PmodCLP Demo Reference Manual

Revision: February 14, 2012

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Introduction

The PmodCLP is a 16x2 character LCD module that uses two Pmod connectors to present a 3.3V, 8-bit parallel data interface to system boards. It is based on a Sunlike LCD panel that uses a Samsung KS0066 (or equivalent) LCD controller. The module can be attached to any number of Digilent system boards to create a character LCD subsystem. This document provides a demonstrational overview of the driver library(ClpLib) included for operation of this module.

Required Hardware:

1. Cerebot 32Mx4cK or Cerebot 32Mx4
2. PmodCLP – J1 plugged into JA and J2 plugged into JB.

Note: Project was created using MPLAB v8.83

Overview

The module is capable of displaying any one of more than 200 predefined characters in each of the 32 display locations (organized as 16 characters on two rows). Most characters use ASCII codes (see the Samsung KS0066 data sheet for a complete list of character codes). The module can also execute a variety of instructions, such as erasing specific characters, setting different display modes, scrolling, and displaying user-defined characters.

Library Operation

Library Interface

The header file ClpLib_config.h is a configuration file to setup the target platform for the specific PMOD connections being used (default: JA and JB).

The header file ClpLib.h contains the declarations for the actual functions that are available to the user for this module. Actual definitions for these functions are found within ClpLib.c.

For more information see PmodCLP Library Reference Manual.

Demo Documentation

The application implements (in a circular manner) a number of steps, each demonstrating the use of some CLP functions. Buttons are used to trigger some actions and to move through the steps.

No.	Step	Actions	Demonstrates
1.	Welcome screen	Press Any button to continue	ClpWriteStringAtPos - Write to Clp
2.	Backlight On / Off	BTN2: toggle backlight BTN1: continue	ClpWriteStringAtPos - Write to Clp ClpSetBackLight - Set Backlight
3.	Display Shift left / right	BTN2: shift left BTN1: shift right Double buttons action to continue	ClpWriteStringAtPos - Write to Clp ClpDisplayShift – shift display ClpDisplayClear – clear display
4.	Display On / Off	BTN2: toggle display BTN1: continue	ClpWriteStringAtPos - Write to Clp ClpDisplaySet - Set Display On / Off ClpReturnHome – Cursor Home
5.	Cursor On / Off	BTN2: toggle cursor BTN1: continue	ClpWriteStringAtPos - Write to Clp ClpDisplaySet - Set cursor On / Off
6.	Cursor Blink On / Off	BTN2: toggle cursor blink BTN1: continue	ClpWriteStringAtPos - Write to Clp ClpDisplaySet - Set Blink On / Off
7.	Cursor shift left / right	BTN2: cursor shift left BTN1: cursor shift right Double buttons action to continue	ClpWriteStringAtPos - Write to Clp ClpCursorShift – shift cursor ClpDisplayClear – clear display
8.	User defined character	Any button to continue	ClpWriteBytesAtPosCgram – write user defined character data to CGRAM ClpWriteDataByte – send CGRAM character to CLP

Features:

- Buttons are debounced.
- Recognizes three button actions:
  1. BTN1 pressed
  2. BTN2 pressed
  3. Releasing one button while the other is pressed (double buttons action)

Project Files

File	Containing
main.c	Main application file. Contains main application loop, interface ISR, buttons processing. As it implements application functionality, is modified for each application.
util.h	Common Utility Procedures. This is a standard Digilent file, also used in other applications. It was not modified for this application.
util.c	Common Utility Procedures. This is a standard Digilent file, also used in other applications, implementing some led and wait functions. It was not modified for this application.
stdtypes.h	Digilent Standard Type Declarations. This is a standard

	Digilent file, also used in other applications. It was not modified for this application.

Resources Used

Timer5 configuration is done using Plib macros in AppInit function. SFRs approach is also provided in the commented code.

Plib macro used:

OpenTimer5(T5_ON | T5_SOURCE_INT | T5_PS_1_8, 99);
ConfigIntTimer5(T5_INT_ON | T5_INT_PRIOR_7 | T5_INT_SUB_PRIOR_3);

Meaning:

- T5_ON – timer 5 is ON.
- T5_PS_1_8 (Prescaler 1/8): so the frequency is 1/8 of Peripheral bus freq, which is 1/8 of SYSCLK (65 MHz) = 1 MHz
- Period = 99, so Timer period = (99 + 1) * 1/1MHz = 100 us
- T5_SOURCE_INT : Timer5 triggers interrupt
- T5_INT_ON – T5 interrupt is ON
- T5_INT_PRIOR_7, T5_INT_SUB_PRIOR_3 - timer interrupt priority level 7, subpriority level 3

More information about Timer5 is shown in Timer5Handler.

Functions defined in main.c

void __ISR(_TIMER_5_VECTOR, ipl7) Timer5Handler(void)

Parameters:
None

Interrupt service routine for Timer 5interrupt. Timer 5 is used to perform software debouncing of the on-board buttons.

int main(void)

Parameters:
none

Main program module. Performs basic board initialization (by calling DeviceInit and AppInit) and then enters the main program loop, where steps are cycled. For every step, WaitUntilBtnPressed is called in order to wait for the buttons.

void WaitUntilBtnPressed(BOOL *pfBtn1Process, BOOL *pfBtn2Process)

Parameters:

pfBtn1Process	used as output parameter fTrue if the button 1 causes processing fFalse if the button 1 does not cause processing.
pfBtn2Process	used as output parameter fTrue if the button 2 causes processing fFalse if the button 2 does not cause processing.

Waits in a loop until a button action is detected, then configures the output parameters and returns. It reads flags set by Timer5Handler, detecting the following actions:

- single button: one of BTN1, BTN2 is pressed - *pfBtn1Process, *pfBtn2Process are set to fTrue
- double buttons: one of BTN1, BTN2 is released while the other is pressed – both *pfBTn1Process and *pfBtn2Process are set to fTrue

This function is called from main program loop, so its loop is the place where other functionality can be placed.

void DeviceInit(void)

Parameters:
none

This routine initializes the on-chip and on-board peripheral devices to their default state. I t also calls ClpPinsConfigure() toconfigure CLP pins.

void AppInit(void)

Parameters:
none

This routine performs application specific initialization. It configures devices and global variables for the application. It configures TIMER5 and its interrupt using Plib macros. SFRs approach is also provided in the commented code.