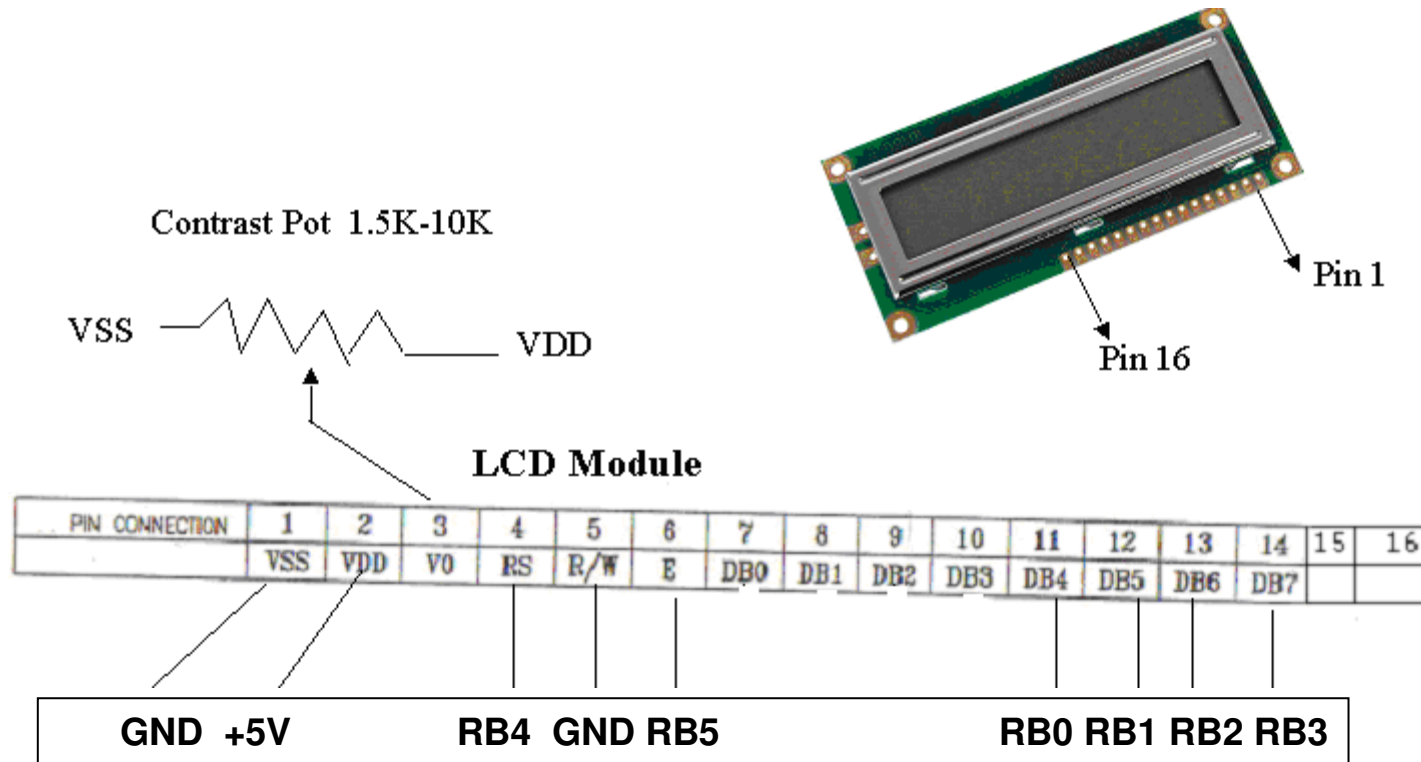


LCD

# LCD

- Contains its own controller HITACHI standard that maintains LCD segments and provides display features ( blinking, character position ,refresh, special characters)
- Has a multiplexed command and data parallel interface
- Low cost ( <\$6 for small displays)
- Variety of sizes and shapes
- Fairly ubiquitous in modern products
- PIC 16917 has internal peripheral that can drive LCD segments directly without need for HITACHI

