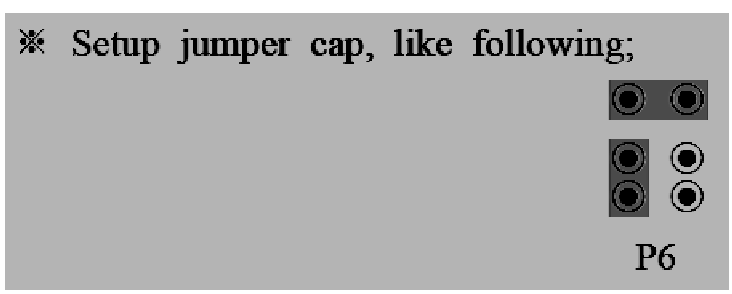
Figure 5.4: Direction of Current Flow

For LEDs in MATRIX: Glow 1st ROW LEDs

You will need to select the column A0/B0 (which means A0/B0 is pulled low), and deselect other columns by blocking their ground paths (by pulling A1/B1 through A7/B7 pins to logic high). Now, the first column is active, and you will need to turn on the LEDs in the rows C0 through C7 of this column, which can be done by applying forward bias voltages (HIGH) to all rows.

Jumper Setup:

You need to set the jumper as shown below before running any program with MDA kit.



Task: Understand the Setting of the Jumper.

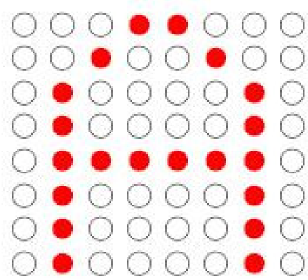
Go to top right corner of DOT MATRIX figure.

Experimental tools:

MDA-Win8086, Computer, Microprocessor emulator Software with Integrated Assembler.

The following rules are needed to perform the lab work.

Task: Run the HEX CODE of the following program to display the letter 'A' on the LED matrix.



```
; PPIC_C     EQU   1EH    ; control register
; PPIC EQU   1CH    ; c port
; PPIB EQU   1AH    ; b port
; PPIA EQU   18H    ; a port

Address: Op code     instructions

                     ORG    1000H
1000:   B0 80        MOV    AL, 10000000B
1002:   E6 1E        OUT    PPIC_C, AL    ; program PPI
                                   ;
1004:   B0 FF        MOV    AL, 11111111B        ; OFF LEDs connected to port A
1006:   E6 18        OUT    PPIA, AL
                                   ;
1008:  BE 2C 10   L1:       MOV   SI, OFFSET FONT
100B:   B4 01        MOV    AH, 00000001B
100D:  2E 8A 04   L2:MOV    AL, BYTE PTR CS:[SI]
1010:   E6 1A        OUT    PPIB, AL
```

```
1012:   8A C4           MOV   AL, AH
1014:   E6 1C           OUT   PPIC, AL
1016:   E8 09 00        CALL TIMER

1019:   46              INC   SI
101A:  F8               CLC
101B:  D0 C4            ROL   AH, 1
101D:  73 EF(EE)        JNC   L2
101F:  EB E7            JMP   L1
1021:  CC               INT   3
1022:   B9 2C 01        TIMER:       MOV  CX, 300
1025:   90              TIMER1: NOP
1026:   90                    NOP
1027:   90                    NOP
1028:   90                    NOP
1029:   E2 FA                 LOOP TIMER1
102B:  C3                     RET
                                    ;
102C:  FFFONT: DB             11111111B
102D:  C0               DB    11000000B
102E:  B7               DB10110111B
102F:  77               DB01110111B
1030:  77               DB01110111B
1031:  B7               DB10110111B

1032:  C0               DB    11000000B
1033:  FF               DB    11111111B
```

Task 4: Write a program of display another letter

Session 6

Session Objective:

- To understand basic theory of digital to analog converter.
- To understand the operation theory and characteristics of DAC0800.
- To understand the connectivity between MDA 8086 board and DAC0800, and Interfacing with 8255.
- To simulate example program (DAC.asm) to trace the Analog voltage changes due to the change in digital input.
- To understand basic theory of Stepper Motor.
- To understand the operation theory and characteristics of Stepper Motor.
- To understand 1-phase, 2-phase and 1-2 phase excitations.

