Library Routine for External LCD module
(for C Language)

1. Introduction

This is a general purpose LCD C language library module for PIC18xxx family of microcontrollers. This module configures the external LCD (XLCD), based on the Hitachi HD44780 LCD controller or equivalent. This module configures the I/O pins of the microcontroller, sets up the LCD for 4 or 8-bit mode and provides APIs for different LCD functions.

2. Module Features

This library module supports the following features (please refer Appendix A for different connections)

- Selecting the interface between the LCD module and the microcontroller, i.e. whether to select 8 or 4 bit interface.
- Selection for upper or lower nibble in case of 4-bit interface.
- Port selection for data transfer.
- Port and pin selection for the control signals.
- Facility to ground the R/W pin of the LCD (if read not required), which can help in saving a port pin for the microcontroller.
- Selection of the mode, whether the user wants delay or read busy flag in between commands.
- Selection of Blocking or non-Blocking functions.
- Configuring the parameters like single-line or two-lines, font selection, cursor on, blink on, etc.

3. List of Component Modules

`XLCD.P18.ex.txt`	This is main the test file developed to demonstrate the use of the library functions for the PIC18 family
`XLCD.c`	This is the LCD code implementation file.
`XLCD.h`	This is the header file, all functions and control ports are defined here. One needs to include this into their project

4. Using the Library Module in a Project

Please follow the steps below on how to use this library module in your project.

1. Use the Application Maestro to configure your code as required.
2. At the Generate Files step, save the output to the directory where your code project resides.
3. Launch the MPLAB, and open the project's workspace.
4. Verify that the Microchip language tool suite is selected (*Project>Select Language Tool-suite>Microchip C18Toolsuite*).
5. Got to *(Project>build options>project)* and select the path for linker, library etc.
6. In the Workspace view, right-click on the "Source Files" node. Select the "Add Files" option. Select XLCD.C and click **OK.**
7. Now right-click on the "Linker Scripts" node and select "Add Files". Add the appropriate linker file (`.lkr`) for the project's target microcontroller.
8. Add any other files that the project may require. Save and close the project.
9. To use the module in your application, invoke the functions as needed.

5. List of Shared Parameters

Shared Functions

XLCDInit()	It is used to initialize the LCD module according to the Application Maestro options.
XLCDCommand(Command)	It sends clocking signal and instructions to the LCD.
XLCDPut(data)	It sends the clocking signal and data to be displayed to the LCD.
XLCDIsBusy()	Reads the Busy Flag status from the LCD module.
XLCDGet()	It reads the data from the present address in the LCD.
XLCDL1home()	Points to the first address location of line one of the LCD.
XLCDL2home()	Points to the first address location of line two of the LCD
XLCDClear()	Clears the DDRAM content of the LCD and points to the 00 address location.
XLCDReturnHome()	Points to the 00 address location, the DDRAM content remains unchanged.
XLCDGetAddr()	Reads DDRAM Address
XLCDPutRomString(addr)	Displays String in Program memory
XLCDPutRamString(addr)	Displays String in Data memory

6. Functions

Function	`XLCDInit`
Pre-conditions	None
Overview	This is the initialization routine which initializes the LCD module like, which port to use for data the transmission and which port to use for control signal, which are taken from Application Maestro options. It also takes options such as font the selection, number of lines, cursor on, blink on, etc, from the Application Maestro
Input	None
Output	None
Side Effects	None

Function	XLCDCommand
Pre-conditions	User needs to pass the command.(in non blocking mode, the user may require to call the XLCDIsBusy before calling this function to ensure if the LCD module is free).
Overview	It sends clocking signal and instructions to LCD.It checks the busy flag or the call delay before sending the instruction to make sure that the LCD module is free (if Blocking is selected).
Input	None
Output	None
Side Effects	None

Function	XLCDPut
Pre-conditions	User needs to pass the data to be displayed. (in non-blocking mode the user may require to call the XLCDIsBusy before calling this function to ensure if the LCD module is free).
Overview	It sends clocking signal and data to be displayed to the LCD.It checks the busy flag or the call delay before sending the instruction to make sure that the LCD module is free (if Blocking is selected).
Input	None
Output	None
Side Effects	None

Function	XLCDGetAddr
Pre-conditions	None
Overview	Reads the DDRAM address
Input	None
Output	Wreg
Side Effects	None

Function	XLCDIsBusy
Pre-conditions	None
Overview	User must call this function in Non-blocking mode. It reads the busy flag of the LCD .In Non-blocking mode, this function returns 1 in W register if the module is busy, else it returns with 0.
Input	None
Output	W reg
Side Effects	None

Function	XLCDGet
Pre-conditions	In non blocking mode the user may require to call XLCDIsBusy before calling this function, to ensure that theLCD module is free
Overview	Reads the data from DDRAM present address and return the data in w register
Input	None
Output	W reg
Side Effects	None