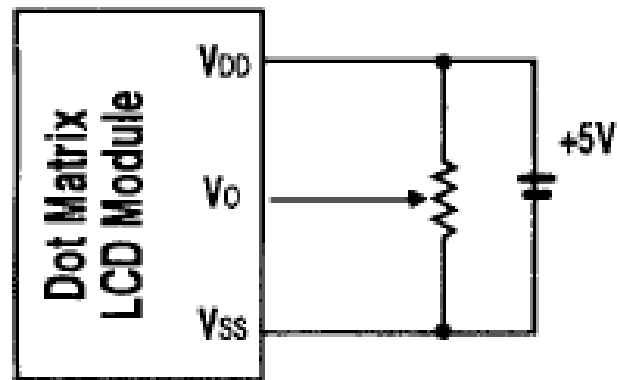
# Connections-4 bit data and command bus



# LCD Interface multiplexed Data and Control

PIN NO.	SIGNAL	LEVEL	DESCRIPTION	FUNCTIONS
1	VSS	-	Ground	0V
2	VDD	-	Supply voltage for logic & LCD (+)	5v±5%
3	V0	-	Supply voltage for LCD	Decision by user system
4	RS	H/L	Register selection	H : Date L: Instruction code
5	R/W	H/L	Read/Write	H : Read L : Write
6	E	H,H→ L	Enable signal	<p>8 bits</p> <p>4 bits</p>
7	DB0	H/L	Data bit 0	
8	DB1	H/L	Data bit 1	
9	DB2	H/L	Data bit 2	
10	DB3	H/L	Data bit 3	
11	DB4	H/L	Data bit 4	
12	DB5	H/L	Data bit 5	
13	DB6	H/L	Data bit 6	
14	DB7	H/L	Data bit 7	

# Contrast Control



For Modules With Normal Temperature Range Fluid

Important bias setting to discern segments –too much all dark—too little invisible

# Instruction ( Commands)

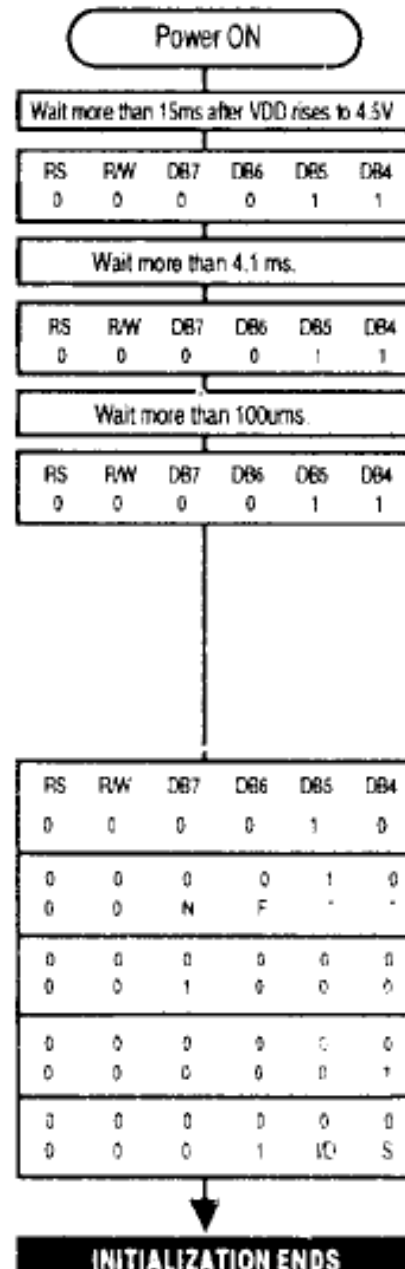
INSTRUCTION	CODE										DESCRIPTION	EXECUTE TIME (MAX)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).	1.64mS
Cursor at Home	0	0	0	0	0	0	0	0	1	-	Returns the cursor to the home position (Address 0). Also returns the display being shifted to the original position. DDRAM contents remain unchanged.	1.64mS
Entry Mode Set	0	0	0	0	0	0	0	1	I / D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40 $\mu$ S
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D) cursor ON/OFF (C), and blink of cursor position character(B).	40 $\mu$ S
Cursor / Display Shift	0	0	0	0	0	1	S/C	R / L	-	-	Moves the cursor and shifts the display without changing DDRAM contents.	40 $\mu$ S
Function Set	0	0	0	0	1	DL	N	F	-	-	Sets interface data length (DL) number of display lines (L) and character font (F).	40 $\mu$ S
CGRAM Address Set	0	0	0	1	ACG					Sets the CGRAM address. CGRAM data is sent and received after this setting.		40 $\mu$ S
DDRAM Address Set	0	0	1	ADD					Sets the DDRAM address. DDRAM data is sent and received after this setting.		40 $\mu$ S	
Busy Flag / Address Read	0	1	BF	AC					Reads Busy flag (FB) indicating internal operation is being performed and reads address contr contents.		0 $\mu$ S	
CGRAM / DDRAM Data Write	1	0	Write Data					Writes data into DDRAM or CGRAM.		40 $\mu$ S		
CGRAM / DDRAM Data Read	1	0	Read Data					Reads data from DDRAM or CGRAM.		40 $\mu$ S		

