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

chipKIT™ LCDS Library Reference Manual

Revised April 7, 2014

Overview

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

The display is a serial device that is accessed via an SPI, I2C, or UART interface. It is, however, a write-only device from the user's point of view because access to status or data coming from the display is not provided. The user can only transmit commands or data to be displayed to the LCD controller which will forward, interpret, and dispatch the commands to the display. For more information about the hardware interface of the PmodCLS, refer to the PmodCLS reference manual available for download from the Digilent website www.digilentinc.com.

1 Library Operation

1.1 Library Interface

The header file LCDS.h defines the functions of the PmodCLS driver. The library is accessed via the methods defined for the LCDS object class. In order to use this library, the user has to instantiate one library object. If more than one CLS module is connected to a board and the application needs to use them all, then more instances of the LCDS class have to be defined; one for each module. The PmodCLS allows multiple communication interfaces but the library has only implemented one: the SPI interface.

1.2 Display and Communication Initialization

The CLS display has a power on/power off sequence that should be followed. That sequence is not under the control of the user since it is being managed by the firmware in the LCD controller at a lower level. However, before using the device, some initialization procedure has to be run for the communication interface.

The LCDS.Begin (uint8_t bAccessType) function selects the communication interface and port and configures it. In the case of the SPI interface, it will attach the SS pin to the class methods, configure it as output, and then it will set the SPI frequency through the setSpeed(uint8_t Spd) method. Another operation is to set the communication mode for the SPI interface, using the setMode(uint8_t bMode) method. The begin function initializes the previously mentioned parameters to their default values: frequency: 1MHz, communication mode: 0.

1.3 Display Management Mode

The module is capable of performing a set of commands that are sent using escape sequences via any of the three mentioned interfaces. Some of the functions are:

- Clear the display.
- Cursor blinks on/off.
- Cursor on/off.
- Display on/off.
- Display scroll left/right.
- Set cursor to a specified position (line and column).

1.4 Characters Handling Mode

The mode refers to sending and displaying characters on the module and the way they can be controlled. The two mentioned modes can interleave according to the user's needs. Functions for handling characters, or user defined characters, are called along to the ones for displaying text/character at a certain position.

1.5 EEPROM Control Mode

There are some specific functions for working with the EEPROM memory of the microcontroller found on the CLS module. They allow saving characters tables in EEPROM, saving communication modes, or the settings for cursor and display.

1.6 Custom Characters Mode

User defined characters are also called custom characters and they can be defined either in the RAM memory or in the EEPROM memory. Four tables each containing eight custom characters can be defined, one in the RAM memory, the other three in the EEPROM. They can be handled at the user's discretion according to the needs: copied from RAM to EEPROM or vice-versa. The way a custom character is defined is presented in the PmodCLS manual.

2 LCDS Library Functions

2.1 Errors

Table 1 below shows the possible errors returned by the LCDS functions:

Value	Name	Description
0	LCDS_ERR_SUCCESS	The action completed successfully
1	LCDS_ERR_ARG_ROW_RANGE	The argument is not within 0, 2 range for rows
2	LCDS_ERR_ARG_COL_RANGE	The argument is not within 0, 39 range
3	LCDS_ERR_ARG_ERASE_OPTIONS	The argument is not within 0, 2 range for erase types
4	LCDS_ERR_ARG_BR_RANGE	The argument is not within 0, 6 range
5	LCDS_ERR_ARG_TABLE_RANGE	The argument is not within 0, 3 range for table selection
6	LCDS_ERR_ARG_COMM_RANGE	The argument is not within 0, 7 range
7	LCDS_ERR_ARG_CRS_RANGE	The argument is not within 0, 3 range for cursor modes
8	LCDS_ERR_ARG_DSP_RANGE	The argument is not within 0, 3 range for display settings types
9	LCDS_ERR_ARG_POS_RANGE	The argument is not within 0, 8 range for characters position in the memory

Table 1. List of errors.

2.2 Communication Configuration Functions

void Begin (uint8_t accessType)

Parameters:

- accessType
  - Parameter for selecting the communication port chosen for the application. Can be one of the parameters from Table 2 below:

Value	Name
0	PAR_ACCESS_SPI0
1	PAR_ACCESS_SPI1

Table 2. Values for accessType parameter of Begin function.

