LCD Library

The XMOS LCD library allows you to interface to LCD screens via a parallel bus.

Features

- Standard component to support different types LCD displays with RGB 565
- Resolution of up to 800 * 480 pixels
- Up to 62.5 MHz pixel clock
- Hsync/Vsync and/or Data_Enable signal interfaces supported
- Configurable porch timings

Components

- LCD server
- LCD server with synchronization

Resource Usage

This following table shows typical resource usage in some different configurations. Exact resource usage will depend on the particular use of the library by the application.

Configuration	Pins	Ports	Clocks	Ram	Logical cores
LCD server, 16 bit data, sync mode: DE	18	2 (1-bit), 1 (16-bit)	1	~1.0K	1
LCD server, 16 bit data, sync mode: h_sync, v_sync	19	3 (1-bit), 1 (16-bit)	1	~1.1K	1
LCD server, 16 bit data, sync mode: h_sync, v_sync, DE	20	4 (1-bit), 1 (16-bit)	1	~0.8K	1
LCD server with synchronization, 16 bit data, sync mode: DE	18	2 (1-bit), 1 (16-bit)	1	~1.0K	1
LCD server with synchronization, 16 bit data, sync mode: h_sync, v_sync	19	3 (1-bit), 1 (16-bit)	1	~1.1K	1
LCD server with synchronization, 16 bit data, sync mode: h_sync, v_sync, DE	20	4 (1-bit), 1 (16-bit)	1	~0.8K	1

Software version and dependencies

This document pertains to version 3.0.0 of this library. It is known to work on version 14.0.1 of the xTIMEcomposer tools suite, it may work on other versions.

The library does not have any dependencies (i.e. it does not rely on any other libraries).

Related application notes

The following application notes use this library:

- AN00168 - Using the LCD library

Copyright 2015 XMOS Ltd.

1 Hardware characteristics

The signals from the xCORE required to drive the LCD are:

Pixel Clock	Clock line, all other signals with be clocked off of this.
Data	The pixel data supplied to the LCD over a parallel bus.
Data Enabled (DE)	Strobe to indicate that the data on the parallel bus is valid
Horizontal Sync (h_sync)	A signal which sends a pulse to indicate the start of the horizontal scan.
Vertical Sync (v_sync)	A signal which sends a pulse to indicate the start of the vertical scan.

Table 1: LCD data and signal wires

1.1 Connecting to the xCORE LCD server

The LCD wires must be connected to the xCORE device as shown in Figure 1. The control signals can be connected to any of the one bit ports on the device provided they do not overlap with any other used ports and are all on the same tile. The parallel data bus must not overlap with any of the control signals.

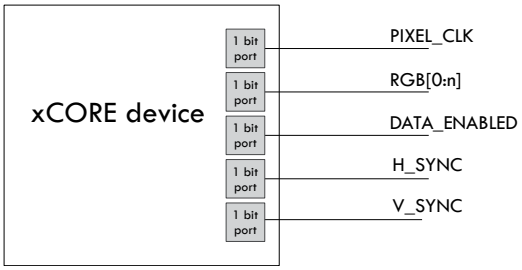

Figure 1: LCD connection to the xCORE device

The data bus may be wired to the LCD in any electrically sound configuration. For example, a 24 bit colour LCD with 8 bits for each of red, green and blue, could be driven by a 16 bit bus. For the standard RGB565,format this can be achieved by the wiring configuration of:

- `data[ 4: 0]` from the xCORE drives `red[7:3]` on the LCD
- `data[10: 5]` from the xCORE drives `green[7:2]` on the LCD
- `data[15:11]` from the xCORE drives `blue[7:3]` on the LCD
- `red[2:0]` should be grounded on the LCD
- `green[1:0]` should be grounded on the LCD
- `blue[2:0]` should be grounded on the LCD

where `data` is a 16 bit port. This has the advantage that port buffering can be used, improving the maximum pixel clock frequency. Likewise, rgb888 would be achieved by:

- `data[ 7: 0]` from the xCORE drives `red[7:0]` on the LCD
- `data[15: 8]` from the xCORE drives `green[7:0]` on the LCD
- `data[23:16]` from the xCORE drives `blue[7:0]` on the LCD

where `data` is a 32 bit port in this case.

The DE, h_sync and v_sync signals are all optional, however, every LCD will require some of them. See the datasheet of the LCD to find out which signals are required.

Copyright 2015 XMOS Ltd.

1.2 Timing configuration

The LCD's datasheet will give all the timing characteristics necessary to setup the LCD. Refer to Figure 2 and Figure 3 for clarification of the LCD timing definitions.

