WELCOME TO ALL
MY PRESENTATION

Microcontroller Application

Sub Code: 66663

T	C	P
0	2	6

Computer Technology

6^{th} Semester

BulBul Ahamed
Chief Instructor & Head of the Department(Computer)
Mymensingh Polytechnic Institute.

AN INTRODUCTION TO MICROCONTROLLERS

What Is A Microcontroller?

- A microcontroller is an integrated circuit that is programmed to do a specific task.

- Microcontrollers are really just “mini-computers”.

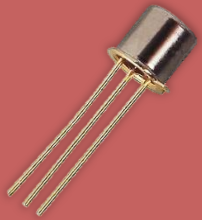
Where do you find them?

- Microcontrollers are hidden in tons of appliances, gadgets, and other electronics.

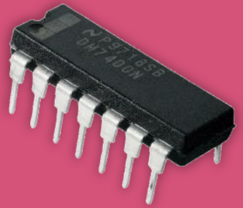
- They’re everywhere!

History of Microcontrollers

Vacuum Tube
1939



Transistor
1947



Logic Gate
1960

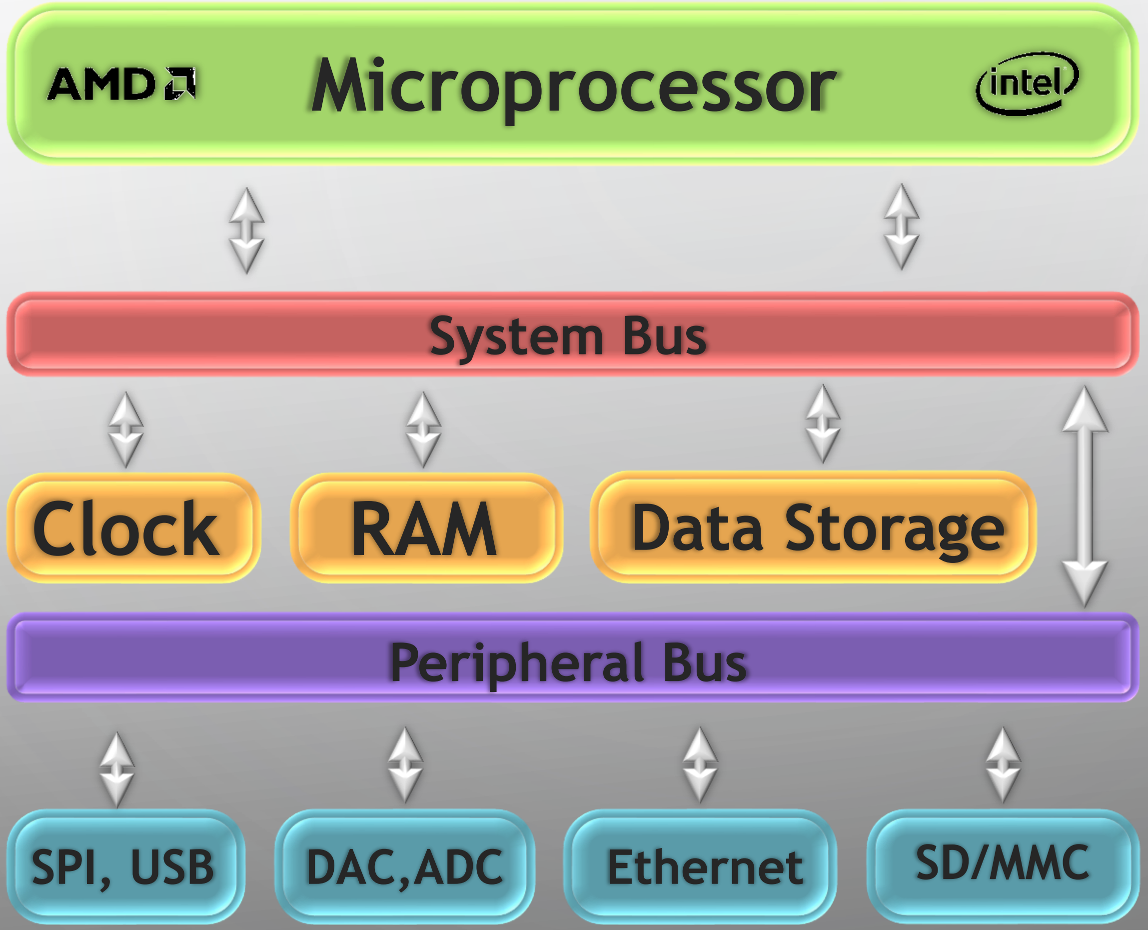
Microcontroller
1971

Microprocessor –vs- Microcontroller

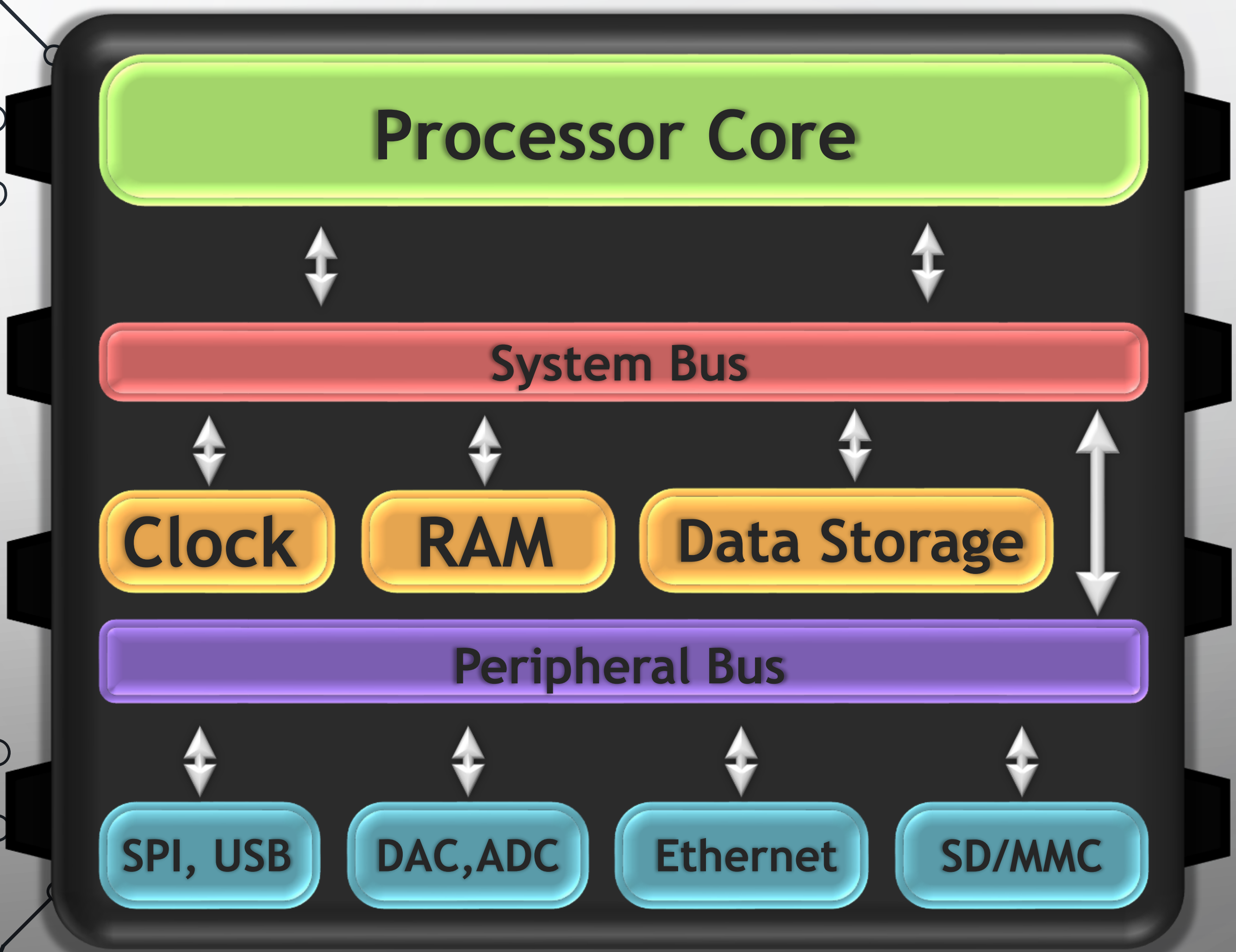
- You may have heard of the term "microprocessor" or just "processor" before. You may ask, "Is there a difference between a microprocessor and microcontroller?"