# Command Delineator descriptions

CODE	DESCRIPTION	EXECUTE TIME(MAX)
VD = 1 : Increment VD = 0 : Decrement S = 1 : With display shift SiC = 1 : Display Shift SiC = 0 : Cursor movement RiL = 1 : Shift to the right RiL = 0 : Shift to the left DiL = 1 : 8 bit DL = 0 : 4 bit N = 1 : 1/16 Duty N = 0 : 1/8 Duty, 1/11 Duty F = 1 : 5 X 10 dots F = 0 : 5 X 7 dots BF = 1 : Internal Operation is being performed BF = 0 : Instruction acceptable	DDRAM : Display Data RAM CGRAM : Character Generator RAM ACG : CGRAM ADDRESS ADD : DDRAM ADDRESS Corresponds to Cursor Address AC : Address Counter, used for both DDRAM and CGRAM ⚡ Invalid	fcp or fosc = 250KHz However, when frequency changes, execution time also changes When fcp or fosc = 270KHz, $40\mu\text{S} \times \frac{250}{270} = 37\mu\text{S}$

# Initialization Scheme

- 1.) Display Clear
- 2.) Function Set
  - DL=1 : 8 bit interface data
  - DL=0 : 4 bit
  - F=0 : 5x7 dot character font
  - N=1 : 1/16 Duty
  - N=0 : 1/8 Duty, 1/11 Duty
- 3.) Display ON/OFF Control
  - D=0 : Display OFF
  - C=0 : Cursor OFF
  - B=0 : Blink OFF
- 4.) Entry Mode Set
  - 1/D=1 : + 1 (increment)
  - S=0 : No shift



**When interface is 4 bits long.**

BF cannot be checked before this instruction  
Function set (interface is 8 bits long)

BF cannot be checked before this instruction  
Function set (interface is 8 bits long)

BF cannot be checked before this instruction  
Function set (interface is 8 bits long)

BF can be checked after the following instruction.  
When BF is not checked, the waiting time between instructions is longer than execution instruction time. Function set (Set Interface to be 4 bits long). Interface is 8 bits length.

Function set (interface is 8 bits long. Specify the number of display lines and character font.)  
The number of display lines and character font cannot be changed afterwards.

Display OFF

Display ON

Entry Mode Set

# Display Positions

## ■ 16X1

	DISPLAY POSITION															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	DD RAM ADDRESS															

