بسم الله الرحمن الرحيم

In the name of Allah

Graphical LCD for beginners and interfacing with PIC MCU

By

Eng. Mustafa H. Abyad

Cairo, Egypt

March 2009

Table of contents

1. Introduction

Really graphical LCDs are widely used in many applications for different fields, such as mobile phones, calculators, digital oscilloscopes, and in the medical field as ECG, monitors, defibrillators, ventilators …
They are used for displaying data, pictures, shapes, graphs (real time or fixed), trends, tables...
They can be divided into two main categories; two color GLCD and color GLCD, we will deal with the 2 color LCD through this course.
I have two targets behind writing these papers; 1st to write a basic primer on the operation of any available graphical LCD in our market and 2nd describe how to drive it using PIC microcontrollers (with assembly and Pic basic languages) to be easily used by students in their projects.
The graphical LCD type I choose was the TOPWAY LM12864LFC (128 x 64 pixel) graphical display driven by Samsung KS0108 driver.I choose this type especially for 2 reasons; 1st I searched the Egyptian market and I found this type in ***"RAM electronics"*** with low coast, 2nd the KS0108 driver can be easily used.

2. The LM12864LFC

The LM12864LFC is a 128 x 64 pixel graphical LCD with backlight. It is driven by 2 64 x 64 pixel Samsung KS0108 drivers as shown in figure 1.

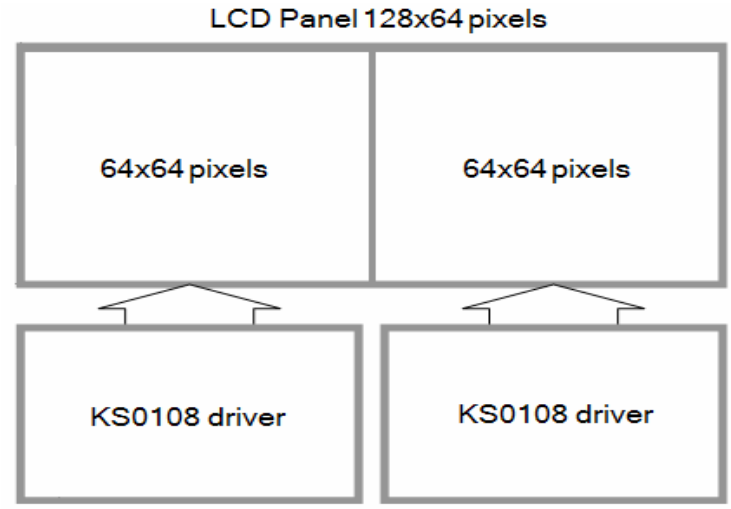

Figure 1: LCD drivers

Figure 2: LM12864LFC LCD

2.1 Pin Assignments

The LM12864LFC has 20 pins can be classified into 3 groups: 1- Power related pins, 2- Data bus, and 3- Control pins. Table 1 contains a detail pin-out of the LCD.

Pin No.	Pin Name	I/O	Descriptions
1	VSS	Power	Negative Power Supply, Ground (0V)
2	VDD	Power	Positive Power Supply
3	V0	Power	LCD Contrast reference
4	RS	Input	RS = H; DB0 – DB7 = Display RAM data RS = L; DB0 – DB7 = Instruction data
5	R/W	Input	In read mode R/W = H; Data read form the LCD module, data appears at DB0 – DB7 and can be read by the host while, E = H and the device is being selected In write mode R/W = L; Data write to the LCD module, data appears at DB0 – DB7 will be written into the LCD module at E = H→L and device is being selected
6	E	Input	
7	DB0	I/O	Data bus; Three state I/O terminal for display data or instruction data
:	:	:	
14	DB7	I/O	
15	CS1	Input	Chip selection, When CS1=1 (*1) enable access to the Left Side (64 column) of the LCD module
16	CS2	Input	Chip selection When CS2=1 (*1) enable access to the Right Side (64 column) of the LCD module
17	/RST	Input	Reset signal /RST = L, Display off display start line register becomes 0 no command or instruction data could be accepted /RST = H, Normal running
18	VOUT	Output	Power Booster output for V0
19	A	Power	Positive Power for LED backlight
20	K	Power	Negative Power for LED backlight

Table 1: a detail pin-out of the LCD

2.1.1 Power & Setting up pins

The important pins for this section are as follows:

Pin #	Symbol	Function/Typical Value
2	VDD	Supply voltage – connect to 5V
1	VSS(GND)	Ground – connect to ground
3	V0	Operating voltage – Connect through pot as is described below
18	VOUT	Negative Voltage output – outputs -5V – connect to pot
19	A	Power supply for backlight(+)
20	K	Power supply for backlight(-)

Table 2: Power & Setting up pins

In order for the LCD to power up, Pin 2 must be connected to + 5V, Pin 1 must be connected to GND. Pin 3 & Pin 18 must be connected as illustrated in Fig. 2. Pin 18 generates -5V as an output and it must be run through a trim pot (or voltage divider) and fed into Pin 3. This provides the voltage differential of Vdd-V0 which must be at least 7.5V. Adjusting this value adjusts the contrast but the Pins must be connected in this way in order for the image to be seen on the screen (if it isn't there is essentially no contrast and nothing will be displayed.