- Yes there is, they are two different things!

	MICROPROCESSOR	MICROCONTROLLER
Applications	General Computing (I.E. Laptops, Tablets)	Appliances, Specialized Devices
Speed	Very Fast	Relatively Slow
External Parts	Many	Few
Cost	High	Low
Energy Use	Medium To High	Very Low To Low
Vendors	intel, AMD, ARM	ATMEL, ST, Texas Instruments, Microchip



Microprocessor



Microcontroller

Basic Principles of Operation

- Microcontrollers are used for specific applications.
- They do not need to be powerful because most applications only require a clock of a few MHz and small amount of storage.
- A microcontroller needs to be programmed to be useful.
- A microcontroller is only as useful as the code written for it. If you wanted to turn on a red light when a temperature reached a certain point, the programmer would have to explicitly specify how that will happen through his code.

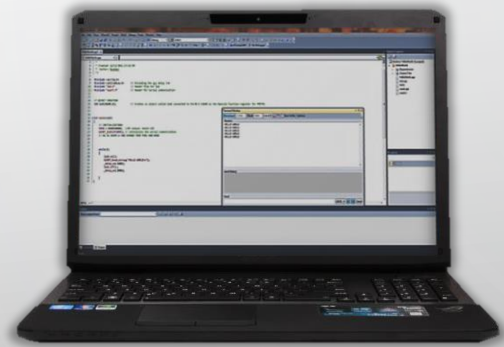
Microcontroller Programming

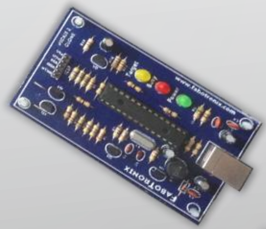
1.) Code is written for the microcontroller in an integrated development environment, a PC program. The code is written in a programming language. (e.g. C, BASIC or Assembly).

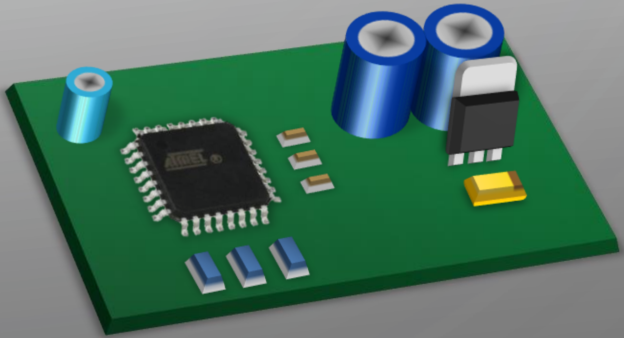
2.) The IDE debugs the code for errors, and then compiles it into binary code which the microcontroller can execute.

3.) A programmer (a piece of hardware, not a person) is used to transfer the code from the PC to the microcontroller. The most common type of programmer is an ICSP (In-circuit serial programmer).

Microcontroller Programming







Voila!

The Analog to Digital Converter (ADC)

- Just about every modern microcontroller contains an ADC(s).

- It converts analog voltages into digital values.

- These digital representations of the signal at hand can be analyzed in code, logged in memory, or used in practically any other way possible.

The Digital to Analog Converter (DAC)

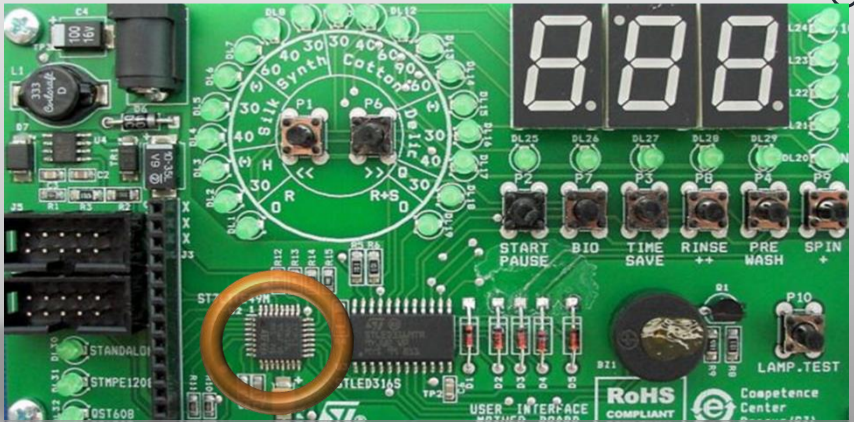
- You guessed it! Microcontrollers have accompanying DACs.

- It does exactly the opposite function of an ADC. It takes a digital value and converts it into an pseudo-analog voltage.

- It can be used to do an enormous amount of things. One example is to synthesize a waveform. We can create an audio signal from a microcontroller. Imagine that!

Microcontroller Applications

● This is the controller board for a washing machine. If a button is pushed or if a knob is turned, the microcontroller knows how to react to the event.

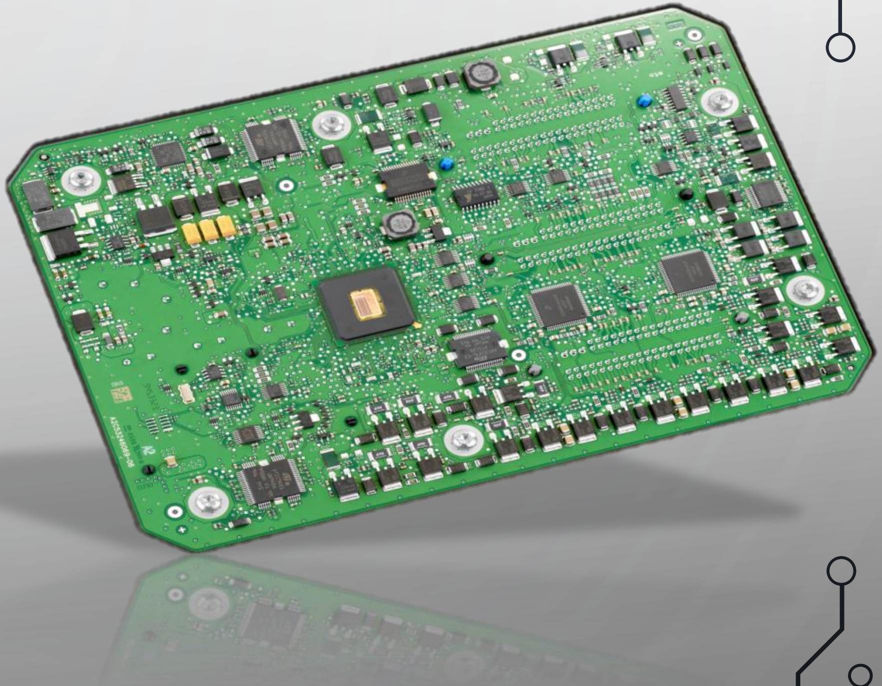


● Ex. If “start” is pushed, the microcontroller knows to switch a relay which starts the motor.

Microcontroller Applications

- This is the main controller from a Buick Regal. This board has several microcontrollers each for a specific task.

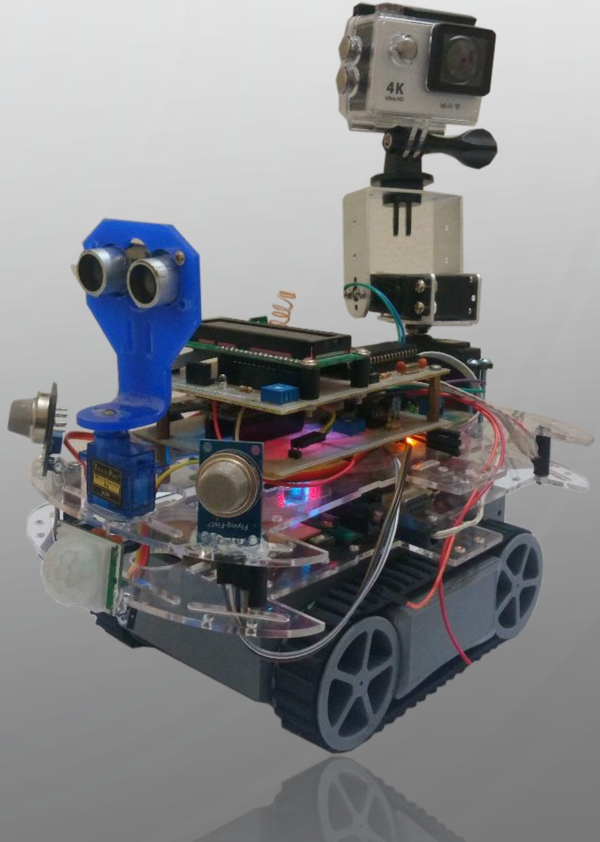
- Ex. A microcontroller may handle dashboard controls or it may even control something more complex like the ignition system.