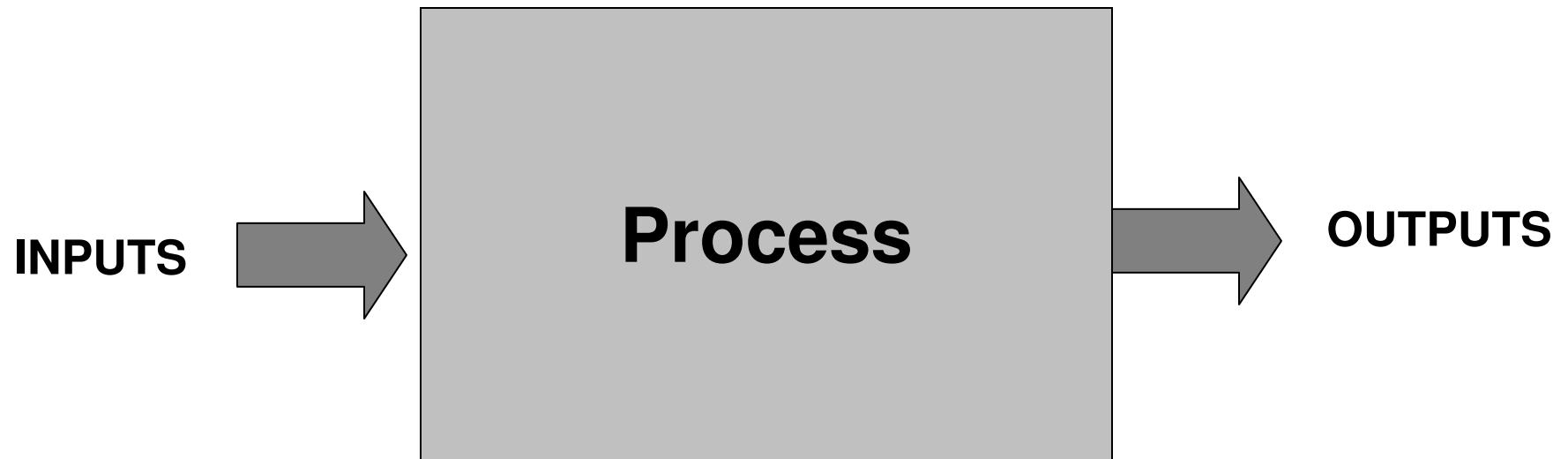
## ■ 16X2

	DISPLAY POSITION															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Line 2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
	DD RAM ADDRESS															

## ■ 20X2

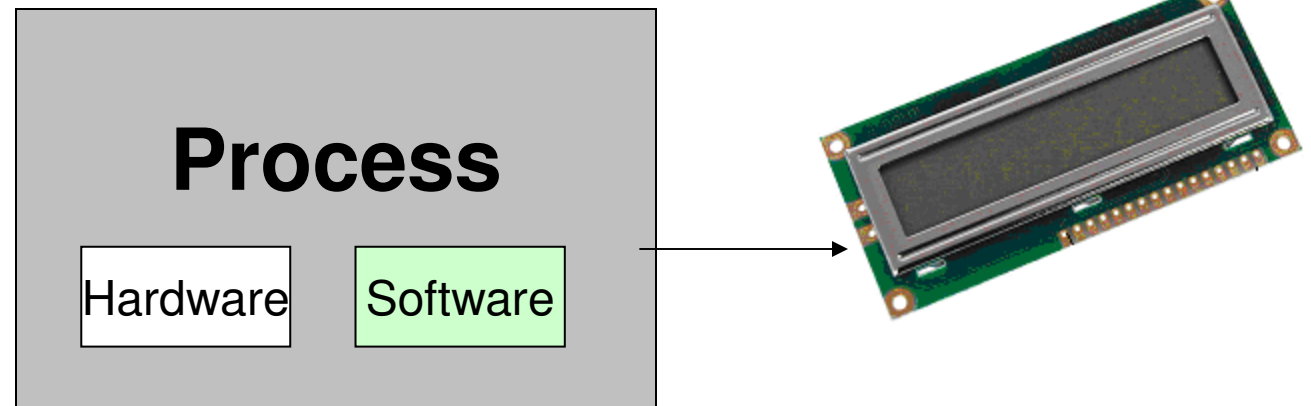
	DISPLAY POSITION																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Line 1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
Line 2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
	DD RAM ADDRESS																			