This function initializes the driver and configures the communication interface used for controlling the display. 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 DisplaySet(boolean setDisplay, boolean setBckl)

Parameters:
- setDisplay
  - Boolean parameter through which the display is set on or off.
- setBckl
  - Boolean parameter through which the backlight is set on or off.

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

void CursorModeSet(boolean setCursor, boolean setBlink)

Parameters:
- setCursor
  - Boolean parameter through which the cursor is set on or off.
- setBlink
  - Boolean parameter through which the blink option is set on or off.

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

uint8_t DisplayScroll(boolean fDirection, uint8_t idxCol)

Parameters:
- fDirection
  - Parameter for selecting the direction of the scroll operation; true for scroll right, false for left.
- idxCol
  - Parameter for the number of columns to be shifted right or left.

Return value:
- uint8_t
  - LCDS_ERR_SUCCESS
    - The action completed successfully.
  - LCDS_ERR_ARG_COL_RANGE
    - The argument is not within 0 - 39 range.

This function scrolls the display left or right with a specified number of columns

void SaveCursor()

Parameters:

- None.

This function saves the current position of the cursor.

void RestoreCursor()

Parameters:

- None.

This function restores the previously saved position of the cursor.

void DisplayMode(boolean charNumber)

Parameters:

- charNumber
  - Parameter for selecting the wrapping type of the line; to 16 or 40 characters.

When the charNumber is set to true value, the command for wrapping the line to 16 characters is sent; when the parameter is false, the command for wrapping the line to 40 characters will be sent to the display. This function wraps the line at 16 or 40 characters.

void Reset()

Parameters:

- None.

This function resets the PmodCLS.

2.4 Character Functions

The following functions use the character column and row number for the cursor position. The row and column numbers are 0 based. Therefore, the column numbers will be in the range 0 – (N-1) and the row numbers will be in the range 0 – (M-1), where N and M are the number of columns and the number of rows: N = 39 and M = 2.

uint8_t SetPos(uint8_t idxRow, uint8_t idxCol)

Parameters:

- idxRow
  - Horizontal character position (column).

- idxCol
    - Vertical character position (row).

Return value:
- uint8_t
    - LCDS_ERR_SUCCESS
        - The action completed successfully.
    - a combination of the following errors (OR-ed):
        - LCDS_ERR_ARG_COL_RANGE
            - The argument is not within 0 - 39 range.
        - LCDS_ERR_ARG_ROW_RANGE
            - The argument is not within 0 - 2 range for rows.

The function sets the character cursor position to the specified location.

Uint8_t WriteStringAtPos(uint8_t idxRow, uint8_t idxCol, char* strLn)

Parameters:
- idxRow
    - The line/row where the string is written: 0 or 1.
- idxCol
    - The start column for the string to be written:0 to 39.
- strLn
    - Pointer to the first element of the string to be written.

Return value:
- uint8_t
    - LCDS_ERR_SUCCESS
        - The action completed successfully.
    - a combination of the following errors (OR-ed):
        - LCDS_ERR_ARG_COL_RANGE
            - The argument is not within 0 - 39 range.
        - LCDS_ERR_ARG_ROW_RANGE
            - The argument is not within 0 - 2 range for rows.

This function writes a string on the display at a specified position. If the length of the text is greater than the maximum number of characters from a line, then it will be truncated to that number.

Uint8_t EraseInLine(uint8_t eraseParam)

Parameters:
- eraseParam
    - The selection of erasing operation: from current position, entire line, from the beginning of the line
        - 0 = current position to end of line.
        - 1 = start of line to current position.
        - 2 = entire line.

Return value:

- uint8_t
  - LCDS_ERR_SUCCESS
    - The action completed successfully.
  - LCDS_ERR_ARG_ERASE_OPTIONS
    - The argument is not within 0 – 2 range for erase types.

This function erases characters from a line, according to the parameter set: from the beginning of the line until the current position of the cursor, from the current position to the end of line, or entire line.

void EraseChars(uint8_t charsNumber)

Parameters:

- charsNumber
  - The number of characters to be erased.