Microcontroller Applications

- Many robots use microcontrollers to allow robots to interact with the real world.

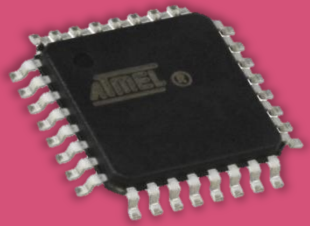
- Ex. If a proximity sensor senses an object near by, the microcontroller will know to stop its motors and then find an unobstructed path.



Microcontroller Packaging

DIP
(Dual Inline Package)
Through hole
8 pins
9mm x 6mm
0.15pins/mm²

SOIC
(Small Outline IC)
Surface Mount
18 pins
11mm x 7mm
0.23pins/mm²

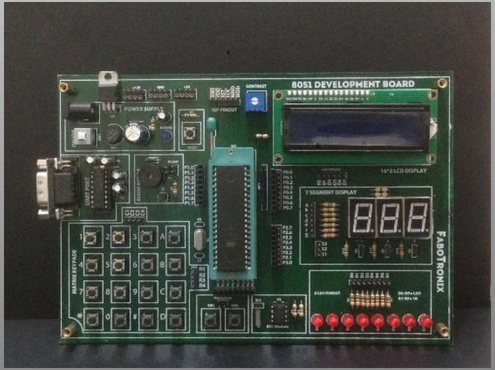


QFP
(Quad Flat Package)
Surface Mount
32 pins
7mm x 7mm
0.65pins/mm²

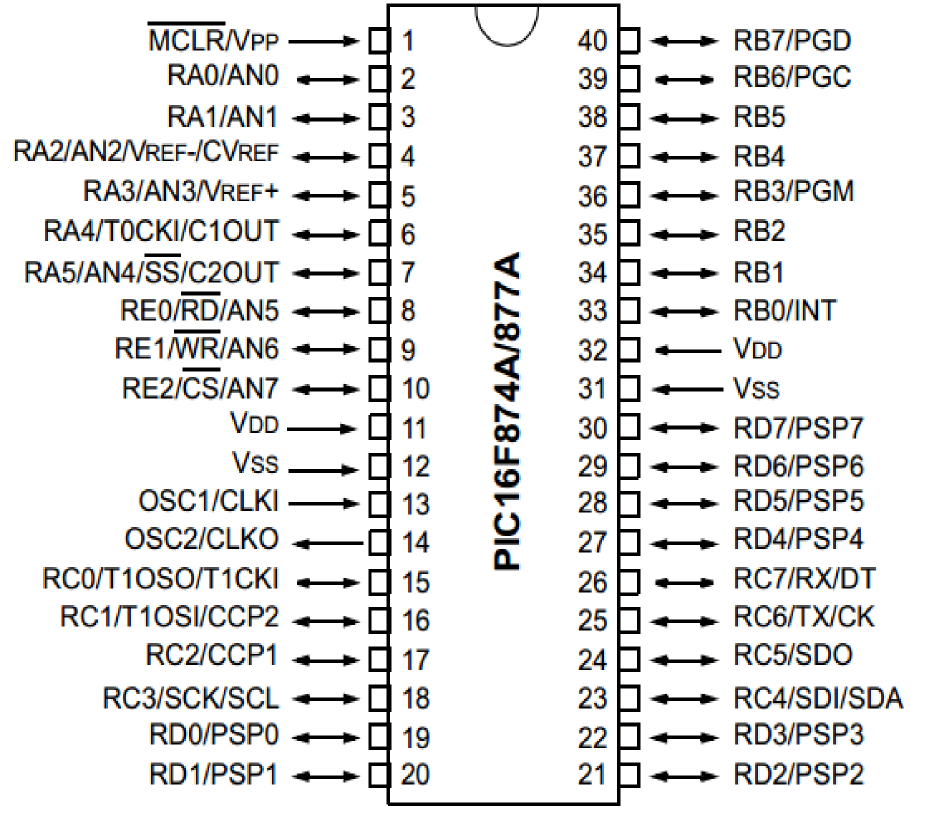
BGA
(Ball Grid Array)
Surface Mount
100 pins
6mm x 6mm
2.78pins/mm²

How can I get started?

- If you want to develop for microcontrollers, you can purchase a development board which includes a microcontroller and all of the necessary parts to get it working. (i.e. power supply and a USB interface)