Digital to analog convertor (D/A, DAC) is an electronics device in form of IC, which converts digital signal to its equivalent analog signal. D/A converters are available as integrated circuits. DAC0800 is a cheap and commonly used 8-bit DAC. Internal chip consists of reference voltage power supply (±4.5V to ±18 V), R-2R ladder resistors network and transistor switch. The setting times of around 100ns.

There are two (2) methods of creating a DAC: Binary weighted and R-2R ladder. To achieve higher degree of precision DAC0800 uses R-2R method. DAC resolution is decided by the analog output levels equal to 2n and 2n-1 steps size, where n is the number of inputted data bits. Thus, an 8-input DAC provides 256 discrete voltage or current levels of output.

This is basically a summing amplifier designed with suitable resistances, as shown below.

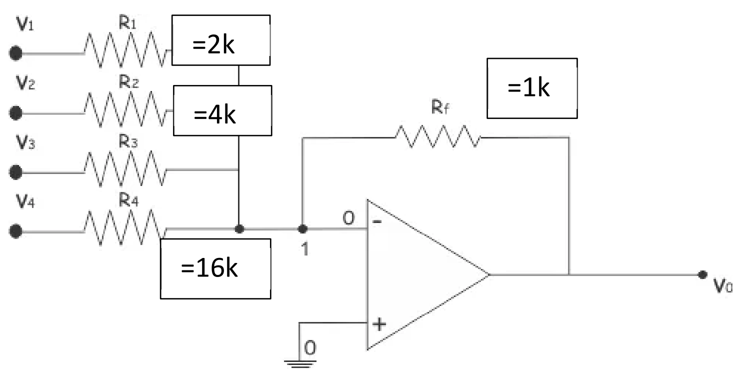


Figure 6.1: Summing Amplifier with Binary Weighted Input Resistors

According to Kirchhoff Current Law and Kirchhoff Voltage Law

I_O=I_T=I_1+I_2+I_3+I_4

=V_in/R1+V_in/R2+V_in/R3+V_in/R4

=V_in/Rf (1/2+1/4+1/8+1/16)

The voltage output is:

V_o=-R_f I_T=|R_f I_T |

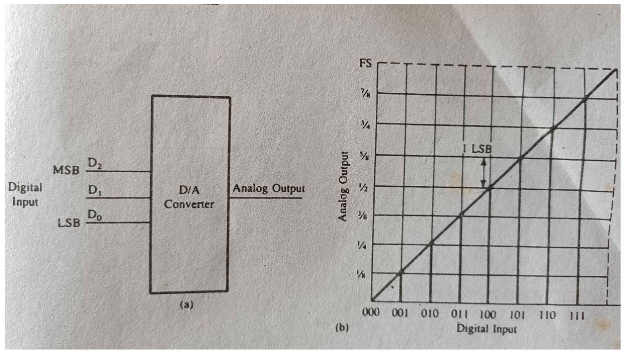


Figure 6.2: A 3 Bit D/A Converter Block Diagram and Digital Input vs Analog Output

Figure 6.2 expresses the property of a 3-bit DAC. Three input lines (D2, D1 and D0) assume 8 input combinations from 000 to 111. D2 is the MSB and D0 is the LSB. If the input range 0V to 1V, it can be divided into eight equal parts (1/8 V) and each successive input is 1/8 V higher than the previous combination, as shown in figure 1. The following points can be summarized from the graph:

- The 3 bits eight possible combinations. If a converter has n input lines, it can have 2n input combinations.
- If the full-scale analog voltage is 1 V, the smallest unit or the LSB (0012) is equivalent to 1/2n of 1V. This is defined as resolution. In this example, the LSB =1/8V.
- The MSB represents the half of the full-scale value. In this example, the MSB (1002) =1/2 V.
- For the maximum input signal (1112), the output signal is equal to the value of the full-scale input signal minus the value of 1 LSB input signal. Thus, the maximum input signal (1112) represents 7/8 V.
- Calculate the values of the LSB, MSB and full-scale output for an 8-bit DAC for the 0 to 10V range.