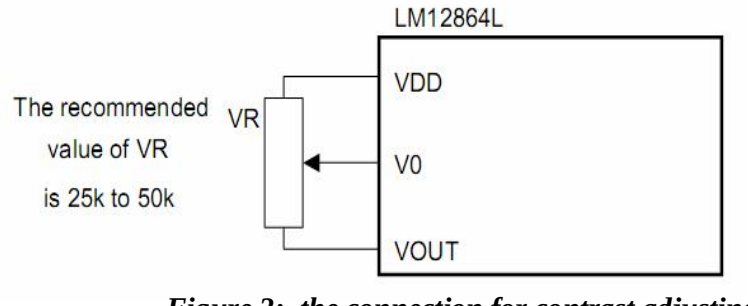

Figure 3: the connection for contrast adjusting

Pin 19 & 20 connects the power needed for backlight Pin 19 +5V and Pin 20 GND, these pins may be inverted in other models.

NOTE: *RST Pin must be high in order to be able to write on the LCD because while the /RST is low no instruction can be accepted i.e. Display off.*

2.1.2 Data bus

There are 8 data bits that provide a number of functions in the operation of the LCD. They are the main information carriers to the LCD or called the Data bus. They are located on Pins 7 - 14 with Pin 7 assigned to Data Bit 0 and Pin 14 assigned to Data Bit 7. Table 3 gives a brief overview of these assignments.

Pin #	Symbol	Function/Typical Value
4	DB0	Data bit 0
5	DB1	Data bit 1
6	DB2	Data bit 2
7	DB3	Data bit 3
8	DB4	Data bit 4
9	DB5	Data bit 5
10	DB6	Data bit 6
11	DB7	Data bit 7

Table 3: Data bus

The functions of the data bits will become clearer in later sections of the paper.

2.1.3 Control pins

There are 6 control pins which are used to control the operation of the KS0108 hardware drivers and display data on the LCD. They are listed in table 4.

# Pin	Symbol	Function/Typical Value
15	CS1	Chip Select 1 – Selects the left KS0108 driver which is also left half of the screen
16	CS2	Chip Select 2 – Selects the right KS0108 driver which is also right half of the screen
17	/RST	Reset – set low to reset the display, high otherwise
5	R/W	Read/Write – set high to read from the LCD to the MCU, set low to write from the MCU to the LCD
4	RS (or D/I)	Data/Instruction – tells the LCD whether or not data is being written to the screen or the MCU is using the data bits to perform an instruction – set high for data transfer and set low to designate and instruction is being performed
6	E	Enable – The enable is used to clock operations to the LCD

Table 4: control pins

R/W and D/I are used to determine the mode of operation that the LCD is in. Table 5 illustrates how these two control bits are used to control operations.
Enable must be set high and then low in order for an operation to be passed to the LCD.

D/I	R./W	Mode
0	0	An Instruction is being written (such as clear the LCD)
0	1	MCU reads the status of the LCD – whether it is busy or ready for another command
1	0	Data write – data is written to the display ram at whatever X-Y coordinates have been set
1	1	Data read – the data in the display RAM is read to the MCU

Table 5: R/W and D/I configurations

3. Writing Data to the Screen

3.1 Turning the Display (RAM) On

First things first, the display needs to be turned on this is done by sending the following instruction to the screen:

Instruction	D/I	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Display ON	0	0	0	0	1	1	1	1	1	1

Table 6: Display ON instruction

Lets see how to turn on the display using MCU PIC16f877/A by assembly and by Picbasic languages, first the LCD is connected as shown in figure 7.

Assembly

```
BCF PORTC,2          ; RC2 = 0  (RS or D/I = 0)          Instruction
BCF PORTC,5          ; RC5 = 0 (R/W = 0)                 is being written
MOVLW 0X3F           ; DB7  DB6  DB5  DB4  DB3  DB2  DB1  DB0
MOVWF PORTD          ;  0     0     1     1     1     1     1     1
BSF PORTE,0          ; RE0 = 1 (E = 1)                   Clock to pass the
BCF PORTE,0          ; RE0 = 0 (E = 0)                   order to the LCD
```

PicBasic

```
PORTC.2 = 0          ; RS or D/I = 0                     Instruction
PORTC.5 = 0          ; R/W = 0                           is being written
PORTD = $3F          ;  DB7  DB6  DB5  DB4  DB3  DB2  DB1  DB0
                     ;  0     0     1     1     1     1     1     1
PORTE.0 = 1          ; RE0 = 1 (E = 1)                   Clock to pass the
PORTE.0 = 0          ; RE0 = 0 (E = 0)                   order to the LCD
```

The reason that I drew this example out and included the code (instead of just referring to later sections) is so that it is clear how to go from the instruction outlined in Table 6 to performing it assembly or Pic basic.