MICROCONTROLLER PIN DIAGRAM



Microcontroller Basic Hardware Designing

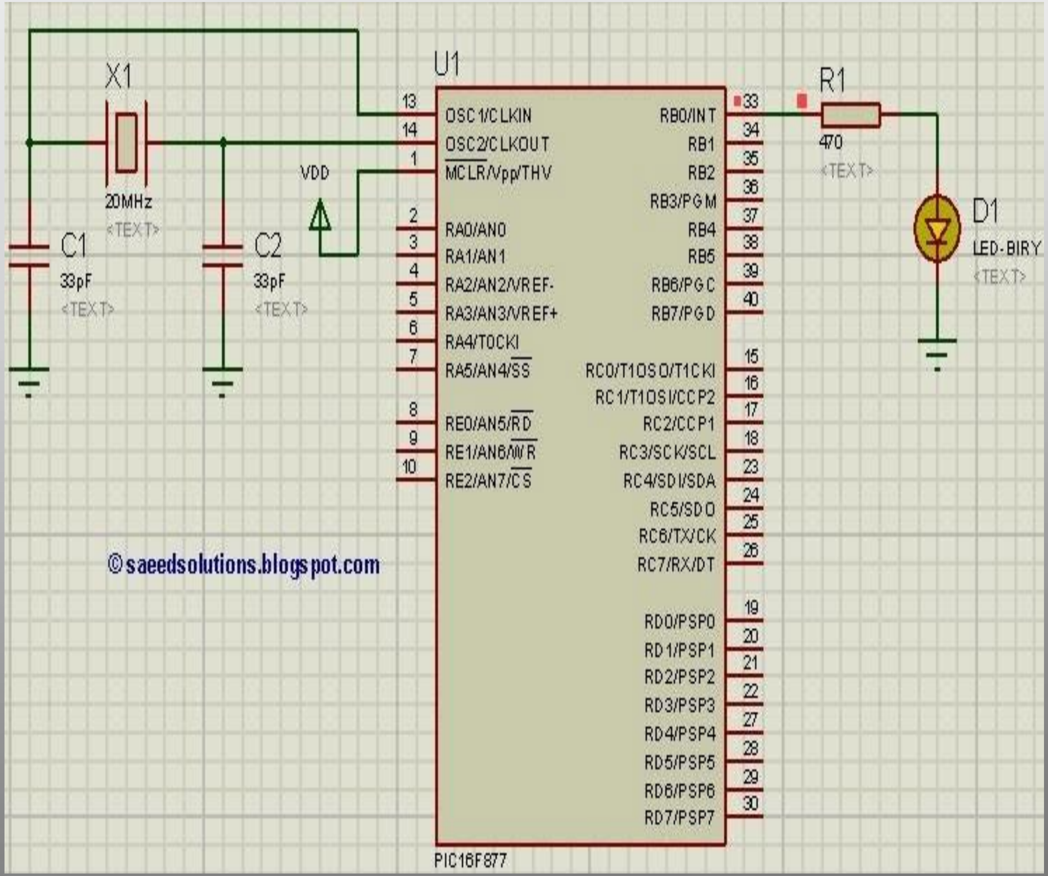
Basics of Hardware Programming

Introduction to MIKRO C Pro for PIC

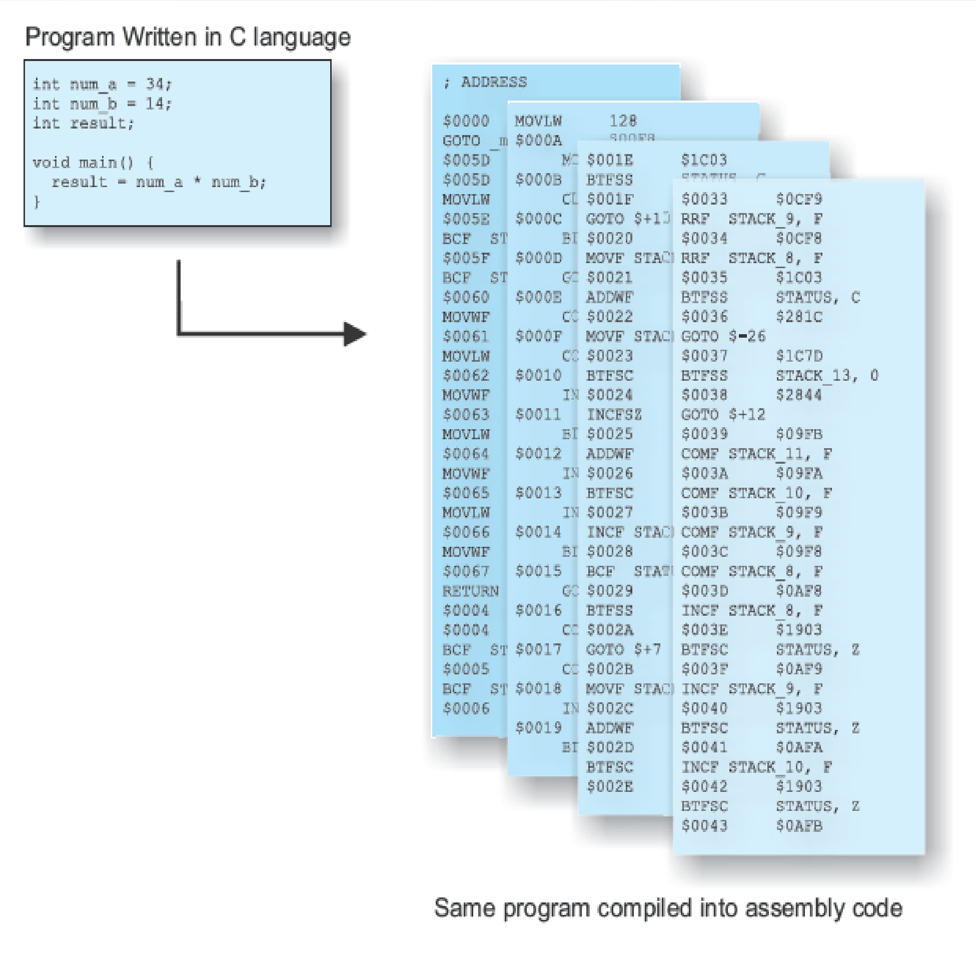
Embedded Systems

Has Two Parts:

- Hardware
- Software
  - Low Level Language
  - High Level Language



Assembly Language Vs C Language



Example for C Language

```
void main ()
{
PORTD = 0x00;                    // load 00000000 to portb
TRISD = 0x00;          // load 00000000 to trisb to make portb as output port
        while(1)                                    // infinity loop
        {
        PORTD = 0B11111111;           // load 00000001 to portb
        delay_ms(500);                        // wait 500ms
        PORTD = 0B00000000;          // load 00000010 to portb
        delay_ms(500);                        // wait 500ms
        }
    }
```

DATA TYPES

DATA TYPE	DESCRIPTION	SIZE (NUMBER OF BITS)	RANGE OF VALUES
char	Character	8	0 to 255
int	Integer	16	-32768 to 32767
float	Floating point	32	from ±1.17549435082 $x10^{-38}$ to ±6.80564774407 $x10^{38}$
double	Double precision floating point	32	from ±1.17549435082 $x10^{-38}$ to ±6.80564774407 $x10^{38}$

Data Types Cont...

DATA TYPE	DATA TYPE WITH PREFIX	SIZE (NUMBER OF BITS)	RANGE
char	signed char	8	-128 to 128
int	unsigned int	16	0 to 65535
	short int	8	0 to 255
	signed short int	8	-128 to 127
	long int	32	0 to 4294967295
	signed long int	32	-2147483648 to 2147483647

Declaring Variable

- Variable name can include any of the alphabetical characters A-Z (a-z), the digits 0-9 and the underscore character '_'.
- Variable names must not start with a digit.
- Some of the names cannot be used as variable names as already being used by the compiler itself.

Example:

```
unsigned int gate1;
signed int start, sum;
gate1 = 20;
```

OPERATORS

OPERATOR	OPERATION
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Reminder

ASSIGNMENT OPERATORS

OPERATOR	EXAMPLE	
	Expression	Equivalent
+=	a += 8	a = a + 8
-=	a -= 8	a = a - 8
*=	a *= 8	a = a * 8
/=	a /= 8	a = a / 8
%=	a %= 8	a = a %8
++	++a	Variable "a" is incremented by 1
	a++	
--	--b	Variable "b" is decremented by 1
	b--	

RELATIONAL OPERATORS

OPERATOR	MEANING	EXAMPLE	TRUTH CONDITION
>	is greater than	b > a	if **b** is greater than **a**
>=	is greater than or equal to	a >= 5	If **a** is greater than or equal to **5**
<	is less than	a < b	if **a** Is less than **b**
<=	is less than or equal to	a <= b	if **a** Is less than or equal to **b**
==	is equal to	a == 6	if **a** Is equal to **6**
!=	is not equal to	a != b	if **a** Is not equal to **b**

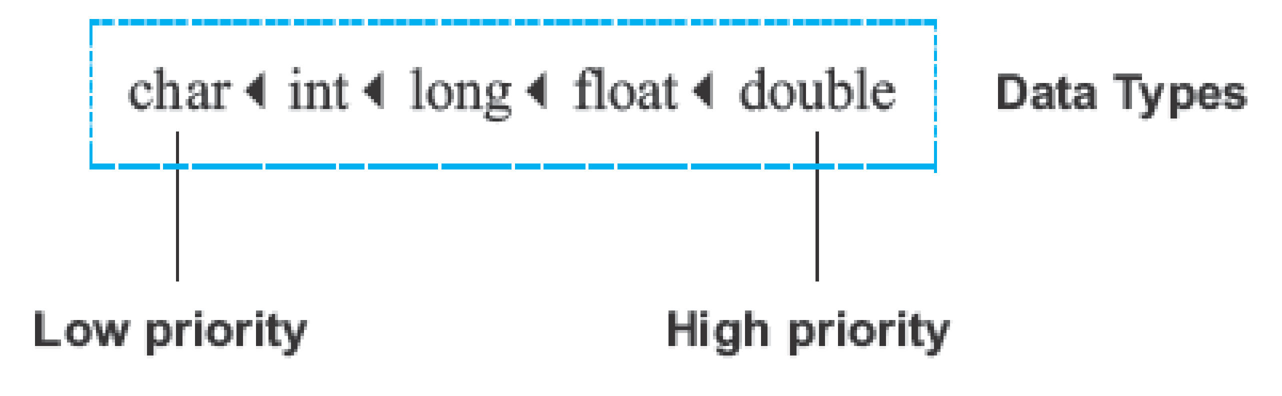
LOGICAL OPERATOR

OPERATOR	LOGICAL AND (&&)		
&&	Operand 1	Operand2	Result
	0	0	0
	0	1	0
	1	0	0
	1	1	1

OPERATOR	LOGICAL OR		
||	Operand1	Operand2	Result
	0	0	0
	0	1	1
	1	0	1
	1	1	1

OPERATOR	LOGICAL NOT	
!	Operand1	Result
	0	1
	1	0

DATA TYPE CONVERSION



BASIC PIN CONFIGURATION

- TRISB= 0 For Output & 1 For Input
- PORTB= 0x00 / 0b00000000 sets all the bits in the port to 0
- PORTB.F1= 1; Sets the port B1 to 1;

LED FLASHING

```
void main() {
TRISB= 0b00000000;
PORTB= 0b00000000;
while(1){
        PORTB.F0 =1;
        delay_ms(2000);
        PORTB.F0= 0;
        PORTB.F1= 1;
        delay_ms(2000);
        PORTB.F1= 0;
        PORTB.F2= 1;
        delay_ms(2000);
        PORTB.F2= 0;
        PORTB.F3 = 1;
        delay_ms(2000);
        PORTB.F3= 0;
        }
}
```

