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

chipKIT™ OLED2 Library Reference Manual

Revised April 10, 2014

Overview

Digilent provides an MPIDE driver library for an organic LED (OLED) graphics display. This document provides an overview of the driver library and describes the functions that make up its programming interface.

The OLED display uses a Solomon SSD1322 display controller. This controller is capable of working with displays up to 480x128 pixel resolution, although the display on the PmodOLED2 has a 256x64 pixel resolution. The display is 4-bit gray scale. The brightness of each individual pixel is represented by 4-bits. For more information about the hardware interface of the PmodOLED2, refer to the PmodOLED2 reference manual available for download from the Digilent website www.digilentinc.com.

The display uses parallel communication, and specifically, an 8080 Parallel Interface. Communication with the display occurs via the Parallel Master Port (PMP) module present on the PIC microcontroller. The PMP data signals can only be accessed on one Pmod header. You must connect the display to the correct header. Please see the PmodOLED2 reference manual for more details. It is, however, a write-only device. It is not possible to read back status or display data from the display. For this reason, this driver library maintains a frame buffer in memory. All drawing and read-back operations are performed using this memory frame buffer. The display is updated with new image data by copying the frame buffer to the display.

Note: The included demo project was written in MPIDE 0023. It is valid for programming the chipKIT mX3, chipKIT Pro MX4, chipKIT Pro MX7, and the chipKIT Uno32 (with chipKIT PmodShield Uno).

1 Library Operation

1.1 Library Interface

The header file PmodOLED2.h defines the interfaces to the OLED driver.

1.2 Display Initialization

The OLED display has a power on/power off sequence that should be followed. Before making calls to any other library functions, the Oled2Init() function must be called. This function initializes the PIC32 resources used by the library, and then turns on power to the display and initializes it.

1.3 Character Mode Operation

The display hardware is a graphical display. The library, however, supports a character oriented display where the display is treated as if it were a character display. The character mode functions use character row and character column numbers for cursor position.

1.4 Graphic Mode Operation

The graphic functions support reading and writing pixels, drawing lines and rectangles, and other graphical operations. The graphic functions also support drawing characters at any position on the display. The character 'mode' and graphics 'mode' are not really modes of operation of the library; they are simply sets of functions. Calls to character functions and graphic functions can be interleaved without restriction.

1.5 Drawing Mode

The library supports four drawing modes for the graphic drawing operations. These modes specify the operation to be performed between the current drawing color and the current state of the pixel to determine the final pixel value. The following modes in Table 1 are supported:

Value	Description
modeSet	set the pixel to the current drawing color
modeOr	OR the current pixel value with the current drawing color
modeAnd	AND the current pixel value with the current drawing color
modeXor	XOR (exclusive or) the current pixel color with the current drawing color.

Table 1. Supported modes.

1.6 Display Organization

Each address in the display memory corresponds to a column on the display with a width of 4 pixels. The first byte in memory (byte 0) corresponds to the two adjacent pixels on the right of the column. The second byte in memory (byte 1) corresponds to the two adjacent pixels on the left of the column, etc. The least significant nibble of each byte contains gray scale level data for the pixel on the right, and the most significant nibble containing gray scale level data for the pixel on the left. This continues across the display to byte 127, which is the rightmost column of two pixels. Byte 128 corresponds to the next column of two pixels at the left of the display.

2 OLED Library Functions

2.1 Display Management Functions

void begin(void)

Parameters:
- None

Initializes the driver, turns on power to the display, and initializes it. This function must be called before any other functions in the library are called.

void end(void)

Parameters:
- None

Turns off power to the display and then releases the PIC32 SPI controller and the pins used for the OLED interface.

void displayOn(void)

Parameters:
- None

Activates the display.

void displayOffvoid)

Parameters:
- None

Blanks the display.

void clear(void)

Parameters:
- None

Clears the memory frame buffer and then updates the display, thus clearing it.

void clearBuffer(void)

Parameters:
- None

Clear the display memory buffer without updating the display.

void updateDisplay(void)

Parameters:
- None

Updates the OLED display with the contents of the memory buffer.

2.2 Character Functions

The following functions treat the display as if it were a character-mode display. They use the character column number and row number for the cursor position. The row and column numbers are 0-based. Therefore, the column numbers range from 0 – (N-1) and the row numbers range from 0 – (M-1), where N and M are the number of columns and the number of rows.

void setCursor(int xch, int ych)

Parameters:
- xch
    - Horizontal character position (column)
- ych
    - Vertical character position (row)

Set the character cursor position to the specified location. If either the specified X or Y location is off the display, it is clamped to be on the display.

void getCursor(int *pxcy, int *pych)

Parameters:
- pxch
    - Pointer to variable to receive horizontal position
- pych
    - Pointer to variable to receive vertical position

Returns the current cursor position.

int defineUserChar(char ch, BYTE *pbDef)

Parameters:
- ch
    - Character code to define
- pdDef
    - Definition for character

Return Value:
- True if successful, false if not.