3.2 Placing data on the screen

In order to place any information on the screen, it is important to understand how the bits control what is on the screen. There are 8192 pixels on a 128 X 64 pixel screen divided into 2 halves left and right, each half is 64 X64 pixel which is divided into 8 pages (X axis), each page is 8 X 64 pixels divided into 64 column (Y axis). Each column is 8 pixels vertically from D0 to D7 and so data is entered 8 pixels by 8 pixels. Figure 4 and Figure 5 show how the screen is broken down into its X and Y axes. Fig 4 shows the left half of the screen (if CS1 is 1 and CS2 is 0). The Y address refers to which line the pixels should be written to and the X page sets the column to which they will be written to.

Page (X) address	data	LCD Display (front view)	
0	D0 : D7		
1	D0 : D7		
2	D0 : D7		
3	D0 : D7	128x64 pixels	
4	D0 : D7		
5	D0 : D7		
6	D0 : D7		
7	D0 : D7		
Column(Y) Address		00h → 3Fh (0 → 64)	00h → 3Fh (0 → 64)
Chip Select		CS1=1, CS2=0	CS1=0, CS2=1

Figure 4: show how the screen is broken down into X and Y axes for each of the 2 halves.

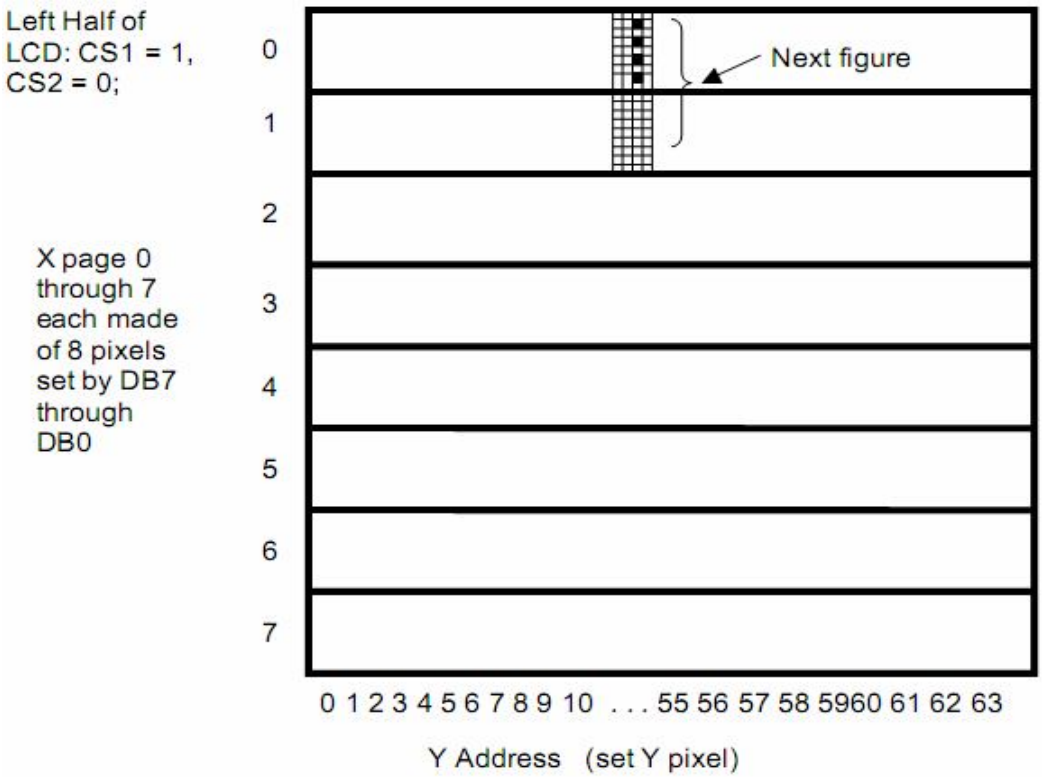

Figure 5: illustrates how the left half (or right half if CS2 = 1 and CS1 = 0) works. The Y – address determines what Y line the pixels will be placed on, the X page determines which of the eight vertical 8 pixel strips the pixels will be placed on and the Data bits of the write data instruction will the be places across the X page on the specified Y line.

