1300 Henley Court
Pullman, WA 99163
509.334.6306
www.digilentinc.com

chipKIT™ LCDP Library Reference Manual

Revised April 9, 2014

Overview

The Digilent PmodCLP contains a 16x2 characters display. Digilent provides a driver library for this module. This document provides an overview of the operation of this driver library and describes the functions that control its programming interface.

The module is connected to a board using two Pmod connectors that group both command and data pins. The library implements read and write cycles to communicate with the display, and offers user access to all of the LCD functionality.

For more information about the hardware and functional interface of the PmodCLP, refer to the PmodCLP reference manual available for download from the Digilent website www.digilentinc.com.

1 Library Operation

1.1 Library Interface

The header file LCDP.h defines the functions of the PmodCLP driver. The library is accessed via the methods defined for the LCDP object class. In order to use this library, the user has to instantiate one library object. If more CLP modules are connected to a board, more instances of the LCDP class have to be defined; one for each module.

1.2 Display and Communication Initialization

The CLP display initialization sequence is called from the begin function. The user is responsible to call this function prior to accessing any other library functionality.

The PmodCLP is physically inserted in two Pmod connectors. When the begin function is called, the pin numbers for specific PmodCLP command and data pins must be provided as arguments. The library defines some standard pin number configurations for each Pmod pair of connectors, so in this case the user only has to select one of these configurations. Table 1 below shows the possible connection configurations, indicating the Arduino pins involved for backlight, RS, RW, Enable control pins, and data pins d0 – d7.

Pmod Connectors	Name	Arduino Pins: backlight, RS, RW, Enable, d0, d1, d2, d3, d4, d5, d6, d7
JA, JB	LCDP_JA_JB_ARGS	15, 12, 13, 14, 0, 1, 2, 3, 4, 5, 6, 7
JB, JC	LCDP_JB_JC_ARGS	23, 20, 21, 22, 8, 9, 10, 11, 12, 13, 14, 15
JB, JC	LCDP_JC_JD_ARGS	31, 28, 29, 30, 16, 17, 18, 19, 20, 21, 22, 23
JD, JE	LCDP_JD_JE_ARGS	39, 36, 37, 38, 24, 25, 26, 27, 28, 29, 30, 31
JE, JF	LCDP_JE_JF_ARGS	47, 44, 45, 46, 32, 33, 34, 35, 36, 37, 38, 39
JH, JJ	LCDP_JH_JJ_ARGS	63, 60, 61, 62, 48, 49, 50, 51, 52, 53, 54, 55
JJ, JK	LCDP_JJ_JK_ARGS	71, 68, 69, 70, 56, 57, 58, 59, 60, 61, 62, 63

Table 1. Possible standard Pmod connections

1.3 LCD Functionality

The library is capable of performing the following display management functions:

- Clear the display
- Display on/off
- Cursor on/off
- Cursor blink on/off
- Display scroll left/right
- Cursor scroll left/right
- Set current position
- Return home
- Display text at a specific position
- Define user characters
- Display user characters at a specific position

2 LCD Library Functions

2.1 Errors

Table 2 below shows the possible errors returned by some of the LCDP functions:

Value	Name	Description
0	LCDP_ERR_SUCCESS	The action completed successfully
0x20	LCDP_ERR_UCHAR_POSITION_INVALID	The user character position is not correct
0x80	LCDP_ERR_ARG_ROW_RANGE	The row index is not valid
0x40	LCDP_ERR_ARG_COL_RANGE	The column index is not valid

Table 2. List of errors

2.2 Communication Configuration Functions

void begin (uint8_t bkl, uint8_t rs, uint8_t rw, uint8_t enable, uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3, uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7)

Parameters:
- uint8_t bkl
  - The Arduino pin number corresponding to backlight pin
- uint8_t rs
  - The Arduino pin number corresponding to RS pin
- uint8_t rw
  - The Arduino pin number corresponding to RW pin
- uint8_t enable
  - The Arduino pin number corresponding to Enable pin
- uint8_t d0, d1, d2, d3, d4, d5, d6, d7
  - The 8 pins corresponding to data pins

This function initializes the library and declares the pins where specific control and data pins are connected. Instead of specifying each pin, the user may choose from a list of standard pre-defined connections corresponding to possible board Pmod connectors.