INTRODUCTION TO HARDWARE PROGRAMMING II

ARRAY, CONDITIONAL STATEMENT AND LOOPS

ARRAYS

- A group of variables of the same type is called an array
- Example:

```
short int led[5];

int a;

led[] = (5, 4, 6, 2, 7, 9);

a = led[3];
```

- Two Dimensional Array

```
char super[3][2];

array_type array_name [rows][columns];
```

led[0]	5
led[1]	4
led[2]	6
led[3]	2
led[4]	7
led[5]	9

CONDITIONAL STATEMENT – IF & IF-ELSE

Example:

```
int a =  0;
While(1){
        If (a ==5){
                a=0;
                }
        a++;
       }
```

CONDITIONAL STATEMENT – IF & IF-ELSE CONT'D

```
If (PORTD.F1==0){
                PORTB= 0b11111111;
                delay_ms(500);
                PORTB= 0b00000000;
                delay_ms(500);
                }
else {
    PORTB= 0b00001111;
    delay_ms(500);
    PORTB= 0b11110000;
    delay_ms(500);
    }
```

CONDITIONAL STATEMENT - SWITCH

```
temp=0
    ;
while(1)
    {
        switch (temp){
                        case1:{PORTB =  0b11111111 ;
                               delay_ms(500);
                               PORTB=
                        0b00000000);}   case2:PORTB =
                        0b11110000;
                        case3: {delay_ms(500);
                               PORTB= 0b00001111;}
                        case4: temp=0
```

WHILE LOOP

While(expression){

…..

……

}

```
int a = 1;
while ( a < 4 )
{
printf ( "Hello World\n" );
a ++;
}
```

Output

codesdope.com

FOR LOOP

- for(initial_expression; condition_expression; change_expression) { operations
- ...
- }
- Example

```
for(k=1; k<5; k++) {                //Increase variable k 5 times (from 1 to 5) and
                operation          / /   repeat expression operation every time
                 ...
                }
```

DO - WHILE LOOP

```
do{
    operation
    }while (check_condition);


Example:
a = 0; // Set initial value do{
    a = a+1; // Operation in progress
    PORTD.F0 = 1;
    delay_ms(500); PORTD.F0=0;
    delay_ms(500);
   }while (a <= 25); // Check condition
```

Decimal (Base 10)	Binary (Base 2)	Hexadecimal (Base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

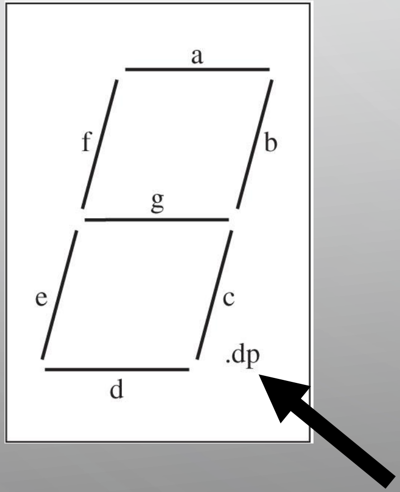
7 SEGMENT DISPLAY

INTERFACING WITH PIC MICROCONTROLLER

SEVEN SEGMENT DISPLAY

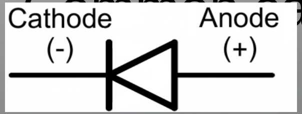
- A seven-segment display is a set of seven bar-shaped LED
- Arranged to form a squared-off figure 8
- Seven segment displays can only display 0 to 9 numbers , decimal “.” and some letters.

TYPES OF SEVEN SEGMENT DISPLAY

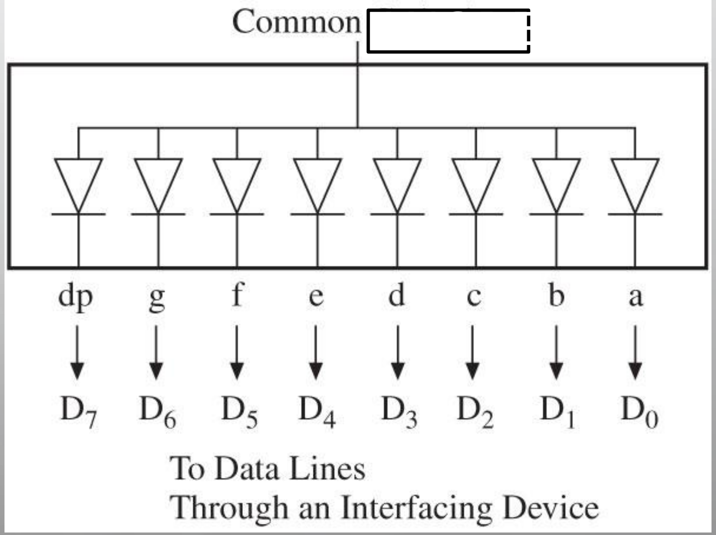


decimal point

- Two types of seven-segment Display
  - Common anode
  - Common cathode

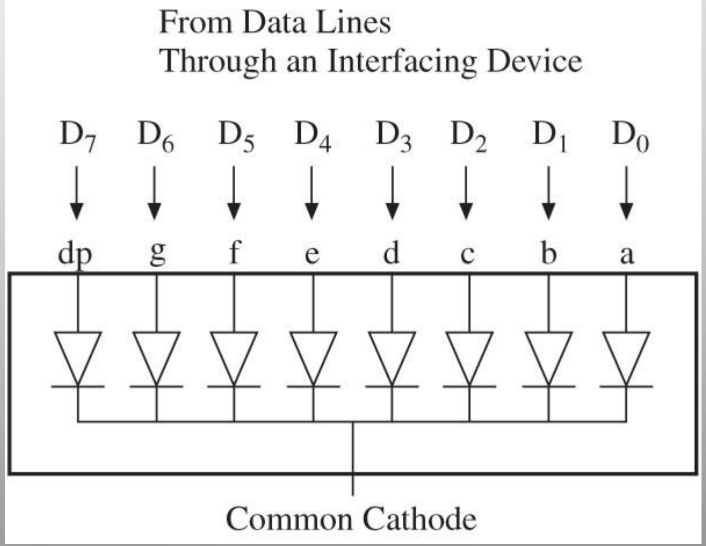


COMMON ANODE

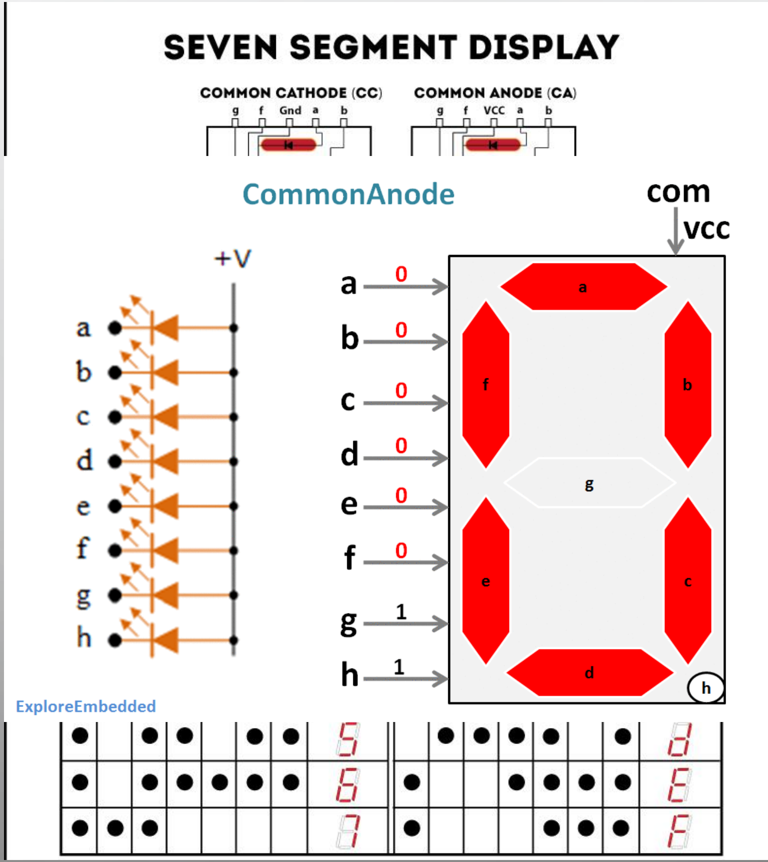


- In a common anode seven-segment LED
  - All anodes are connected together to a power supply and cathodes are connected to data lines
- Logic 0 turns on a segment.
- Example: To display digit 1, all segments except b and c should be off.
- Byte 11111001 = F9H will display digit 1.