# Input Process Output (IPO) Diagram



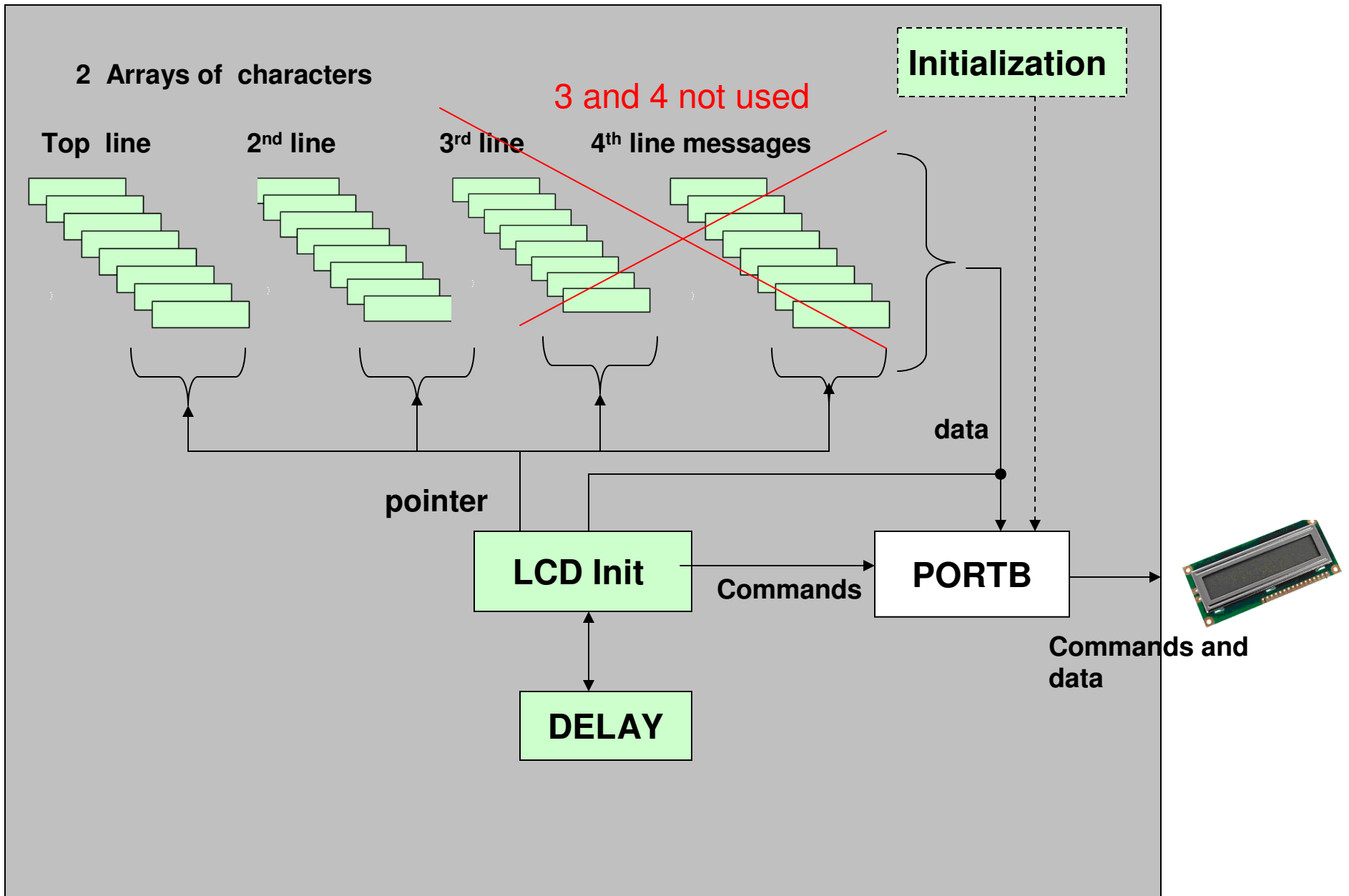
# LCD IPO

**Microcontroller**



Displays simple message in its 16 characters x1 rows character space:  
" I Love Micros"