Function	XLCDL1home
Pre-conditions	In non blocking mode the user may require to call XLCDIsBusy before calling this function to ensure that the LCD module is free
Overview	It points to the line one 00 address of the DDRAM
Input	None
Output	None
Side Effects	None

Function	XLCDL2home
Pre-conditions	In non blocking mode the user may require to call XLCDIsBusy before calling this function to ensure that the LCD module is free
Overview	It points to the line two first address of the DDRAM
Input	None
Output	None
Side Effects	None

Function	XLCDClear
Pre-conditions	In non blocking mode the user may require to call XLCDIsBusy before calling this function to ensure that the LCD module is free
Overview	It clears the DDRAM content and point to 0 address location
Input	None
Output	None
Side Effects	None

Function	XLCDReturnHome
Pre-conditions	In non blocking mode the user may require to call XLCDIsBusy before calling this function to ensure that the LCD module is free
Overview	It point to the 0 address location but DDRAM content remain unchanged
Input	None
Output	None
Side Effects	None

Function	XLCDPutRomString
Pre-conditions	None
Overview	Displays String in Program memory
Input	Start address
Output	None
Side Effects	None

Function	XLCDPutRamString
Pre-conditions	None
Overview	Displays String in Data memory
Input	Start address
Output	None
Side Effects	None

Note:

- The user should make the PORTA pins digital, if used as control signal or for data transmission.
- The user must check the port availability before using it (for example the upper nibble of PORTA and PORTG may not be used for data transmission).
- If non-blocking mode is selected the user must call the XLCDIsBusy function and check the busy condition before any command. This is to ensure that the LCD module is free. In blocking mode, the busy condition is checked inside the commands by calling delay or by polling for busy flag.
- The user can save a micro-controller pin by grounding the R/W pin of the external LCD (as shown in Figure 1 and Figure 3 of Appendix A). But by doing so the user will not be able to call any read command, like XLCDReadData, XLCDIsBusy, etc.
- The 'C' library routines for LCD does not provide functions for delays. The user is expected to write delay functions like **XLCDDelay15ms ()** used in XLCD init, **XLCDDelay4ms ()** used in XLCD init, **XLCDDelay500ns ()** used in command instructions and **XLCDDelay***()(the user is required to provide XLCDDelay() only if the mode selected is delay)*in the project .

8. Appendix A

8-bit Interface:

Here in **Figure-1**, a micro controller port pin can be saved if the LCD RW pin is grounded. But if RW pin is grounded reading data or reading busy flag from the LCD is not possible.

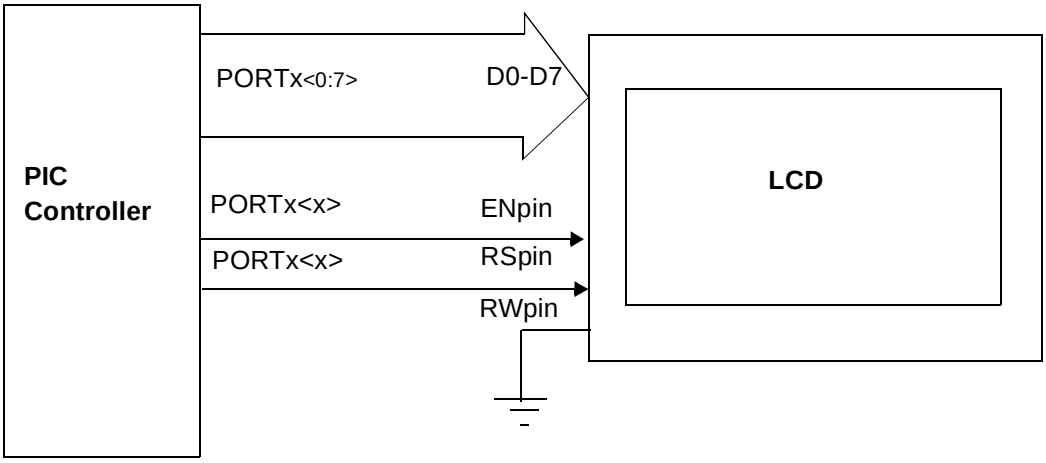

Figure-1:(RW pin grounded, no read back, 8-Bit interface)