COMMON CATHODE



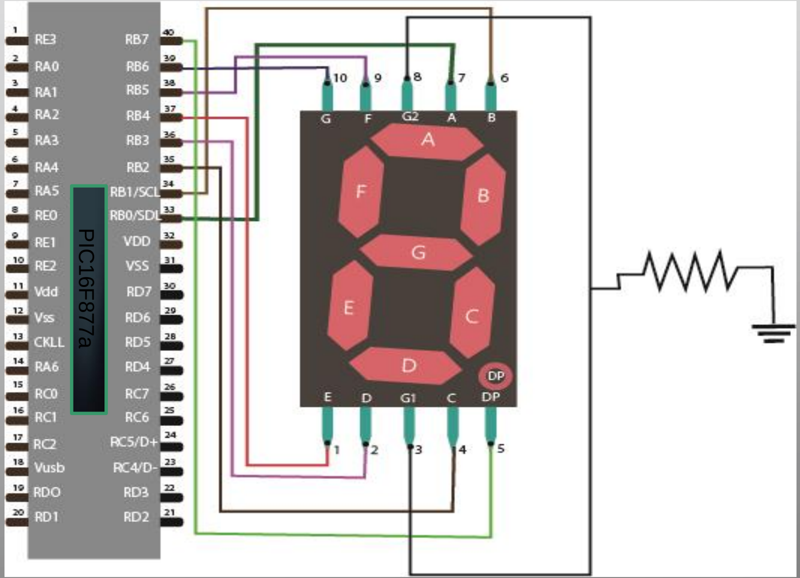
- In a common cathode seven-segment LED
  - All cathodes are connected together to ground and the anodes are connected to data lines
- Logic 1 turns on a segment.
- Example: To display digit 1, all segments except b and c should be off.
- Byte 00000110 = 06H will display digit 1.



SEVEN SEGMENT DISPLAY
COMMON CATHODE (CC)
g f Gnd a b
COMMON ANODE (CA)
g f VCC a b
CommonAnode
com
vcc
+V
a b c d e f g h
a 0
b 0
c 0
d 0
e 0
f 0
g 1
h 1
a b c d e f g h
ExploreEmbedded

7-SEGMENT DISPLAY INTERFACING WITH PIC MICROCONTROLLER

Segment	Port Connection
A	RB0
B	RB1
C	RB2
D	RB3
E	RB4
F	RB5
G	RB6
DP	RB7



EXAMPLE CODE FOR SINGLE 7 SEGMENT

COMMON CATHODE

```
void main() {
    trisb=0x00;
    while(1)
    {
    portb=0b00111111;
    delay_ms(500);
    portb=0b00000110;
    delay_ms(500);
    portb=0b01011011;
    delay_ms(500);
    portb=0b01001111;
    delay_ms(500);
    portb=0b01100110;
    delay_ms(500);
    portb=0b01101101;
    delay_ms(500);
    portb=0b01111101;
    delay_ms(500);
    portb=0b00000111;
    delay_ms(500);
    portb=0b11111111;
    delay_ms(500);
    portb=0b01101111;
    delay_ms(500);
    }
}
```

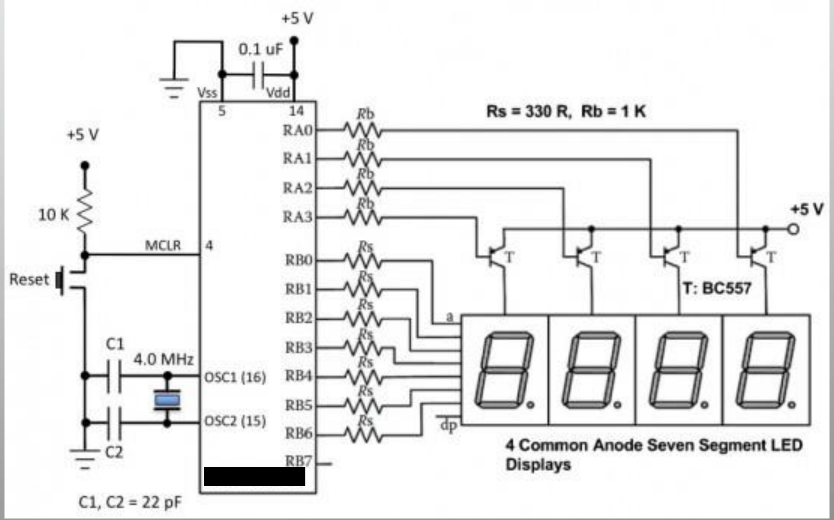
EXAMPLE CODE FOR SINGLE 7 SEGMENT

```
int main() {
    char seg_code[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
    int i;

    /* Configure the ports as output */
    TRISB = 0x00;

    while (1)
    {
        for (i = 0; i <= 9; i++) // loop to display 0-9
        {
            PORTB = seg_code[i];
            delay_ms(100);
        }
    }
}
```

MULTIPLEXING WITH 7 SEGMENT



EXAMPLE OF MULTIPLEXING CODING

```c
int main() {
    char seg_code[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
                     0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e};
    int cnt, num, temp,i;

    /* Configure the ports as output */
    TRISB = 0x00; // Data lines
    TRISD = 0x00; // Control signal PORTD0-PORTD3

    while (1)
    {
        for (cnt = 0x00; cnt <= 9999; cnt++) // loop to display 0-9999
        {
            for (i = 0; i < 100; i++)
            {
                num = cnt;
                temp = num / 1000;
                num = num % 1000;
                PORTD = SegOne;
                PORTB = seg_code[temp];
                DELAY_ms(1);

                temp = num / 100;
                num = num % 100;
                PORTD = SegTwo;
                PORTB = seg_code[temp];
                DELAY_ms(1);

                temp = num / 10;
                PORTD = SegThree;
                PORTB = seg_code[temp];
                DELAY_ms(1);

                temp = num % 10;
                PORTD = SegFour;
                PORTB = seg_code[temp];
                DELAY_ms(1);
            }
        }
    }
```

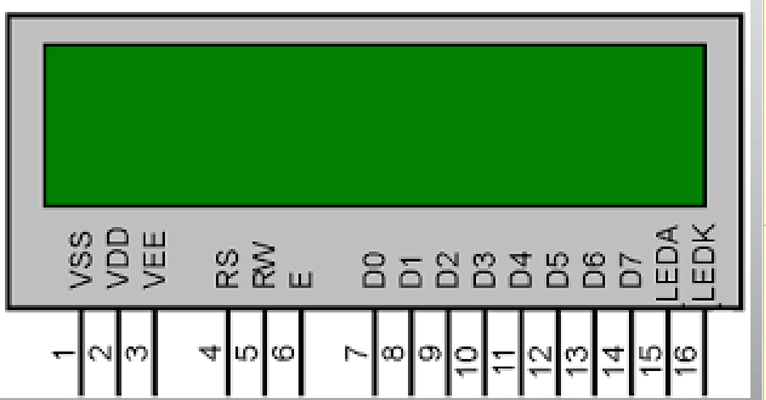
LCD INTERFACING WITH PIC MCU

16X2 LCD, 20X4 LCD

INTERFACING

- Can Use 4 bit or 8 bit
- 8 bit slightly faster
- Use 4 bit becaus not much faster and saves IO pins

LCD DISPLAY PIN CONFIG



Function	Pin Number	Name	Logic State	Description
Ground	1	Vss	–	0V
Power supply	2	Vdd	–	+5V
Contrast	3	Vee	–	0V to +5V
Control of operation	4	RS	0	D0-D7 are interpreted as commands
			1	D0-D7 are interpreted as data
	5	R/W	0	Write data (from microcontroller to LCD)
			1	Read data (from LCD to microcontroller)
	6	E	0	Access to LCD disabled
			1	Normal operation
			From 1 to 0	data/ commands are sent to LCD
Data/ commands	7	D0	0/1	Bit 0 LSB
	8	D1	0/1	Bit 1
	9	D2	0/1	Bit 2
	10	D3	0/1	Bit 3
	11	D4	0/1	Bit 4
	12	D5	0/1	Bit 5
	12	D6	0/1	Bit 6
	14	D7	0/1	Bit 7 MSB