LSB=1/28=1/256, for 10V LSB=10V/256=39mV

MSB=1/2 full scale=5V

Full Scale Output = Full Scale Value – 1 LSB

= 10V-0.039V

= 9.961 V

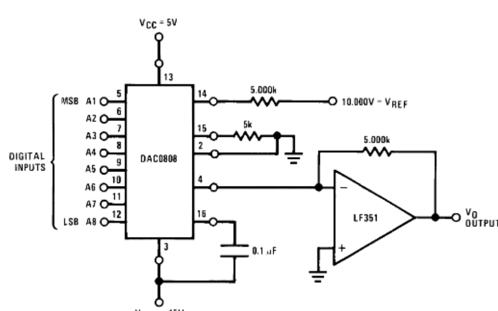


Figure 6.3: Interfacing DAC0808 and LF351

- It needs two power supplies VCC and VEE. VCC (13 pin) is +5V. A negative power supply of -15v (VEE, 3 pin) is required for proper operation of DAC0808. This -15v supply is attached with VEE pin of DAC0808 and also with 4th pin of Op-Amp LF351.
- VREF- pin (15 pin) of DAC0808 is attached with ground through a 5k resistor as specified in DAC0808 datasheet. Also, VREF+ (14 pin) is attached with +10v supply through a 5k resistor. This means that output of DAC can vary from 0v to 10v only. You can increase this reference voltage in order to get more voltage change, for example by attaching +15v with VREF+ pin
- Using this circuit, digital input given to DAC0808 can be converted in to analog output using this formula.

$$I_{out} \approx \frac{V_{ref}}{R_1}\left(\frac{D_7}{2}+\frac{D_6}{4}+\frac{D_5}{8}+\frac{D_4}{16}+\frac{D_3}{32}+\frac{D_2}{64}+\frac{D_1}{128}+\frac{D_0}{256}\right)$$

=10V/5k(255/256) =2mA(255/256)=1.992mA when D0/A8 (LSB)=1, D0/A8=1,....

D7/A1 (MSB)=1

Output Voltage Vo= Rf*Iout=2mA x 5k x255/256= 9.961V

Directly we can write the Vo equation as follows:

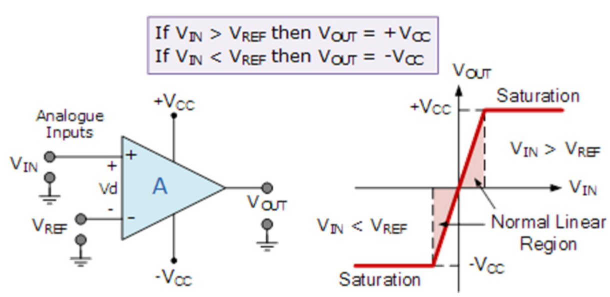
$$V_O = 10V\left(\frac{A1}{2}+\frac{A2}{4}+\ldots\frac{A8}{256}\right)$$

Operational Amplifiers (Op amp) is extensively used as main building block of digital to analog convertor. Op-Amp IC 741 or LM741 is one of the most used operational amplifier integrated circuits that performs both mathematical operations and amplification functions. This small chip mainly performs mathematical operations like addition, subtraction, multiplication, division, differentiation, integration, etc. in various circuits.

The functionality of each pin is as follows:

Power Supply Pins (Pin 4 and Pin 7): Pin 4 and Pin 7 are negative and positive voltage supply terminals respectively. The power required for IC to operate is received from both these pins. The voltage level between these pins can be in the range of 5V to 18V.

Input Pins (Pin 2 and Pin 3): Pin 2 and pin 3 are input pins for the op-amp IC. Pin 2 is considered as inverting input and pin 3 is considered as non-inverting input. When the voltage at pin 2 is greater than the voltage at pin 3, i.e., the voltage at inverting input is higher, then the output signal is low. Similarly, when the voltage at pin 3 is greater than the voltage at pin 2, i.e., the voltage at the non-inverting input is higher, then the output signal is high.