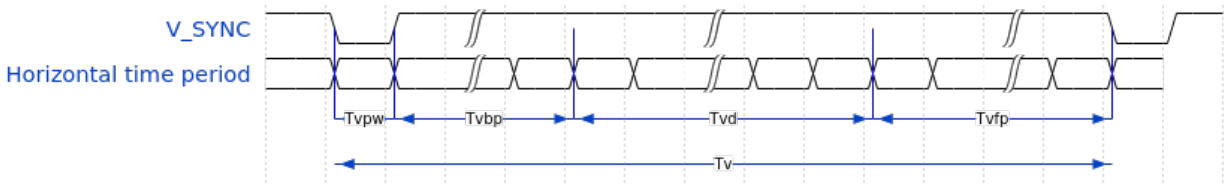

Figure 2: LCD vertical timing

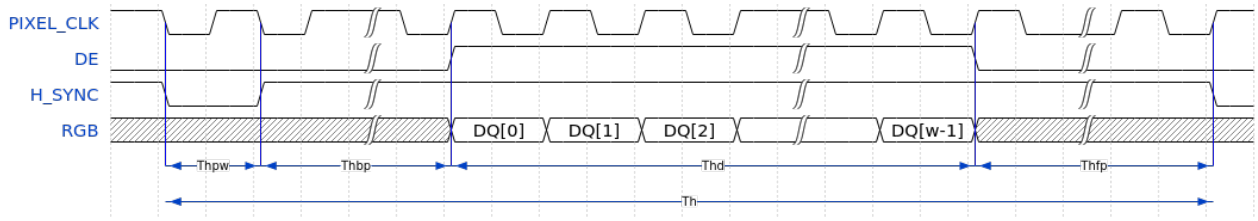

Figure 3: LCD horizontal timing

Tvpw	Vertical pulse width
Tvbp	Vertical back porch
Tvd	Vertical data period
Tvfp	Vertical front porch
Tv	Vertical total time
Thpw	Horizontal pulse width
Thbp	Horizontal back porch
Thd	Horizontal data valid
Thfp	Horizontal front porch
Th	Horizontal total time

Table 2: LCD Timing

1.3 Output Mode

The correct output mode must be selected for the application and physical setup. The data may be up to 32 bits per pixel and the data bus can be from one to 32 bits wide.

Copyright 2015 XMOS Ltd.

2 LCD operational overview

When in operation the LCD presents a constant real-time requirement on the xCORE. An LCD works by sequentially refreshing its pixels, typically working from a corner, refreshing each horizontal line and progressing vertically until the whole screen has been refreshed. This constant refreshing produces the real-time requirement that the xCORE must satisfy. To make matters easier there are porch intervals between the refresh of each line. At the end of a line refresh, the LCD will pause for a moment before refreshing the next line. This gives the xCORE opportunity to prepare the line buffers to be output.

3 LCD API

All LCD functions can be accessed via the `lcd.h` header:

```
#include <lcd.h>
```

You will also have to add `lib_lcd` to the `USED_MODULES` field of your application Makefile.

LCD server and client are instantiated as parallel tasks that run in a `par` statement. The client (application on most cases) can connect via a streaming channel.

For example, the following code instantiates an LCD server and connects and application to it:

```
out buffered port:32   lcd_rgb                = XS1_PORT_16B;
out port               lcd_clk                = XS1_PORT_1I;
out port               lcd_data_enabled       = XS1_PORT_1L;
out buffered port:32   lcd_h_sync             = XS1_PORT_1J;
out port               lcd_v_sync             = XS1_PORT_1K;
clock                  lcd_cb                 = XS1_CLKBLK_1;

int main(void) {
  streaming chan c_lcd;
  par {
    lcd_server(
        c_lcd,
        lcd_rgb,
        lcd_clk,
        lcd_data_enabled,
        lcd_h_sync,
        lcd_v_sync,
        lcd_cb,
        480,
        272,
        5, 40, 1,
        8, 8, 1,
        data16_port16,
        3);
    my_application(c_lcd);
  }
  return 0;
}
```

Note: The client and LCD server must be on the same tile as the line buffers are transfered by moving pointers from one task to another.

`lcd_init` is used to start the LCD running, before which the pixel clock is not running and no other signals are outputting. This gives the application time to prepare any necessary line buffers before beginning. As soon as the `lcd_init` has been executed there is a constant real-time requirement on the client to update the LCD server with more line buffers. The LCD server requests new line buffers by inserting a token into the channel to the client. The client must call `lcd_req` either by selecting on it or by explicitly called normally to acknowledge this request.

Between the client and the LCD server is a command buffer capable of holding of up to two line buffers from the client to the server. This means that the client can call `lcd_update` more than once per `lcd_req` to increase the time between subsequenct updates if need be. However, for every `lcd_req` there must be one `lcd_update` on average.

Copyright 2015 XMOS Ltd.

3.1 Optimizing the decoupling between client and server

To maximize performance the client must make best use of the command buffer between the `lcd_server` and the client application. To do this the client must consider that at any given time there can be up to 4 line buffers in the LCD pipeline; these buffers are:

- one being worked on by the client application
- two in the command buffer between the client and `lcd_server`
- one being outputted to the LCD by the `lcd_server`

This means that if the client application has to service another request it can have up to the time of outputting 3 lines to the LCD before sending the next `lcd_update` to the `lcd_server`. This is provided that the command buffers are full and the `lcd_server` has just started outputting a line.