Give a definition for the glyph for the specified user character code. User definable character codes are in the range 0x00 – 0x1F. If the code specified by ch is outside of this range, then the function returns false. Current functionality of the library defines a character as 8 bytes. Each byte is a vertical column of 8 pixels, where a ‘0’ signifies that pixel is off, and a ‘1’ signifies that pixel is on. The definitions of the character are rotated 90˚ clockwise. For example, if a user was to define their own character, they should do it as follows:

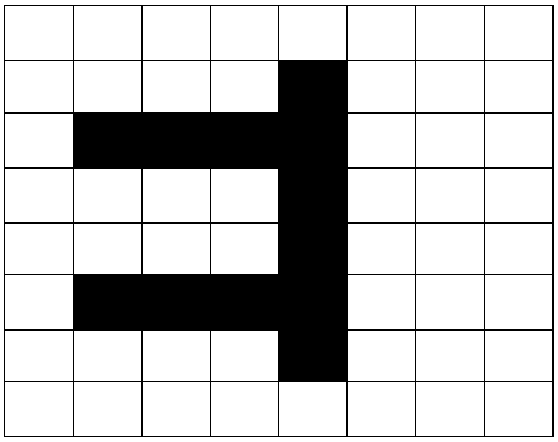
Figure 1. Bitmap of pi.

Figure 1 is an 8x8 bit map of the character π (pi). The vertical bytes from left to right are: 0x00, 0x24, 0x24, 0x24, 0x7E, 0x00, 0x00, 0x00. Therefore, the following definition would define this character.

- byte MyChar[] = {0x00, 0x24, 0x24, 0x24, 0x7E, 0x00, 0x00, 0x00};

An example User definition is included in the demo project for this library.

void setCharUpdate(int f)

Parameters:

- f
  - Enable/disable automatic update

Sets the character update mode. This determines whether or not the display is automatically updated after a character or string is drawn. A zero value turns off automatic updating and a non-zero value turns automatic updating on.

int getCharUpdate(void)

Parameters:

- None

Returns the current character update mode.

void putChar(char ch, BYTE clr)

Parameters:

- ch
  - Character to write to display
- clr
  - Gray scale level for character (value must be 0-15)

Writes the specified character to the display at the current cursor position and advances the cursor.

void putString(char *sz, BYTE clr)

Parameters:

- sz
  - Pointer to the null terminated string
- clr
  - Gray scale level for character (value must be 0-15)

Writes the specified null terminated character string to the display and advances the cursor.

2.3 Graphic Functions

void setDrawColor(BYTE clr)

Parameters:

- clr
  - Drawing color to set

Sets the foreground color used for pixel draw operations.

void setDrawMode(int mod)

Parameters:

- mod
  - Drawing mode select

Sets the specified mode as the current drawing mode. The following values listed in Table 2 are the drawing mode options.

Value	Description
modeSet	Set the pixel to the current drawing color
modeOr	OR the pixel value with the current drawing color
modeAnd	AND the pixel value with the current drawing color
modeXor	XOR the pixel value with the current drawing color

Table 2. Drawing mode options.

int getDrawMode(void)

Parameters:

- None

Returns the current drawing mode.

BYTE* getStdPattern(int ipat)

Parameters:

- ipat
  - Index to standard fill pattern (0-7)

Return Value:

- Returns a pointer to the byte array for the specified standard fill pattern.

A fill pattern is an array of 8 bytes. The following patterns are available (each pattern fills an 8x8 pixel square):