This function erases a number of characters starting at the current position.

2.5 EEPROM functions

Parameters:

- addrEeprom
  - The location where the TWI address is saved.

This function saves a TWI address to an EEPROM memory location. Be aware that current implementation of the library does not provide I2C connection functionality.

Uint8_t SaveBR(uint8_t baudRate)

Parameters:

- baudRate
  - The baud rate value to be saved in the EEPROM memory. This is a numeric value corresponding to different allowed frequency values:

Value	Frequency
0	2400
1	4800
2	9600
3	19200
4	38400
5	76800

Table 3. Values for baudRate parameter of Uint8_t SaveBR function.

Return value:

- uint8_t
  - LCDS_ERR_SUCCESS
    - The action completed successfully.
  - LCDS_ERR_ARG_BR_RANGE
    - The argument is not within 0, 6 range.

This function saves the baud rate value to an EEPROM memory location.

Uint8_t CharsToLcd(uint8_t charTable)

Parameters:

- charTable
  - The characters table index to program into LCD in EEPROM or RAM.

Return value:

- uint8_t
  - LCDS_ERR_SUCCESS
    - The action completed successfully.
  - LCDS_ERR_ARG_TABLE_RANGE
    - The argument is not within 0 - 3 range for table selection.

This function programs a characters table from EEPROM or RAM into LCD. According to the selected table (0, 1, 2 for EEPROM, 3 for RAM), the characters from the specified table will be displayed on the LCD. Please see the module's manual for more details.

Uint8_t SaveRamtoEeprom (uint8_t charTable)

Parameters:

- charTable
  - The characters table index to save in EEPROM from RAM.

Return value:

- uint8_t
  - LCDS_ERR_SUCCESS
    - The action completed successfully.
  - LCDS_ERR_ARG_TABLE_RANGE
    - The argument is not within 0 - 3 range for table selection.

This function saves a characters table in EEPROM from RAM.

Uint8_t LdEepromToRam(uint8_t charTable)

Parameters:

- charTable
  - The characters table index to save in RAM from EEPROM.

Return value:

- uint8_t
  - LCDS_ERR_SUCCESS
    - The action completed successfully.
  - LCDS_ERR_ARG_TABLE_RANGE
    - The argument is not within 0 - 3 range for table selection.

This function loads a table in RAM from EEPROM.

Uint8_t SaveCommToEeprom(uint8_t commSel)

Parameters:

- commSel
  - Communication mode selection parameter. This is a decimal value corresponding to the communication mode jumper (JP2) setting on the PmodCLS module. The corresponding jumper setting for each value set can be seen below in Table 4:

MD2, MD1, MD0	Protocol	Details
0,0,0	UART	2400 baud
0,0,1	UART	4800 baud
0,1,0	UART	9600 baud
0,1,1	UART	baud rate in EEPROM
1,0,0	TWI	address: 0x48
1,0,1	TWI	address in EEPROM
1,1,0	SPI	
1,1,1	specified in EEPROM	specified in EEPROM

Table 4. Jumper settings.

See the PmodCLS reference manual for more details about the communication modes.

Return value:

- uint8_t
  - LCDS_ERR_SUCCESS
    - The action completed successfully.
  - LCDS_ERR_ARG_COMM_RANGE
    - The argument is not within 0 - 7 range.

This function saves the communication mode to EEPROM.

void EepromWrEn()

Parameters:

- None.

This function enables the write operation to EEPROM. It has to be called before each EEPROM operation for a proper functionality of the EEPROM operations.

Uint8_t SaveCursorToEeprom(uint8_t modeCrs)

Parameters:

- modeCrs
  - Parameter for selecting the mode of the cursor. It's a numeric parameter corresponding to the three modes available for selection: 0, 1, 2.

Return value:

- uint8_t
  - LCDS_ERR_SUCCESS
    - The action completed successfully.
  - LCDS_ERR_ARG_CRS_RANGE
    - The argument is not within 0 - 2 range for cursor modes.

This function saves the cursor mode into EEPROM memory.

Uint8_t SaveDisplayToEeprom(uint8_t modeDisp)

Parameters:

- modeDisp
  - Parameter for selecting the display mode. It's a numeric parameter corresponding to the two modes available for selection: 0 and 1.