# Process Details

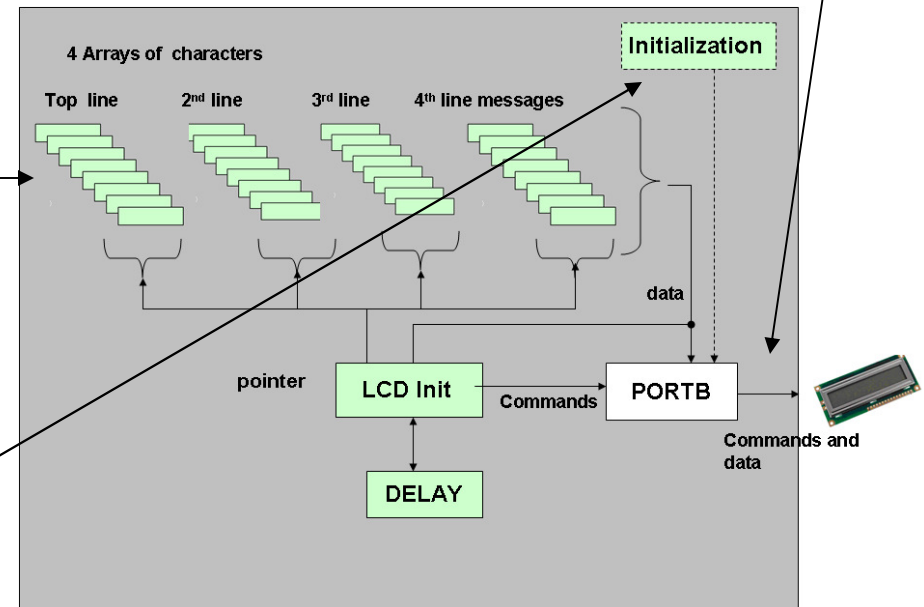


# Initialization

```
#define LCDPORT PORTB
#define E RB5
#define RS RB4
// R/W tied low for always data write not read
//RB3:RB0 - LCD I/O D7:D4 (pins 14:11)
// RB4- LCD E CLOCKING Pin
// RB5 - LCD R/S Pin
```

```
const char TopMessage[] = "EET 250 Intro to ";
const char SecMessage[] = "Microcontrollers ";
const char ThirdMessage[] = "Microchip PIC16F917 ";
const char FourMessage[] = "MPLAB and PICKIT2 ";
```

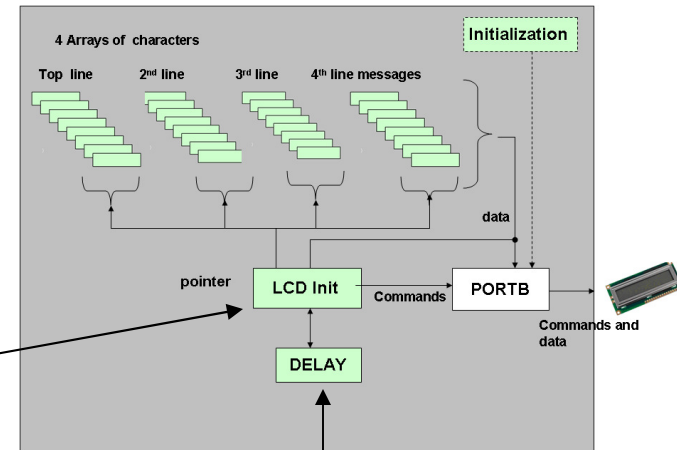
```
void initLCDIo(void)
{
    //configure port B to drive for output
    TRISB0=0;
    TRISB1=0;
    TRISB2=0;
    TRISB3=0;
    TRISB4=0;
    TRISB5=0;
}
```