There are three basic steps that must be done in order to determine where the pixels will go:
1- Setting the Y address.
2- Setting the X address.
3- Sending data.
Figure 6 gives an example of writing data of 8 pixels on the LCD.

The Y address actually has a counter so it need only be set once and then every time there is a data write it will be incremented to the next line.

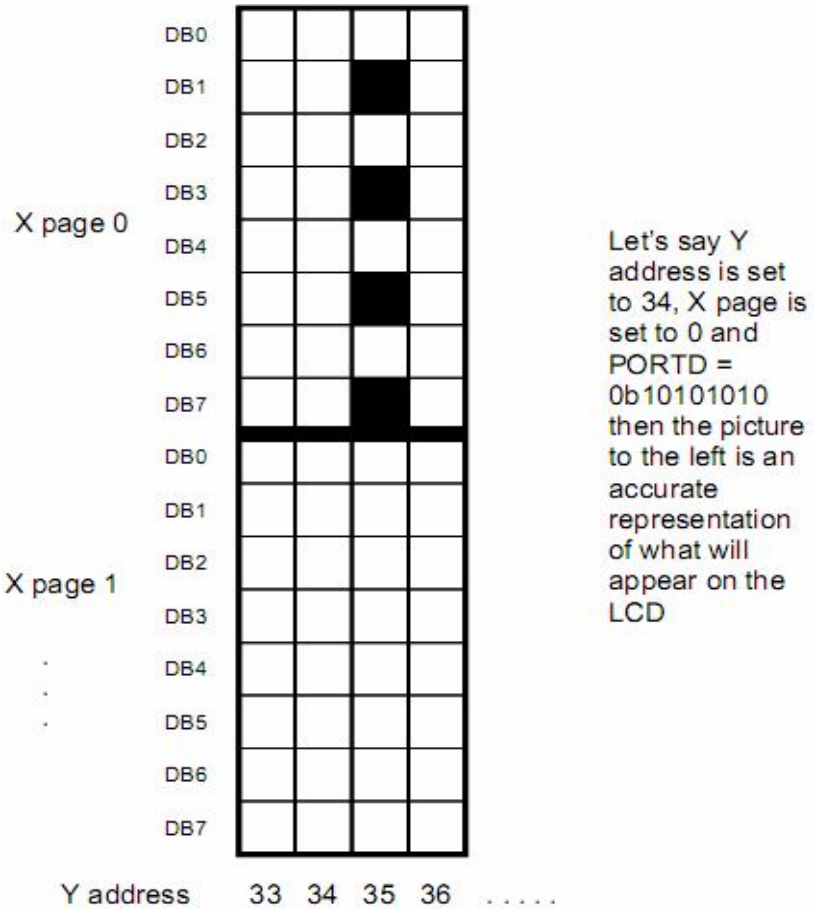

Figure 6: a zoomed in version of Figure 5, it also gives an example of how a pixel is written. If the Y address is set to 34, X page is set to 0, and a data write instruction is given with PORTD = 0b10101010, then the pixels in those locations will be dark.

This is the basic idea of placing data on the LCD but how can you implement these 3 steps and how the LCD knows the Y and X axes you want to write in?
There are special instructions for setting column (Y) address, setting page (X) address and writing data on the display table 7 shows these instructions.

Instructions	**RS**	**R/W**	**DB7**	**DB6**	**DB5**	**DB4**	**DB3**	**DB2**	**DB1**	**DB0**	**Function**
Set column (Y) address	0	0	0	1	Y address (0 - 63)						Set column address into the Y address counter
Set page (X) address	0	0	1	0	1	1	1	X address (0–7)			Set page address into the X address register
Write data	1	0	Data 8 pixels								Write display data into display data RAM. Then Y address counter increases by 1 automatically.

Table 7: display control instructions

Now I think that our 1st target was verified and the operation of the LCD became clear, the following section will describes interfacing of the LCD with MCU PIC and how to drive it using PIC.

4. Interfacing with PIC MCU

4.1 Hardware

Figure 7 shows the connection between the LCD and the PIC that we will build on the software programs. This is not a standard connection, you can make any other one suitable for the requirements of your project.