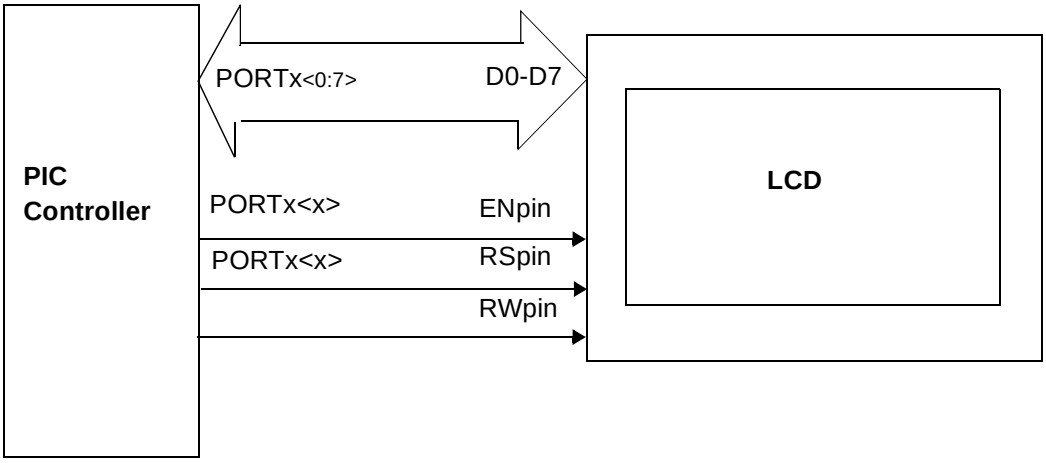

Figure-2:(RW pin not grounded, read back possible, 8-Bit interface)

4-bit Interface:
Data transmission can be through upper nibble or lower nibble

Here in **Figure-3**, a micro controller port pin can be saved if the LCD RW pin is grounded. But if RW pin is grounded reading data or reading busy flag from the LCD is not possible.

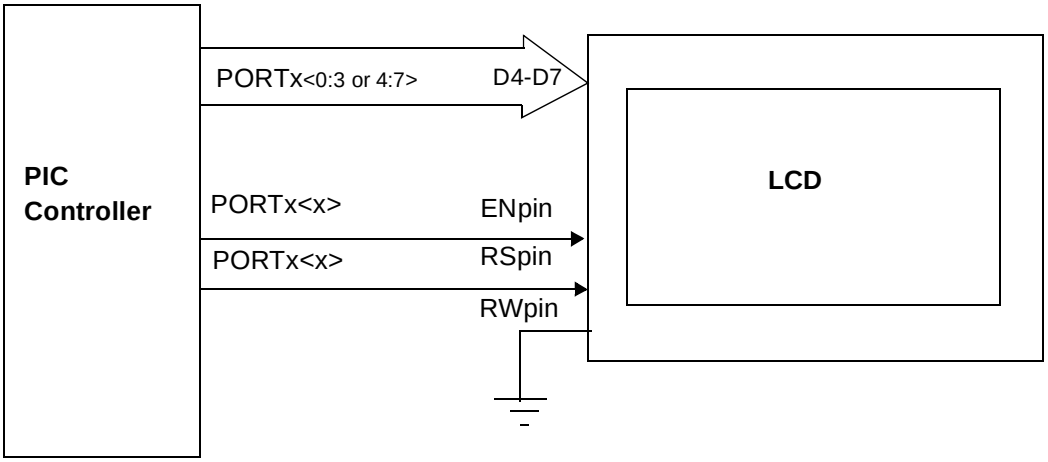

Figure-3:(RW pin grounded, no read back, 4-Bit interface)

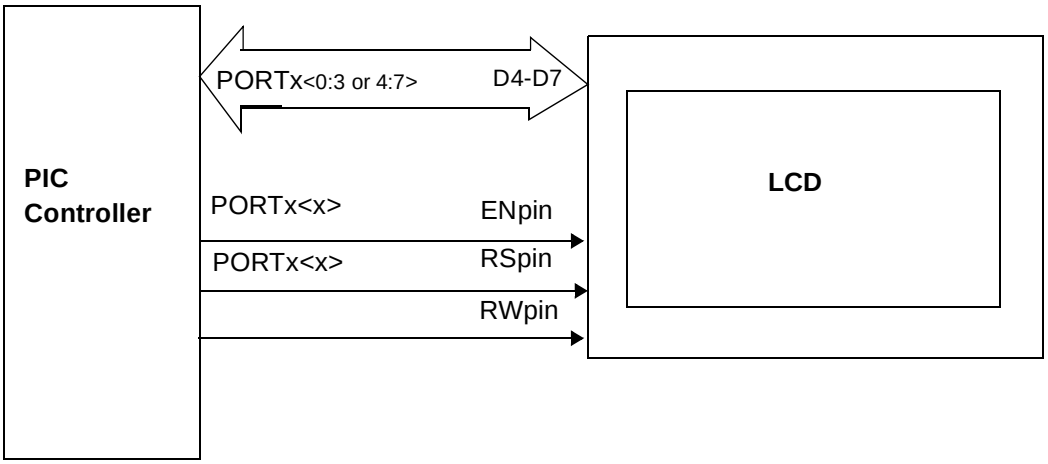

Figure-4:(RW pin not grounded, read back possible, 4-Bit interface