See the Display and Communication Initialization section for more details and for the list of pre-defined standard connections and the pins involved.

This function must be called before any other functions in the library are called.

2.3 Display Management Functions

void DisplayClear()

Parameters:
- None

This function clears the display and returns the cursor position home, on the first line and column.

void ReturnHome ()

Parameters:
- None

This returns the cursor position home, on the first line and column.

void SetDisplay(bool fDisplayOn)

Parameters:
- bool fDisplayOn
  - parameter indicating how the display will be set:
    - True: if the display will be set ON
    - False: if the display will be set OFF

This function turns the display on or off, according to the user's selection.

void SetCursor(bool fCursorOn)

Parameters:

- bool fCursorOn
    - Parameter indicating how the Cursor will be set:
        - True: if the Cursor will be set ON
        - False: if the Cursor will be set OFF

This function turns the cursor on or off, according to the user's selection.

void SetBlink(bool fBlinkOn)

Parameters:

- bool fBlinkOn
    - Parameter indicating how the Blink will be set:
        - True: if the Blink will be set ON
        - False: if the Blink will be set OFF

This function turns the Blink on or off, according to the user's selection.

void SetBacklight(bool fBacklightOn)

Parameters:

- bool fBacklightOn
    - Parameter indicating how the Backlight will be set:
        - True: if the Backlight will be set ON
        - False: if the Backlight will be set OFF

This function turns the backlight on or off, according to the user's selection. Note that there are CLP Pmods that do not have backlight functionality. Using this function for this type of module will have no effect.

uint8_t LCDP::SetPos(uint8_t idxLine, uint8_t idxCol)

Parameters:

- uint8_t idxLine
    - The line where the position will be set
- uint8_t idxCol
    - The column where the position will be set

Return value:

- *LCDP_ERR_SUCCESS*
    - The action completed successfully
- A combination of the following errors(OR'ed):
    - *LCDP_ERR_ARG_ROW_RANGE*
        - The row index is not valid
    - *LCDP_ERR_ARG_COL_RANGE*
        - The column index is not valid

This function sets the corresponding LCD position. This is used for write position and cursor position. If the position set are invalid (outside of the display), errors are returned.

uint8_t WriteStringAtPos(char *szLn, uint8_t idxLine, uint8_t idxCol)

Parameters:

- char *szLn
  - String to be written to LCD
- uint8_t idxLine
  - The line where the string will be displayed
- uint8_t idxCol
  - The column where the string will be displayed

Return value:

- *LCDP_ERR_SUCCESS*
  - The action completed successfully
- A combination of the following errors(OR'ed):
  - *LCDP_ERR_ARG_ROW_RANGE*
    - The row index is not valid
  - *LCDP_ERR_ARG_COL_RANGE*
    - The column index is not valid

The function writes the specified string at the specified position (line and column). It sets the corresponding write position and then writes data bytes when the device is ready. Strings that span over the end of the row are trimmed so that they fit in the row. If the position is invalid (outside of the display), errors are returned.

void DisplayShift(bool fRight)

Parameters:

- bool fRight
  - Parameter indicating the direction of the display shift
    - True: in order to shift the display right
    - False: in order to shift the display left

This function shifts the display one position right or left, depending on the fRight parameter.

void CursorShift(bool fRight)

Parameters:

- bool fRight
  - Parameter indicating the direction of the cursor shift
    - True: in order to shift the cursor right
    - False: in order to shift the cursor left

This function shifts the cursor one position right or left, depending on the fRight parameter.

uint8_t DefineUserChar(uint8_t *pBytes, uint8_t charPos)

Parameters:

- uint8_t *pBytes
    - Pointer to the string that contains the 8 bytes definition of the character
- uint8_t bCharNo
    - The position of the character to be saved in the memory

Return value:

- *LCDP_ERR_SUCCESS*
    - The action completed successfully
- LCDP_ERR_UCHAR_POSITION_INVALID
    - The user character position is not within 0 - 7

This function writes the specified number of bytes to CGRAM starting at the specified position. It sets the corresponding write position and then writes the data bytes when the device is ready. If the user character position is not within 0 - 7 range, errors will be returned.

uint8_t LCDP::WriteUserCharsAtPos(uint8_t* rgCharPos, uint8_t bNoChars, uint8_t idxLine, uint8_t idxCol)

Parameters:

- uint8_t* rgCharPos
    - An array containing the index (position) of the user characters to be displayed
- uint8_t bNoChars
    - The number of user defined characters to be displayed
- uint8_t idxLine
    - The line where the string will be displayed
- uint8_t idxCol
    - The column where the string will be displayed

Return value:

- *LCDP_ERR_SUCCESS*
    - The action completed successfully
- A combination of the following errors(OR-ed):
    - *LCDP_ERR_ARG_ROW_RANGE*
        - The row index is not valid
    - *LCDP_ERR_ARG_COL_RANGE*
        - The column index is not valid
    - LCDP_ERR_UCHAR_POSITION_INVALID
        - The user character position is not within the accepted range (0 – 7)

This function displays one or more user defined characters at the specified positions on the LCD. If the position set or the user character position is not correct, errors will be returned.

3 Library Usage

This section of the document describes the library usage.

- PmodCLP has to be attached to the board using a set of board pins. Normally, it will be connected to a pair of Pmod connectors. See Display and Communication Initialization for more details and for the list of pre-defined standard connections and the pins used.
- Copy the library files according to the README.txt file.
- In the sketch, include the LCDP library header file:
  ```
  #include <LCDP.h>
  ```
- In the sketch, instantiate one library object called, for example, myLCDP:
  ```
  LCDP myLCDP;
  ```
- In the sketch, use library functions by calls such as:
  ```
  MyLCDP.WriteStringAtPos(“Digilent”, 1, 0);
  ```

4 LCDP Demo Application

The demo application was created in order to exemplify the functionality of some of the module’s library functions, as well as the way the library is integrated in the MPIDE work environment.

The application uses the two on-board buttons to control the display. It also implements, in a circular manner, a number of steps; each of them demonstrating the use of a library function. The buttons are used to trigger actions and move to the next step of the application. Table 3 summarizes the actions performed and the functions used from the library:

No.	Step	Actions	Demonstrates
1	Welcome screen	Press Any Button to Continue	WriteStringAtPos: Write to CLP DisplayClear: clear display
2	Backlight On / Off	BTN2: toggle backlight BTN1: continue	WriteStringAtPos: Write to CLP DisplaySet: Set Backlight and Display
3	Display Shift left / right	BTN2: shift left BTN1: shift right Double buttons action to continue	WriteStringAtPos: Write to CLP DisplayShift: shift display DisplayClear: clear display
4	Cursor on/Off	BTN2: toggle cursor BTN1: continue	WriteStringAtPos: Write to CLP CursorModeSet: Set Cursor On / Off
5	Blink On / Off	BTN2: toggle blink BTN1: continue	WriteStringAtPos: Write to CLP CursorModeSet: Set Blink On / Off
8	User defined character	Any button to continue	WriteStringAtPos: Write to CLP DefineUserChar: Define the user characters WriteUserCharsAtPos: Display on the CLP the defined user characters

Table 3. Functions used in Demo and corresponding actions

This demo performs the following operations:
- In the setup() function:
  - Initializes the LCDP library
- In the loop() function:

  - Implements the demo steps
  - After each step the function WaitUntilBtnPressed is called to verify if a button, or both, were pressed, waiting for that action in case they weren't.
- In the WaitUntilBtnPressed function
  - Waits until a button is pressed and returns the state of the buttons