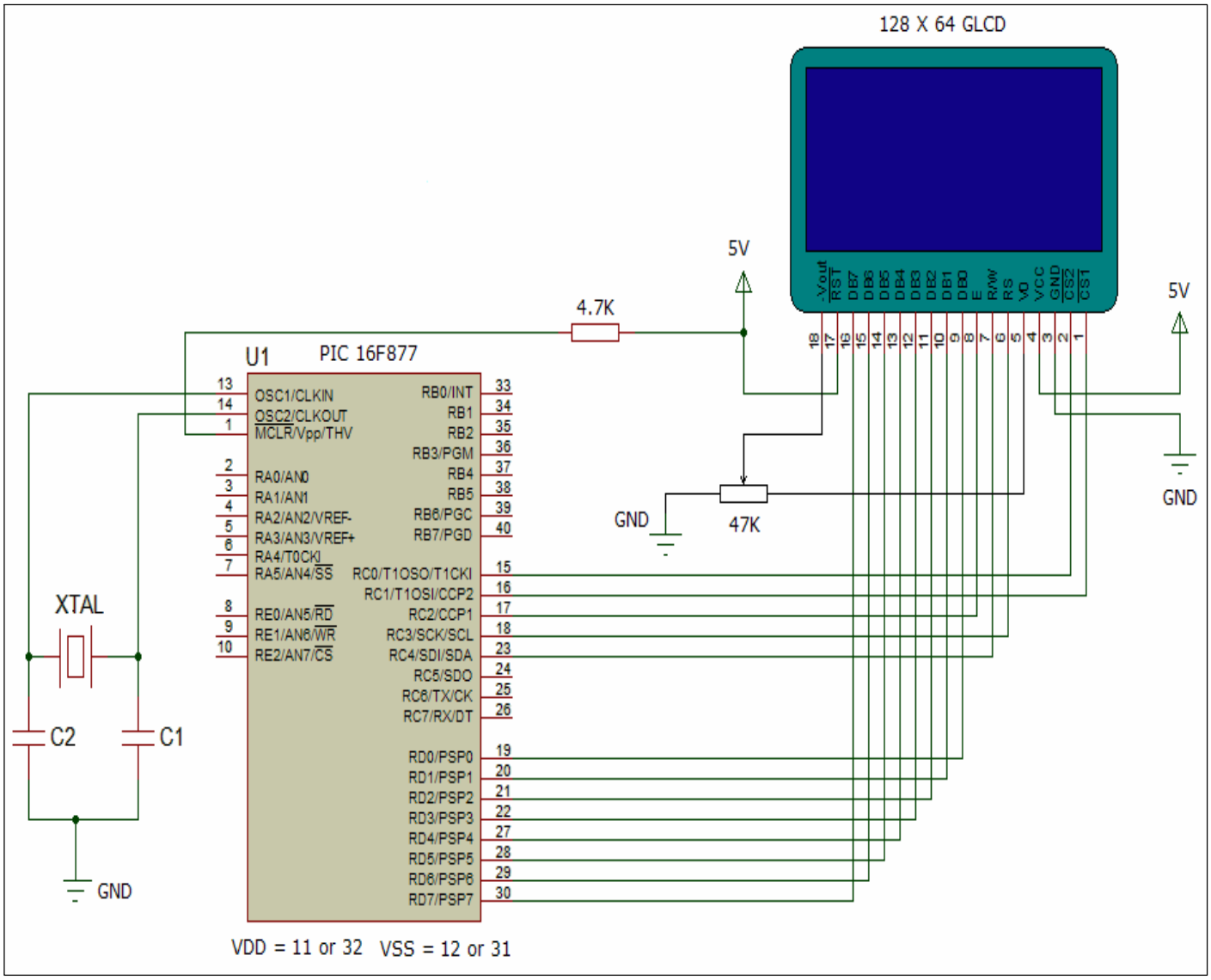

Figure 7 interfacing schematic

Table 8: interfacing connections

LCD Pins	PIC 16F877 Pins
Data bus	PORTD
DB0	RD0
:	:
DB7	RD7
E	RC2
R/W	RC4
RS	RC3
CS1	RC1
CS2	RC0

4.2 Software

In this section we will show how to write a software program by assembly and Picbasic languages for interfacing the GLCD with the PIC.

4.2.1 Assembly

We want to draw an ellipse as shown in figure 8.

Figure 8: ellipse displayed on GLCD

First the image must be converted into pixels (128X64 pixels maximum)
I converted it using a program called "Bmp2asm" that covert the image into hex numbers in a text file.
I downloaded the program from the internet.
Second write the program as follows using MPLAB program
Put the data (display data or instruction data) on port D then call each of three subroutines; the first subroutine for sending command called "S_CMD", CS1 & CS2 are both activated in order not to make two different subroutines for each side of the GLCD then clock the E pin. The second subroutine for sending data into 1st half of the LCD and called "S_DATA1" here we activate CS1 and deactivate CS2 then clock the E pin. The third subroutine for sending data into 2st half of the LCD and called "S_DATA2" and here we activate CS2 and deactivate CS1 then clock the E pin.