Return value:

- uint8_t
  - LCDS_ERR_SUCCESS
    - The action completed successfully.
  - LCDS_ERR_ARG_DSP_RANGE
    - The argument is not within 0 - 3 range for display settings types.

This function saves the display mode into EEPROM memory.

2.6 User Defined Characters Handling Functions

Uint8_t DefineUserChar(uint8_t* strUserDef, uint8_t charPos)

Parameters:

- strUserDef

- Pointer to the bytes array containing the numerical value of each row in the char.

- charPos
  - The position of the character saved in the memory.

Return value:

- uint8_t
  - LCDS_ERR_SUCCESS
    - The action completed successfully.
  - LCDS_ERR_ARG_POS_RANGE
    - The argument is not within 0 - 7 range for characters position in the memory.

This function saves a user defined char in the RAM. It first sends the escape sequence followed by the string of chars created from the bytes that define the custom character, then the character position in memory followed by the command character. A second escape sequence will place the table containing the defined character in the RAM using the specific command.

void BuildUserDefChar(uint8_t* strUserDef, char* cmdStr)

Parameters:

- strUserDef
  - Pointer to the bytes array containing the values to be converted in values that are recognized by the firmware.
- cmdStr
  - Pointer to the converted string to be sent to the display. The format has to be the ASCII code of each character sent over the SPI interface.

This function builds the string to be converted in an interpretable array of chars for the LCD. It was needed because the LCD controller only accepts ASCII codes for the values sent in order to properly recognize and execute the commands and settings.

Uint8_t DispUserChar(uint8_t* charPos, uint8_t charNumber, uint8_t idxRow, uint8_t idxCol)

Parameters:

- charPos
  - Pointer to an array containing the positions of the characters saved in the RAM.
- charNumber
  - The number of custom chars to be displayed on the LCD, meaning the length of the charPos bytes array.
- idxRow
  - The row at which the first character should be displayed.
- idxCol
  - The column starting from where the character should be displayed.

Return value:

- uint8_t
  - LCDS_ERR_SUCCESS
    - The action completed successfully.

- a combination of the following errors(OR-ed):
  - LCDS_ERR_ARG_COL_RANGE
    - The argument is not within 0 - 39 range.
  - LCDS_ERR_ARG_ROW_RANGE
    - The argument is not within 0 - 2 range for rows.

This function displays a user defined character at the specified position on the LCD. It first sets the position of the cursor at the desired location, specified by the user through the last two parameters and then it sends the numerical values of the characters positions in the table. Note that for the positions, no ASCII code is sent, but the actual value of the character's position in the memory (RAM or EEPROM). The cursor position is automatically incremented.

3 PmodCLS demo project

The demo project 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 5 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 Cls DisplayClear – clear display
2.	Backlight On / Off	BTN2: toggle backlight BTN1: continue	WriteStringAtPos – Write to Cls DisplaySet – Set Backlight and Display
3.	Display Shift left / right	BTN2: shift left BTN1: shift right Double buttons action to continue	WriteStringAtPos – Write to Cls DisplayScroll– shift display DisplayClear – clear display DisplayMode – wrap the line at 16 or 40 characters
4.	Cursor on/Off	BTN2: toggle cursor BTN1: continue	WriteStringAtPos – Write to Cls CursorModeSet – Set Cursor On / Off
5.	Blink On / Off	BTN2: toggle blink BTN1: continue	WriteStringAtPos – Write to Cls CursorModeSet – Set Blink On / Off
6.	Erase char	BTN2: erase chars BTN1: continue	WriteStringAtPos – Write to Cls SetPos – Set Cursor position EraseChars – Erase characters from the current position of the cursor
7.	Erase in line	BTN2: Erase in line BTN1: continue	WriteStringAtPos – Write to Cls EraseInLine – erase characters from line starting from current position
8.	User defined character	Any button to continue	WriteStringAtPos – Write to Cls DispUserChar – display the user defined characters on the Cls

Table 5. Demo used functions and corresponding actions.

In order to accurately implement the functionality described above, the buttons need to be debounced and the following actions to be recognized:

- BTN1 pressed.
- BTN2 pressed.
- One of the buttons is pressed while the other is still pressed.