DEFINING LCD CONNECTIONS

```
// LCD module connections
sbit LCD_RS at RC2_bit;
sbit LCD_EN at RC3_bit;
sbit LCD_D4 at RC4_bit;
sbit LCD_D5 at RC5_bit;
sbit LCD_D6 at RC6_bit;
sbit LCD_D7 at RC7_bit;
sbit LCD_RS_Direction at TRISC2_bit;
sbit LCD_EN_Direction at TRISC3_bit;
sbit LCD_D4_Direction at TRISC4_bit;
sbit LCD_D5_Direction at TRISC5_bit;
sbit LCD_D6_Direction at TRISC6_bit;
sbit LCD_D7_Direction at TRISC7_bit;
// End LCD module connections
```

LIBRARY FUNCTIONS

- **Lcd_Init**

  Prototype : *void Lcd_Init();*

  This function initializes the LCD module connected to the above defined pins of the PIC Microcontroller.

- **Lcd_Out**

  Prototype : *void Lcd_Out(char row, char column, char *text);*

  This functions prints the text (string) in a particular row and column.

- **Lcd_Out_Cp**

  Prototype : *void Lcd_Out_Cp(char *text);*

  This function prints the text (string) in the current cursor position. When we write data to LCD Screen, it automatically increments the cursor position.

- **Lcd_Chr**

  Prototype : *void Lcd_Chr(char row, char column, char out_char);*

  It prints the character (out_char) in the specified row and column of the LCD Screen.

- **Lcd_Chr_Cp**

  Prototype : *void Lcd_Chr_Cp(char out_char);*

  It prints the character (out_char) in the current cursor position.

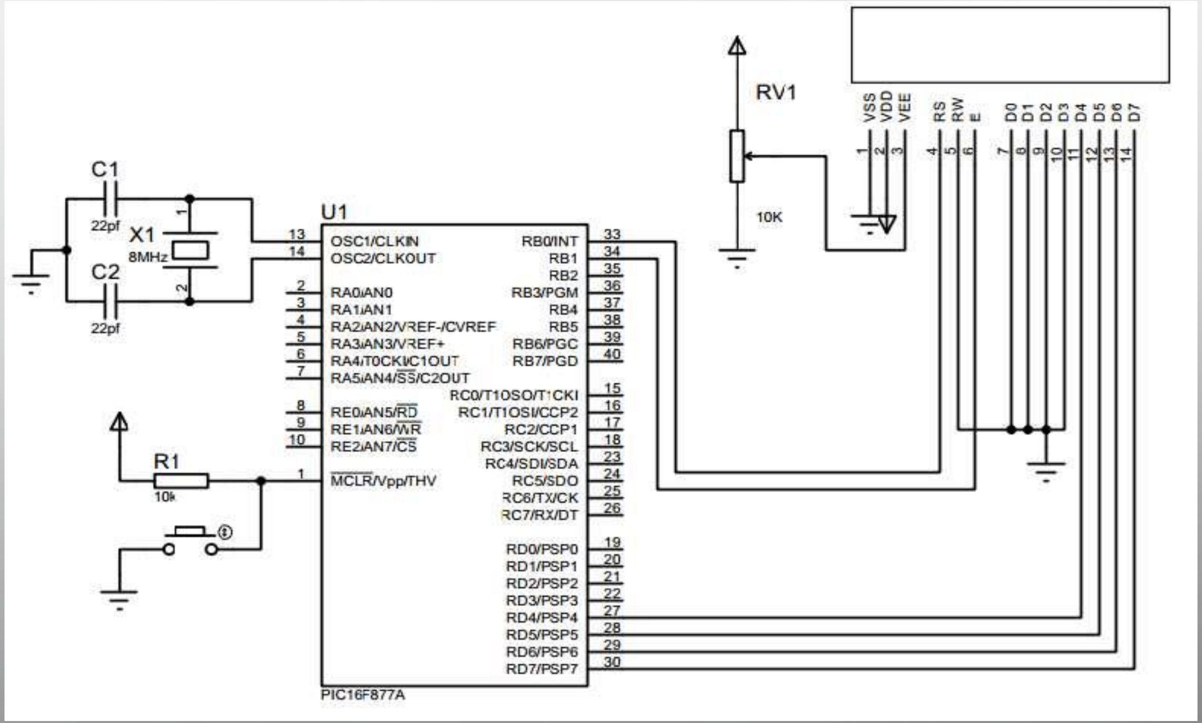
LIBRARY FUNCTIONS CONT'D

Lcd_Cmd

Prototype : *void Lcd_Cmd(char out_char);*

This function is used to send commands to LCD. You can use any one of the following constants as command.

- _LCD_TURN_ON – Turns ON the LCD Display.
- _LCD_TURN_OFF – Turns OFF the LCD Display.
- _LCD_FIRST_ROW – Moves the cursor to the first row.
- _LCD_SECOND_ROW – Moves the cursor to the the second row.
- _LCD_THIRD_ROW – Moves the cursor to the third row.
- _LCD_FOURTH_ROW – Moves the cursor to the fourth row.
- _LCD_CLEAR – Clears the LCD Display.
- _LCD_CURSOR_OFF – Turns ON the cursor.
- _LCD_UNDERLINE_ON – Turns ON the cursor underline.
- _LCD_BLINK_CURSOR_ON – Turns ON the cursor blink.
- _LCD_MOVE_CURSOR_LEFT – Moves cursor LEFT without changing the data.
- _LCD_MOVE_CURSOR_RIGHT – Moves cursor RIGHT without changing the data.
- _LCD_SHIFT_LEFT – Shifts the display left without changing the data in the display RAM.
- _LCD_SHIFT_RIGHT – Shifts the display right without changing the data in the



RV1
10K
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
C1
22pf
C2
22pf
X1
8MHz
U1
OSC1/CLKIN
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
MCLR/Vpp/THV
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
R1
10k
PIC16F877A

ANALOG TO DIGITAL CONVERTER

ELECTRONIC VOLTMETER WITH
PIC16F877A

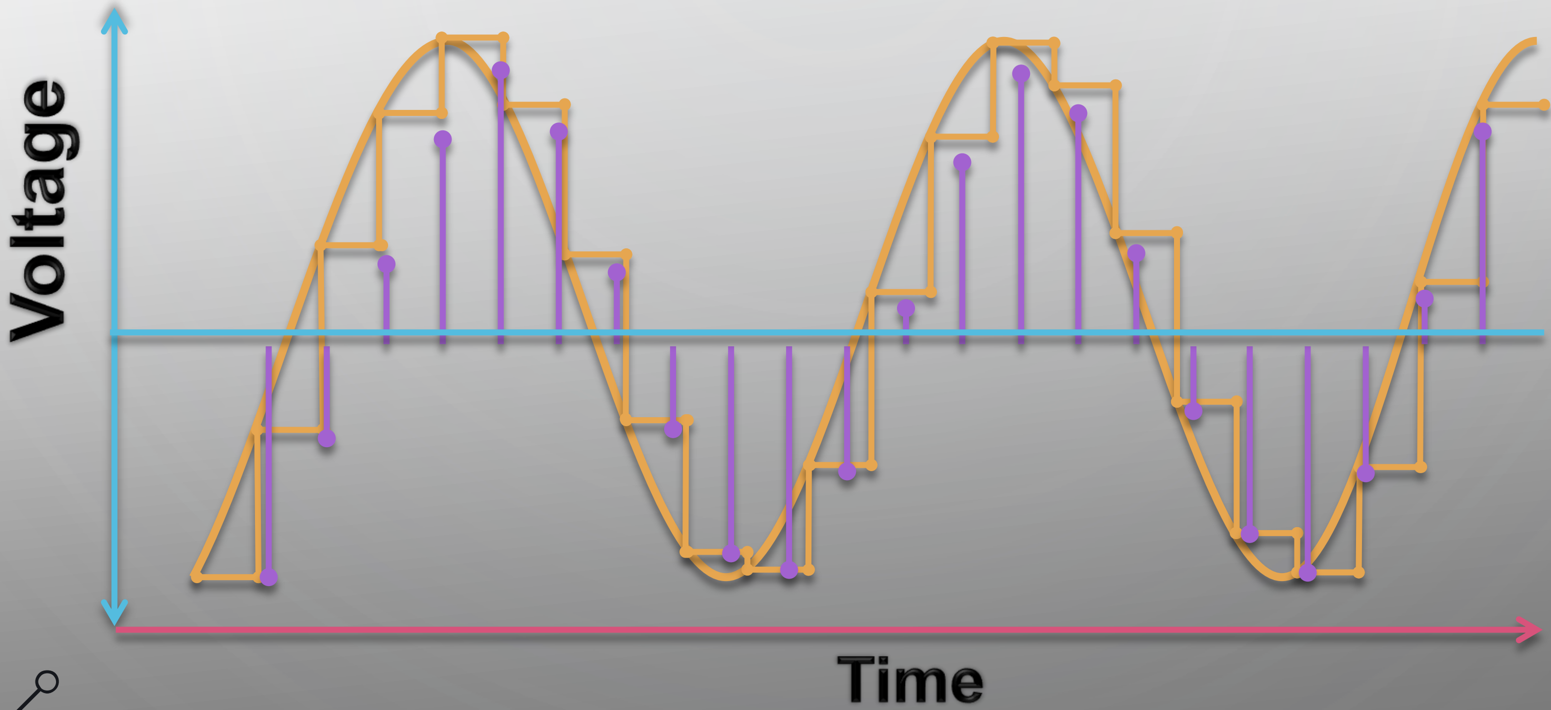
Analog to Digital Converter(ADC)