Output Pin (Pin 6): Pin 6 is the output pin of op-amp IC 741. The output voltage at this pin depends on the voltage level on input pins and the feedback approach used. When the voltage at this pin is high, this means that the output voltage is similar to the positive supply voltage. Similarly, when the voltage at this pin is low, this means that the output voltage is similar to the negative output voltage.

D/A Converter Interface

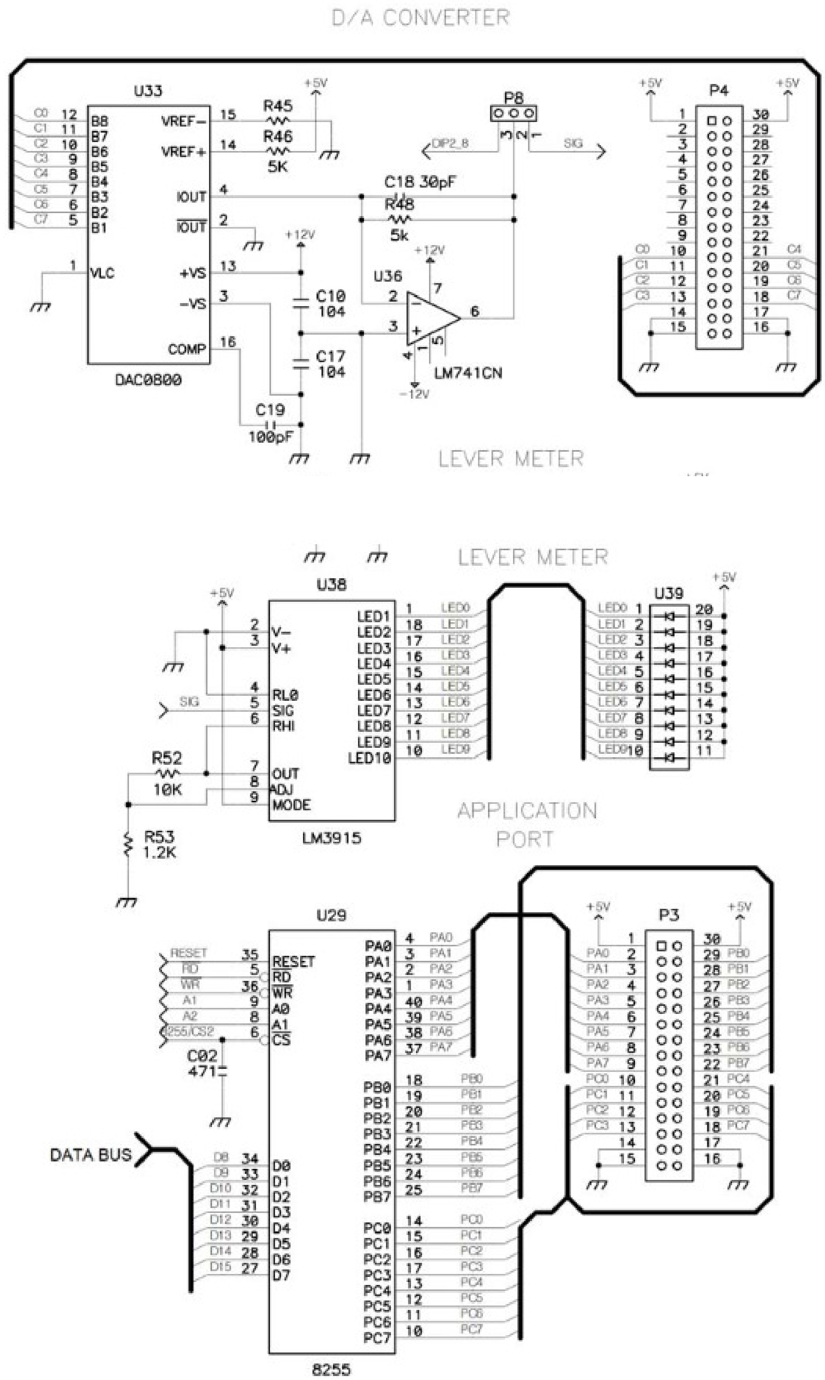


Figure 6.4: Interfacing Level Meter, DAC0808 and 8255

When you increase the amplitude of a sound wave, you are essentially increasing the amount of energy that the wave is carrying, which makes the sound louder. This is because the energy from the wave is distributed over a larger area, which causes the sound to be more intense and louder.

Task2: Run the following sample program to convert the digital input to analog output, and monitor the analog changes in LEVEL METER.

```
                PPIC_C    EQU  1FH
                PPIC EQU  1DH
                PPIB EQU  1BH
                PPIA EQU  19H
                     ;
                     ORG  1000H
B0 80                MOV  AL,10000000B
E6 1F                OUT  PPIC_C,AL
                     ;
B0 FF                MOV  AL,11111111B
E6 19                OUT  PPIA,AL
B0 F0                MOV  AL,11110000B
E6 1B                OUT  PPIB,AL
                     ;
B0 00           L2:  MOV  AL,00000000B
E6 1D           L1:  OUT  PPIC,AL
09 00                CALL TIMER
FE  C0               INC  AL
3C 50                CMP  AL,50H  #80 inputs 80 vol into 10 levels
75 F5                JNE  L1
EB F1                JMP  L2
                     ;
CC                   INT  3
                     ;