Assembly Code

```
#include<P16F877.inc>
ORG 0X00
GOTO MAIN
ORG 0X05
MAIN

BSF STATUS,RP0
BCF STATUS,RP1              ; BANK 1
CLRF TRISD
CLRF TRISC
BCF STATUS,RP0
BCF STATUS,RP1              ; BANK0

MOVLW 0X3F                  ;
MOVWF PORTD                 ; display ON      (see 3.1)
CALL S_CMD                  ;



MOVLW 0X55          ; y address (line) = 21  (0101 0101)
MOVWF PORTD
CALL S_CMD
MOVLW 0XB9          ; x address (page) = 1  (1011 1001)
MOVWF PORTD
CALL S_CMD

MOVLW 0XC0
MOVWF PORTD
CALL S_DATA1
MOVLW 0X20
MOVWF PORTD
CALL S_DATA1
MOVLW 0X20
MOVWF PORTD
CALL S_DATA1
```

```
MOVLW 0X20
MOVWF PORTD
CALL S_DATA1
MOVLW 0XC0
MOVWF PORTD
CALL S_DATA1

MOVLW 0X53          ; y address (line) = 19  (0101 0011)
MOVWF PORTD
CALL S_CMD
MOVLW 0XBA          ; x address (page) = 2  (1011 1010)
MOVWF PORTD
CALL S_CMD

MOVLW 0XF0
MOVWF PORTD
CALL S_DATA1
MOVLW 0X0F
MOVWF PORTD
CALL S_DATA1

MOVLW 0X5A          ; y address (line) = 26  (0101 1010)
MOVWF PORTD
CALL S_CMD

MOVLW 0X0F
MOVWF PORTD
CALL S_DATA1
MOVLW 0XF0
MOVWF PORTD
CALL S_DATA1

MOVLW 0X53          ; y address (line) = 19  (0101 0011)
MOVWF PORTD
CALL S_CMD
MOVLW 0XBB          ; x address (page) = 3  (1011 1011)
MOVWF PORTD
CALL S_CMD
MOVLW 0X01
MOVWF PORTD
CALL S_DATA1
MOVLW 0X1E
MOVWF PORTD
CALL S_DATA1
MOVLW 0X60
MOVWF PORTD
CALL S_DATA1
MOVLW 0X80
MOVWF PORTD
CALL S_DATA1
MOVLW 0X80
MOVWF PORTD
CALL S_DATA1
MOVLW 0X80
MOVWF PORTD
CALL S_DATA1
MOVLW 0X60
MOVWF PORTD
CALL S_DATA1
MOVLW 0X1E
MOVWF PORTD
CALL S_DATA1
MOVLW 0X01
MOVWF PORTD
CALL S_DATA1

HERE
GOTO HERE
```

```
SUBROTINES
S_CMD                ; FOR instructions CS1&CS2 must be activated
BSF PORTC,0          ;CS1=1
BCF PORTC,1          ;CS2=0
BCF PORTC,3          ;RS=0 (low = instruction data)
BSF PORTC,2

BCF PORTC,2
RETURN

S_DATA1              ; FOR DATA one of CS1&CS2 only must be activated
BSF PORTC,0          ;CS1=1
BCF PORTC,1          ;CS2=0
BSF PORTC,3          ;RS=1  (high = RAM data)
BSF PORTC,2
BCF PORTC,2
RETURN

S_DATA2              ; FOR DATA one of CS1&CS2 only must be activated
BCF PORTC,0          ;CS1=0
BSF PORTC,1          ;CS2=1
BSF PORTC,3          ;RS=1  (high = RAM data)
BSF PORTC,2
BCF PORTC,2
RETURN

END
```

4.2.2 Picbasic

Let's take the same example of the assembly, we want to draw an ellipse as in figure 8.
First the image must be converted into pixels (128X64 pixels maximum), see section 4.2.1
I made some symbols to facilitate the program as D is a symbol of port D and so on, I also made some variables as P to put page number into it and L for line number.
Five subroutines are used in this program:

1- "S_CMD" for sending command (instruction), put the instruction into D then call the subroutine.
2- "S_PAGE" for sending page number, put page no. into P then call the subroutine.
3- "S_LINE" for sending line number, put line no. into L then call the subroutine.
4- "S_DATA1" for sending data into 1st half of the LCD, put the instruction into D then call the subroutine (CS1 is activated and CS2 is deactivated).
5- "S_DATA2" for sending data into 2nd half of the LCD, put the instruction into D then call the subroutine (CS2 is activated and CS1 is deactivated).

For the subroutines 1&2&3, CS1&CS2 are both activated in order not to make subroutines for each half of the LCD.
For 2&3 you can put your page no. or line no. directly into P or L (ex: for page no. instruction is 1011 1XXX and page no. is written in the 3 Xs i.e. page 3 instruction = 1011 1011, but here write P=3 only).