- Just about every modern microcontroller contains an ADC(s).

- It converts analog voltages into digital values.

- These digital representations of the signal at hand can be analyzed in code, logged in memory, or used in practically any other way possible.

The Analog to Digital Converter (ADC)



The Analog to Digital Converter (ADC)

PTC Specifications:

100Ω @ 25℃
+ 1Ω/ 1℃

(ex. @ 26℃, R = 101Ω
24℃, R = 99Ω

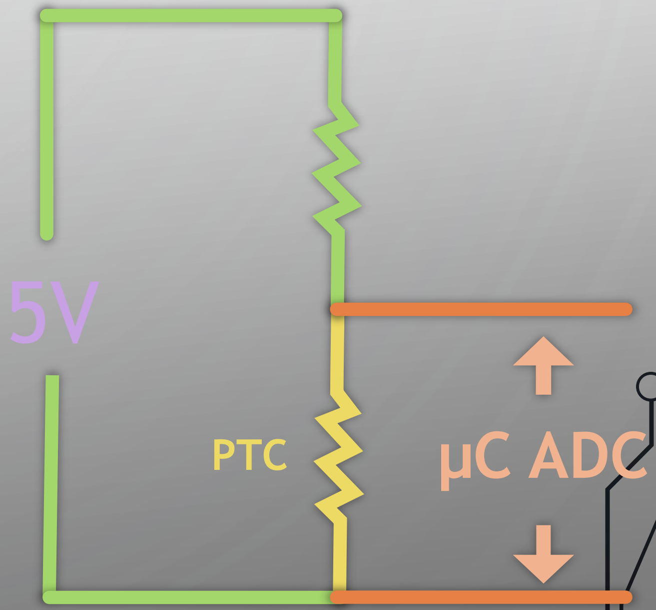
A 10-bit ADC will represent a voltage between 0 to 5 as a number between 0 to 1024.

code

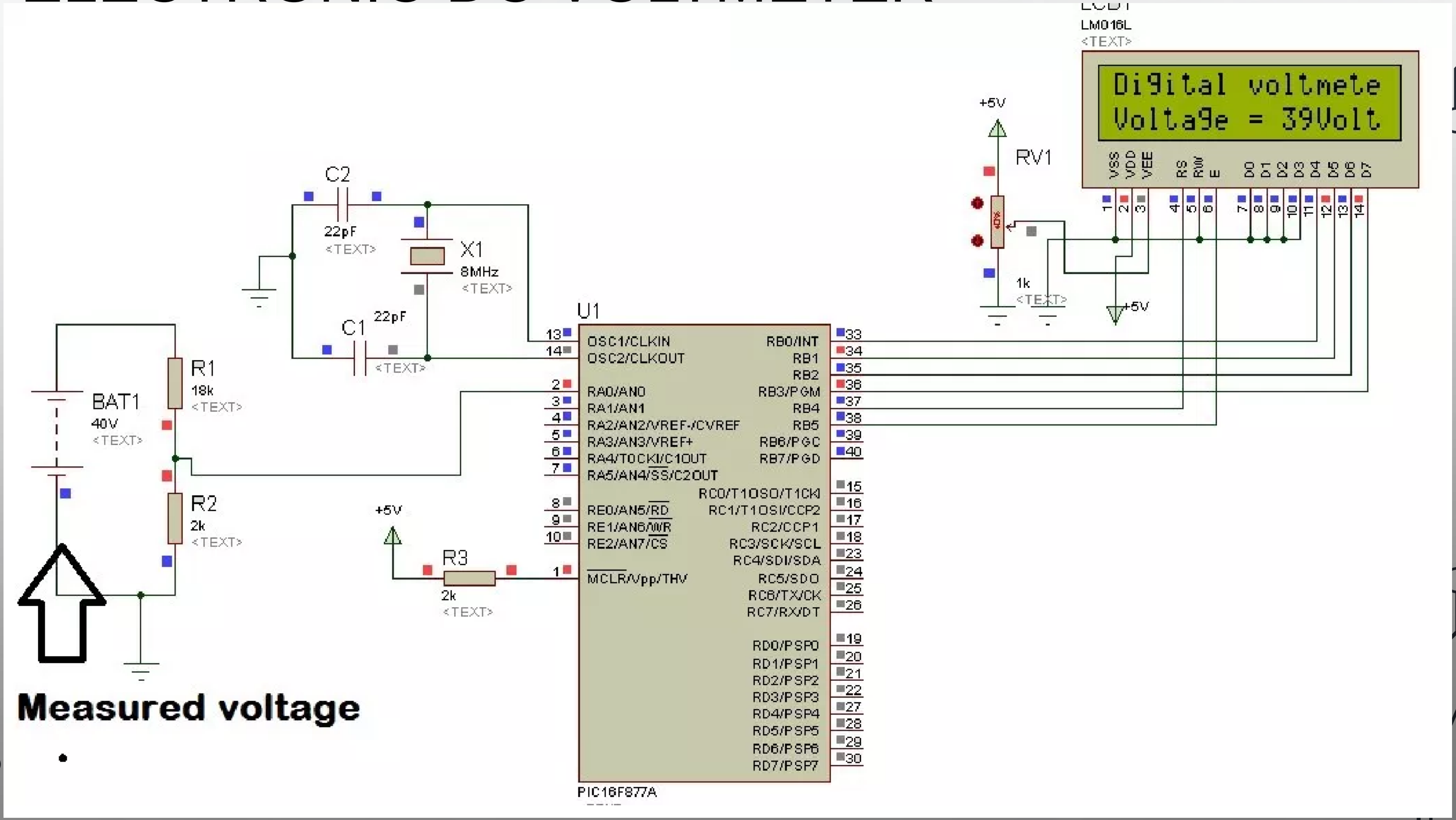
```
Void Loop()

voltage25C = 512
voltageADC = ADC.input(pin1)

ratio = voltageADC / voltage25C
temperature = ratio * 25
```



ELECTRONIC DC VOLTMETER



CALCULATION

Vout = (R2 / R1+R2) * Vinput

Vout = (2 / 18+2) * 40 = 4 volt

- ADC module of pic microcontroller converts analog signal into binary numbers. PIC16F877A microcontroller have 10 bit ADC.
- PIC16F877A microcontroller have 10-bit ADC and it counts binary from 0-1023 for every minimum analog value of input signal.

Explanation of Code

```
while (1)

{

voltage = ADC_Read(0);

voltage = (voltage * 5 * 10)/ (1024);

inttostr(voltage,volt); // it converts integer value into string

Lcd_Out(2,1,"Voltage = ");

Lcd_Out(2,11,Ltrim(volt));

Lcd_Out(2,13,"Volt");

}
```

Thanks!

ANY QUESTIONS