B9 01 00        TIMER:    MOV  CX,1
51              TIMER2:   PUSH CX
B9 00 00                  MOV  CX,0
90              TIMER1:   NOP
E2 FD                LOOP TIMER1
59                   POP  CX
E2 F6                LOOP TIMER2
C3                   RET
```

Experiment 2: Stepper Motor

In general, a motor is a device that transforms electrical energy to mechanical energy. It is a synchronous electric motor capable of dividing a complete rotation into many steps. Stepper motors typically consist of a permanent magnet shaft (rotor) encircled by a stator. As long as the motor is not large, the angular position of the motor can be precisely regulated without the use of a feedback device. As a result, it operates in a simple precise open-loop system in which the output is directly proportional to the input.

Permanent Magnet (PM) Stepper Motors consist of permanent magnet rotors with no teeth, which are magnetized perpendicular to the axis of rotation. By energizing the four phases (in sequence), the rotor rotates due to the attraction of magnetic poles. The stepper motor shown in the diagram below will take 90-degree steps as the windings are energized by passing electricity between the coils of the stator in clockwise sequence: X, X', Y, Y' to complete a revolution. Energized a particular phase by DC current will create N pole of the stator (the source part of electricity) and S pole to the other wise, which interact the N pole of the rotor. 45-degree steps are created by energizing consecutive two phases. As example, S_A and S_B creates 45-degree steps rotate will point to the middle of A and B phases. Anti-clockwise rotation will be created by energized the stators from opposite direction.