Picbasic code using "Proton" program

```
Device 16F877
XTAL 4

TRISC=%00000000
TRISD=%00000000
PORTC=$00
PORTD=$00

Symbol D=PORTD       ;D=data
Symbol DI=PORTC.3    ;Data/Instruction (1=data & 0=instruction)
Symbol clk=PORTC.2   ;Enable of GLCD (clock -ve edge)
Symbol CS1=PORTC.1
Symbol CS2=PORTC.0
Dim P As Byte    ;page number (0:7)
Dim L As Byte    ; Line number (0:63)
```

```
D = $3f
GoSub S_CMD

P = 1
GoSub S_PAGE
L = 21
GoSub S_LINE
D = 0XC0
GoSub S_DATA1
D = 0X20
GoSub S_DATA1
D = 0X20
GoSub S_DATA1
D = 0X20
GoSub S_DATA1
D = 0XC0
GoSub S_DATA1

P=2
GoSub S_PAGE
L=19
GoSub S_LINE
D=0XF0
GoSub S_DATA1
D=0X0F
GoSub S_DATA1

L=26
GoSub S_LINE
D=$0F
GoSub S_DATA1
D=$F0
GoSub S_DATA1

P=3
GoSub S_PAGE
L=19
GoSub S_LINE
D=0X01
GoSub S_DATA1
D=0X1E
GoSub S_DATA1
D=0X60
GoSub S_DATA1
D=0X80
GoSub S_DATA1
D=0X80
GoSub S_DATA1
D=0X80
GoSub S_DATA1
D=0X60
GoSub S_DATA1
D=0X1E
GoSub S_DATA1
D=0X01
GoSub S_DATA1
Stop

S_DATA1:
CS1=0
CS2=1
DI=1
clk=1
clk=0
Return

S_DATA2:
CS1=1
CS2=0
```

```
DI=1
clk=1
clk=0
Return

S_CMD:
CS1=0
CS2=0
DI=0
clk=1
clk=0
Return

S_PAGE:
CS1=0
CS2=0
D = P + %10111000      ; x address (1011 1XXX) D = P + %10111000
DI=0
clk=1
clk=0
Return

S_LINE:
CS1=0
CS2=0
D = L + %01000000      ; y address (01XX XXXX) D = L + %01000000
DI=0
clk=1
clk=0
Return
```

There are a number of commands special for the KS0108 driver in the Proton program which can be used easily, these commands are **LINE, LINETO, BOX, CIRCLE, PLOT, UNPLOT** (refer to proton help).

First LCD type must be declared as follows:

```
LCD_TYPE = GRAPHIC        ;Use a Graphic LCD
```

Then each of the data bus, RS, E, CS1, CS2 and R/W must be declared

```
LCD_DTPORT = PORTD
LCD_RSPIN = PORTC.3
LCD_ENPIN = PORTC.2
LCD_CS1PIN = PORTC.1
LCD_CS2PIN = PORTC.0
```

These decelerations must be written before using any of the above commands.
R/W must be connected to the PIC and declared.
You can use a standard interfacing circuit with no need for declarations mentioned above, just file must be include at the beginning of the program.

```
INCLUDE "PROTON_G4.INT"
```

For example
We want to draw this picture shown in figure 9.

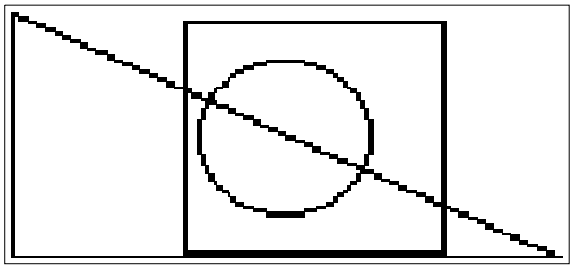

Figure 9: picture displayed on GLCD

By using interfacing circuit in figure 7.

Code

```
Device 16F877
LCD_TYPE = GRAPHIC
LCD_DTPORT = PORTD
LCD_RSPIN = PORTC.3
LCD_RWPIN = PORTC.4
LCD_ENPIN = PORTC.2
LCD_CS1PIN = PORTC.1
LCD_CS2PIN = PORTC.0

Line 1,0,0,127,63
LineTo 1,0,63
LineTo 1,0,0
Box 1,70,32,30
Circle 1,63,32,20

Stop
```

By using the standard interfacing circuit in figure 10.