3.1 Project Files

- *LCDS library files*: *LCDS.h* and *LCDS.cpp* are the two files that define the library and the functions for controlling the module. The header file contains the declaration of the class and the functions instantiated and implemented in the .cpp file. The attribute of the functions is public and there are only few variables declared as private, seen inside the class.
- *DSPI library files: DSPI.h* is a file included in the .pde project in order to be able to use the SPI ports for communication. DSPI library is a standard Arduino library that allows communication via SPI interface and implements dedicated functions for the data transfer. DSPI library files should be provided together with the Arduino environment.

None of the above mentioned files have been modified for the demo application.

- *LCDSDemo.pde* is the application file that implements the demo project using the library functions and files and the standard Arduino functions mentioned above.

3.2 Functions defined in the application

Arduino default functions:

- Setup()
    - This function is called only once at the beginning of the application, and it contains all of the initializations and configurations for the used interfaces, devices, ports, etc. In this case, the SPI interface is configured, the pins of the buttons are set as inputs, and the user defined characters are stored in the memory.
- Loop()
    - This is the main loop in which the functions that need to be executed periodically are called. This function contains the code that implements the steps of the application. After each step, the function defined below is called to verify if a button or both was pressed, waiting for that action in case they weren't.

3.3 Custom Defined Functions

WaitUntilBtnPressed

Prototype: boolean WaitUntilBtnPressed(boolean *pfBtn1Process, boolean *pfBtn2Process)

Description: Waits until a button is pressed and returns the state of the buttons.

This function is called from each of the steps, being used to detect the specific step actions or the move to the next step.

Arguments:

- boolean *pfBtn1Process (acts as output parameter). Pointer to a flag for "button 1 needs processing". Returns True if button 1 was pressed (single action) or one of the buttons was released while the other is pressed (double button action).
- boolean *pfBtn2Process (acts as output parameter). Pointer to a flag for "button 2 needs processing". Returns True if button 2 was pressed (single action) or one of the buttons was released while the other is pressed (double button action).

3.4 Using the libraries

PmodCLS has to be attached to the board by connecting its J1 connector to one of the two SPI ports: JB connector (upper side) for SPI0 or J1 connector for the SPI1 port. As default, the SPI0 is used for the demo application. This can be changed by the user by selecting PAR_ACCESS_SPI1 in the begin() function.

Set the communication mode jumpers to 110, corresponding to MD2 (missing jumper), MD1 (missing jumper), and MD0 (connected jumper). Copy the library files according to the README.txt file. Clicking on the library name automatically includes it in the .pde project and allows its use, after instantiating an object of it.

The rest of the used libraries can be added in the project using the same procedure, but there is no need for changing their location. The functions from the LCDS library are called using an instantiation of an object whose type is LCDS. The instantiation can be done like this:

- `LCDS MyLCDS;`
    - Where MyLCDS is the instantiated object.

Table 6 below presents the functions used by this application that are implemented in the LCDS library.

Function	Location (application function)	Purpose
Begin	setup	Configure CLS communication interface, select the communication port and attach the SPI SS pin to the DSPI library functions.
DisplaySet	setup, loop/step 2, 3,	Set the display and backlight features to the default values
DisplayMode	setup, loop/step 3	Set the wrapping type to 16 or 40 characters on a line
DefineUserChar	setup	Define a custom character and save it in the RAM memory for a later display
DisplayClear	loop/step 1-8	Display clear and return cursor home
WriteStringAtPos	loop/step 1-8	Write a string at a specified position
DisplayScroll	loop/step 3	Display scroll left or right
CursorModeSet	loop/step 4, 5	Cursor on/off, blink on/off
SetPos	loop/step 6, 7	Set position of the cursor
EraseChars	loop/step 6	Erase a specified number of chars starting at the current position of the cursor
EraseInLine	loop/step 7	Erase characters from the cursor position to the end of line
DispUserChar	loop/step 8	Display a previously set number of user defined characters at a specified position
SendBytes	setup, loop/step 1-8	Sends the data bytes to the LCD via SPI interface

Table 6. Functions used in the demo.