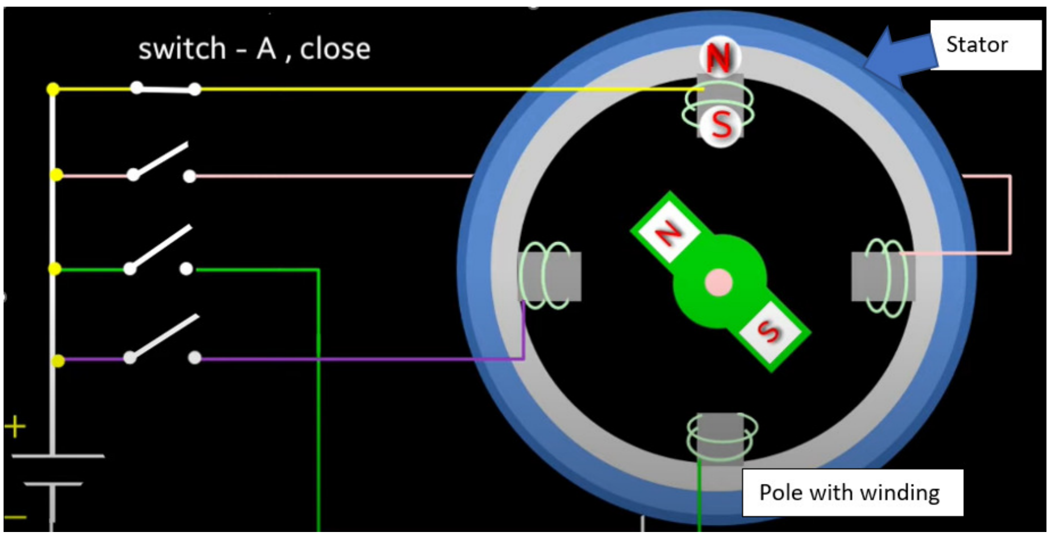
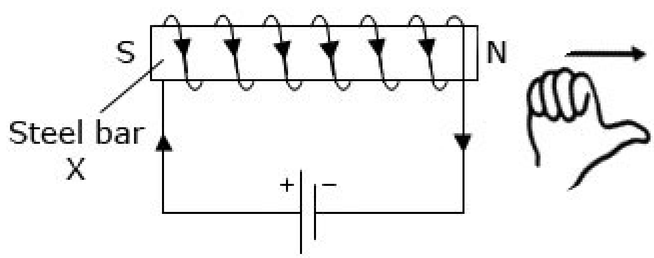


Figure 6.5: Stepper Motor and its Working Strategy



Since stepping motor makes step- by-step movement and each step is equidistant, the rotor and stator magnetic field must be synchronous. During start-up and stopping, the two fields may not be synchronous, so it is suggested to slowly accelerate and decelerate the stepping motor during the start-up or stopping period.

Figure 6.6 is used to explain the operation of simplified stepping motor (90°/step). Here the A coil and B coil are perpendicular to each other. If either A or B coil is excited (a condition which is known as single-phase excitation), the rotor can be moved to 0°, 90°, 180°, 270°-degree position depending on the current's ON/OFF conditions in the coils, see Figure 6.6(a). If both coils have current flowing at the same time, then the rotor positions can be 45°, 135°, 225°, 315°degrees as shown in Figure 6.6(b). This is known as two-phase exception. In Figure 6.6(c), the excitation alternates between 1-phase and 2-phase, then the motor will rotate according to 0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°sequence. This is 1-2 phase excitation; each step distance is only half of step movement of either 1-phase or 2-phase excitation. Stepping motor can rotate in clockwise or counter-clockwise direction depending on the current pulse sequence applied to the excitation coils of the motor. Referring to the truth tables in Figure 6.6(a), (b), (c). If signals are applied to coil A and B according to Step 1,2,3,4,5,6,7,8, then counter-clockwise movement is achieved. And vice-versa is true. If signals are applied according to step 8,7,6,5,4,3,2,1, then clockwise movement is achieved.

Session 7

Session Objective:

The students will give a formal presentation where they present their group project which they proposed in Session 4, to be built based on Arduino and various sensors. They will also submit a project report at this time.

Guidelines of the Presentation:

- The students must keep the following points in their report as well as their presentations, in addition, they may keep other points they deem necessary:
    - objectives
    - social values
    - required components
    - working procedure
    - budget comparison
    - contribution of team-members
    - challenges of the project
    - conclusion
- Each group will get 20 minutes to present.
- Every group member must provide a part of the presentation. If someone does not present, they will not receive marks.
- The presentation should follow the same points as the project report (such as: - objectives, social values, required components etc.) but students must not just copy-paste everything word-for-word from the report. They should make it more concise. Teachers expect not to see lengthy descriptions in the presentation slides.
- There will be a Q&A portion at the end of each presentation, where students will be asked questions regarding their project, so they must be prepared to defend their choice of project as well as having a clear understanding of your project.
- Presentation language: English

THE END