Code

```
Device 16F877
Include "PROTON_G4.INT"

Line 1,0,0,127,63
LineTo 1,0,63
LineTo 1,0,0
Box 1,70,32,30
Circle 1,63,32,20

Stop
```

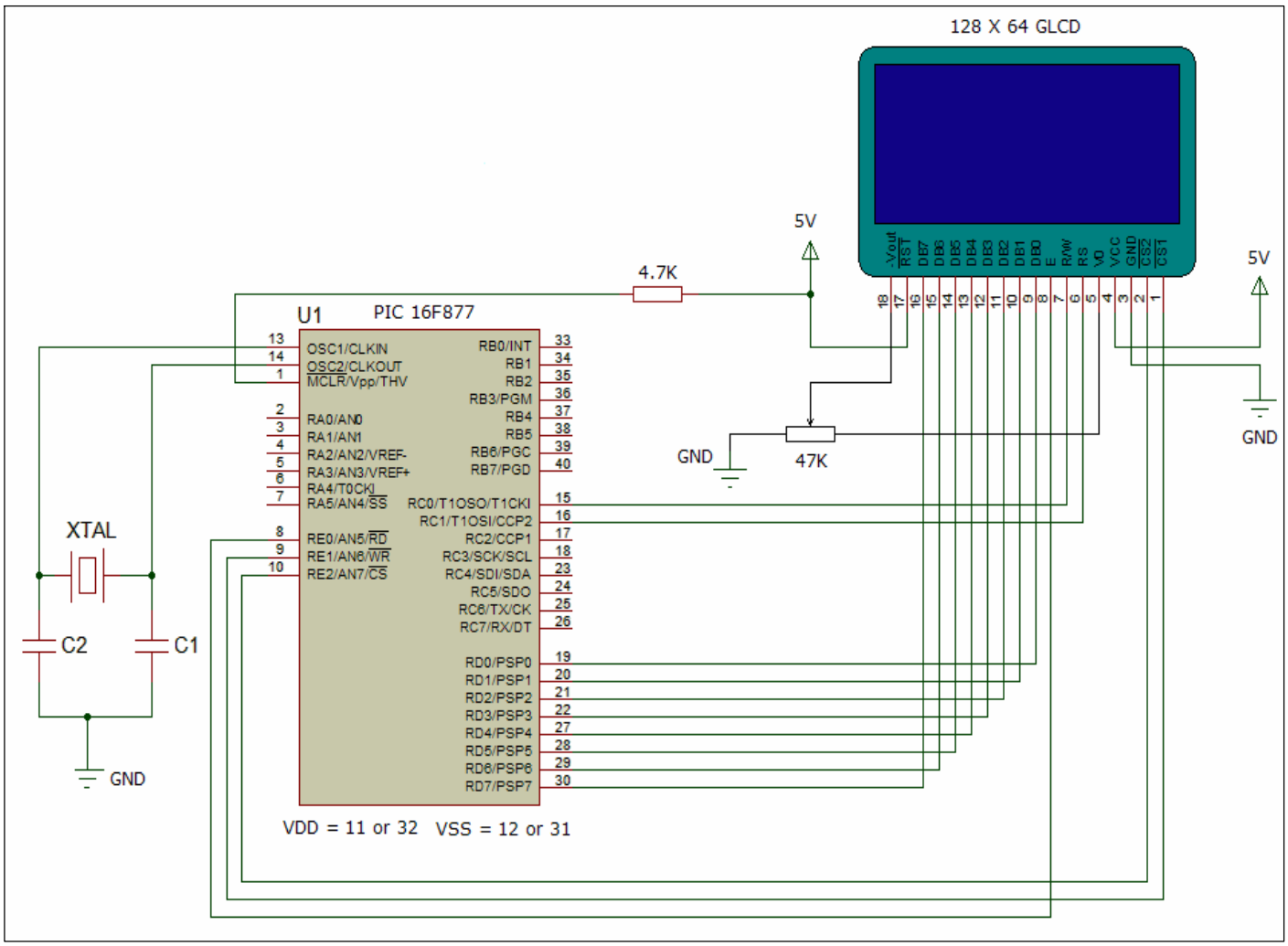

Figure 9: standard interfacing circuit

5. Application note

There must be a delay of at least 1/ LCD frequency in between operations or commands of the LCD (i.e. in two places; first place after setting DB0-7, CS, DI and R/W and before toggling enable high and second place between toggling enable high and toggling it low).
For the LM12864LFC the LCD frequency is not mentioned in datasheet so we can make a delay time of 1ms or 1.5ms to be in the safe side.
This timing can be one of the most common errors because if it is not delayed properly, a seemingly good instruction will not provide the desired result. This problem may not be appear in software simulation (as Proteus) but take care while implementing your hardware from this point.

6. References

[1] Lucas L. Delaney," Design of a Graphical LCD Driver and Educational LCD Primer", Design Project Report.
[2] LM12864LFC LCD user manual.
[3] Proton manual.

تم بحمد الله

bme_mabyad@yahoo.com