# Processing

```

void LCDInit() {
    int i = 0;
    LCDPORT = 0; //start with everything low
    //power up wait
    Delay(20); //20 milliseconds
    LCDPORT = 3; //start initialization process
    E=1;
    E=0;          //send reset command
    Delay(5);
    E=1;
    E=0;          //repeat reset command
    Delay_Us(200);
    E=1;
    E=0;          //send reset command third time
    Delay_Us(200);
    LCDPORT = 2 ; //initialize LCD 4 bit mode
    E=1;
    E=0;
    Delay_Us(200);
    //LCDWrite(0b00101000,0); //LCD is 4 bit I/F, 2 line
    LCDWrite(0x08,0);
    LCDWrite(0b00000001,0); //Clear LCD
    LCDWrite(0b00000110,0); // Move Cursor after each character
    LCDWrite(0b00001110,0); //turn on LCD and Enable Cursor
    for ( i=0; TopMessage[i] !=0; i++)
        LCDWrite(TopMessage[i],1);
    for ( i=0; SecMessage[i] !=0; i++)
        LCDWrite(SecMessage[i],1);
    for ( i=0; ThirdMessage[i] !=0; i++)
        LCDWrite(ThirdMessage[i],1);
    for ( i=0; FourMessage[i] !=0; i++)
        LCDWrite(FourMessage[i],1);
}
    
```



```

void Delay(int x) { //approx 23 msec
    int i, j;
    int iEnd = 30;          // Outside Loop Value
    int jEnd = 40;          // Inside Loop Value

    for (i = 0; i < iEnd; i++) // Delay Loop
        for (j = 0; j < jEnd; j++);
}

void Delay_Us(int x) { //212 usec
    int i, j;
    int iEnd = 2;          // Outside Loop Value
    int jEnd = 3;          // Inside Loop Value

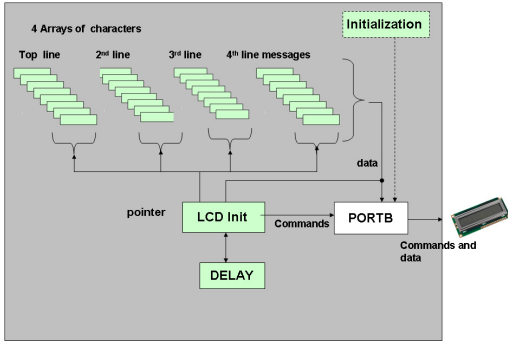
    for (i = 0; i < iEnd; i++) // Delay Loop
        for (j = 0; j < jEnd; j++);
}
    
```

# Functions Used within Processing

```
void update_Display(char*chpointer, int numofchar) {  
    int i=0;  
    // LCDWrite(0b00000001,0); //Clear LCD  
    for (i =0; i<numofchar; i++) {  
        LCDWrite(*chpointer,1);  
        chpointer++;  
    }  
}
```

```
LCDWrite( int LCDDData, int RSValue)  
{  
    LCDPORT = (LCDDData >> 4)&0x0f; //get high 4 bits for output  
    RS =RSValue;  
    //toggle E  
    E =1;  
    E=0;  
    LCDPORT = LCDDData &0x0f; //get low 4 bits for output  
    RS =RSValue;  
    //toggle E  
    E =1;  
    E=0;  
    if ((0==(LCDDData & 0xFC)) && (0==RSValue))  
        Delay(5) ; //delay 5 msec  
    else  
        Delay_Us(200);  
}
```

```
void clear_display() {  
    LCDWrite(0b00000001,0); //clear LCD  
}
```



```
void move_FirstLine() {  
    unsigned char address =0x80+0x00; // line 1  
    LCDWrite(address,0); //move cursor to display  
}  
void move_SecondLine() {  
    unsigned char address =0x80+0x40; // line 2  
    LCDWrite(address,0); //move cursor to display  
}  
void move_ThirdLine() {  
    unsigned char address =0x80+0x14; // line 3  
    LCDWrite(address,0); //move cursor to display  
}
```

# Exercise

1. Disconnect PICKIT2 from DIP prototype
2. Construct interface as shown in connections slide ( pot wire and LCD will be provided)
3. Be sure to conserve space for both LCD display and PICKIT2 plug in on DIP prototype
4. Show construction to instructor
5. Navigate to:C:\EET250\16F887\Lesson 16 LCD Module\16x1LCD
6. Hook PICKIT2 to DIP prototype
7. Open LCD.mcp ,select debugger as PICKIT2
8. Notice if PICKIT2 shows ready in Output window
9. Build, download program and execute
10. Verify that LCD shows “I Love Micros” Message
11. Your turn— change message to two 8 character text string by modifying Arrays.—build, download and verify