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 0x00

0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, // 0x01

0xAA, 0x55, 0xAA, 0x55, 0xAA, 0x55, 0xAA, 0x55, // 0x02

0x11, 0x44, 0x00, 0x11, 0x44, 0x00, 0x11, 0x44, // 0x03

0x92, 0x45, 0x24, 0x92, 0x45, 0x24, 0x92, 0x45, // 0x04

0x49, 0x92, 0x24, 0x49, 0x92, 0x24, 0x49, 0x92, // 0x05

0x22, 0x11, 0x22, 0x00, 0x22, 0x11, 0x22, 0x00, // 0x06

0x11, 0x22, 0x11, 0x00, 0x11, 0x22, 0x11, 0x00  // 0x07
```

The standard fill pattern, 0, is all pixels off (black). Fill pattern 1 is all pixels on (at the current gray scale level last set by Oled2SetDrawColor(clr)). The default draw color is gray scale level 15 (brightest). Notice that fill patterns are defined the same way as characters. There are 8 bytes in a definition, and each byte is a vertical column of 8 pixels. For more details on defining user patterns, see the information above about defining user characters.

void setFillPattern(BYTE *pbPat)

Parameters:

- pbPat
  - Pointer to the fill pattern

Set a pointer to the current fill pattern to use. A fill pattern is an array of 8 bytes. This pattern will be used by the Oled2DrawFillRect() function. Usually, you will call this function as follows:

- Oled2SetFillPattern(Oled2GetStdPattern(ipat));

void moveTo(int xco, int yco)

Parameters:

- xco
  - X coordinate
- yco
  - Y coordinate

Sets the current graphics drawing position.

void getPos(int *pxco, int *pyco)

Parameters:

- pxco
  - Variable to receive x coordinate

- pyco
  - Variable to receive y coordinate

Returns the current graphics drawing positions.

void drawPixel(void)

Parameters:

- None

Sets the pixel at the current drawing location to the specified value.

BYTE getPixel(void)

Parameters:

- None

Returns the value of the pixel at the current drawing location.

void drawLine(int xco, int yco)

Parameters:

- xco
  - X coordinate of the other corner
- yco
  - Y coordinate of the other corner

Draws a line from the current position to the specified position. The specified position becomes the new current position.

void drawRect(int xco, int yco)

Parameters:

- xco
  - X coordinates of other corner
- yco
  - Y coordinates of other corner

Draws a rectangle bounded by the current location and the specified location. The current location is not modified.

void drawFillRect(int xco, int yco)

Parameters:

- xco
  - X coordinates of other corner
- yco
  - Y coordinates of other corner

Fills a rectangle bounded by the current location and the specified location. This does not draw an outline around the rectangle. The current position is not modified.

void getBmp(int dxco, int dyco, BYTE *pbBmp)

Parameters:
- dxco
  - Width of bitmap
- dyco
  - Height of bitmap
- pbBits
  - Pointer to the bitmap bits

This routine returns an array of bytes containing the data corresponding to the rectangle implied by the current location and the specified width and height. The buffer specified by pbBits must be large enough to hold the resulting bytes. The required buffer size in bytes is: (dxco/2) * dyco.

void putBmp(int dxco, int dyco, BYTE *pbBmp)

Parameters:
- dxco
  - Width of bitmap
- dyco
  - Height of bitmap
- pbBits
  - Pointer to the bitmap bits

This routine will draw the specified bitmap into the display buffer at the current location. The source bitmap (BYTE * pbBmp), must be defined a specific way. Unlike the bytes in character and fill pattern definitions, each byte in a drawing bitmap will contain a gray scale level (0x0 – 0xF) for two adjacent pixels on the display. See the above section on Display Organization for more information. Any bitmap that is fetched from Oled2GetBmp() will be in this format. If the user wishes to send a bitmap that they defined, they should do it in the following manner:

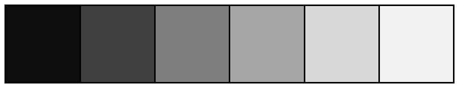

Figure 2. 6x1 bitmap.

If the user defined a 6 x 1 bitmap, such as above, then the definition should be similar to the one below:

- byte MyBmp[] = {0xFC, 0XA8, 0x52 };

- OLED2.putBmp(5, 0, MyBmp);

There is an example in the demo project for this library.

void Oled2PutBmpFlipped(int dxco, int dyco, BYTE *pbBmp)

Parameters:
- dxco
  - Width of bitmap
- dyco
  - Height of bitmap
- pbBits
  - Pointer to the bitmap bits

This routine will draw a bitmap (information stored in BYTE * pbBmp) flipped about the vertical axis (y-axis).

void drawChar(char ch, BYTE clr)

Parameters:

- ch
  - Character to write to display
- clr
  - Gray scale level for character (value must be 0-15)

Draws the specified character to the display at the current cursor position and advances the cursor. Note that this function will advance the current pixel position on the display by 8 pixels.

void drawstring(char *sz, BYTE clr)

Parameters:

- sz
  - Pointer to the null terminated string
- clr
  - Gray scale level for character (value must be 0-15)

Writes the specified null terminated character string to the display and advances the cursor. Note that this function will advance the current pixel position on the display by 8 pixels.