3.2 Synchronization of LCD with an external source

''lcd_server_sync" can synchronize with an external source by stretching or shrinking its vertical back porch. The absolute value of the adjustment must be within half a horizontal time. To synchronize to an external clock the use of a PID control loop is recommended.

To make an adjustment, the client application must call `lcd_update_sync` with the required update amount. Then on the next frame the update will take effect for only that frame. Subsequent frames will revert to the default timings.

A typical use case for this functionality is synchronizing the LCD to an external video stream. For example: streaming video from a image sensor might be produced at the rate of 15.000 frames per second. The LCD server should then be setup to run the LCD at 15.000 frames per second also by adjusting first the clock divider, then the porch timings.

The frame rate can be calculated by solving the following formula:

Frames per second = Pixel clock rate / ((Thpw + Thfp + Thbp + Thd) * (Tvpw + Tvfp + Tvbp + Tvd))

The easiest way to do this is to substitute the knowns: Tvd(screen height), Thd(screen width) and frames per second (external source). Next choose a clock divider that will allow the porch timings to be within specification as given by the LCD's datasheet. Finally pick the porch timings to match the frame rate as exactly as possible to the external source.

3.3 API

Function	**lcd_server**
Description	The LCD server.

Continued on next page

Type	void lcd_server(streaming chanend c_client, out buffered port:32 lcd_rgb, out port lcd_clk, out port ?lcd_data_enabled, out buffered port:32 ?lcd_h_sync, out port ?lcd_v_sync, clock lcd_cb, const static unsigned width, const static unsigned height, const static unsigned h_front_porch, const static unsigned h_back_porch, const static unsigned h_pulse_width, const static unsigned v_front_porch, const static unsigned v_back_porch, const static unsigned v_pulse_width, const static e_output_mode output_mode, const static unsigned clock_divider)

Continued on next page

Parameters	`c_client` The data channel connecting to the client.
	
	`lcd_rgb` The parallel data port.
	
	`lcd_clk` The pixel clock.
	
	`lcd_data_enabled`
	The data enabled signal.
	
	`lcd_h_sync`
	The horizontal sync signal.
	
	`lcd_v_sync`
	The vertical sync signal.
	
	`lcd_cb` A clock block to manage the ports.
	
	`width` Width of the LCD screen in pixels.
	
	`height` Height of the LCD screen in pixels.
	
	`h_front_porch`
	Time of horizontal front porch in pixel clocks.
	
	`h_back_porch`
	Time of horizontal back porch in pixel clocks.
	
	`h_pulse_width`
	Time of horizontal pulse width in pixel clocks.
	
	`v_front_porch`
	Time of vertical front porch in horizontal times.
	
	`v_back_porch`
	Time of vertical back porch in horizontal times.
	
	`v_pulse_width`
	Time of vertical pulse width in horizontal times.
	
	`output_mode`
	The mode of writing line buffers to the data port.
	
	`clock_divider`
	The divider of the system clock to give the pixel clock.

Function	**lcd_server_sync**
Description	The LCD server with synchronization.

Continued on next page

Type	void lcd_server_sync(streaming chanend c_client, streaming chanend c_sync, out buffered port:32 lcd_rgb, out port lcd_clk, out port ?lcd_data_enabled, out buffered port:32 ?lcd_h_sync, out port ?lcd_v_sync, clock lcd_cb, const static unsigned width, const static unsigned height, const static unsigned h_front_porch, const static unsigned h_back_porch, const static unsigned h_pulse_width, const static unsigned v_front_porch, const static unsigned v_back_porch, const static unsigned v_pulse_width, const static e_output_mode output_mode, const static unsigned clock_divider)

Continued on next page

Parameters	`c_client` The data channel connecting to the client.
	
	`c_sync` The synchronisation channel connecting to the client.
	
	`lcd_rgb` The parallel data port.
	
	`lcd_clk` The pixel clock.
	
	`lcd_data_enabled`
	The data enabled signal.
	
	`lcd_h_sync`
	The horizontal sync signal.
	
	`lcd_v_sync`
	The vertical sync signal.
	
	`lcd_cb` A clock block to manage the ports.
	
	`width` Width of the LCD screen in pixels.
	
	`height` Height of the LCD screen in pixels.
	
	`h_front_porch`
	Time of horizontal front porch in pixel clocks.
	
	`h_back_porch`
	Time of horizontal back porch in pixel clocks.
	
	`h_pulse_width`
	Time of horizontal pulse width in pixel clocks.
	
	`v_front_porch`
	Time of vertical front porch in horizontal times.
	
	`v_back_porch`
	Time of vertical back porch in horizontal times.
	
	`v_pulse_width`
	Time of vertical pulse width in horizontal times.
	
	`output_mode`
	The mode of writing line buffers to the data port.
	
	`clock_divider`
	The divider of the system clock to give the pixel clock.

Function	lcd_init
Description	Initialises the LCD with the first line to be rendered. After this completes there is a permanent real time requirement to update the LCD server with more data to render.
Type	`void lcd_init(streaming chanend c_lcd, unsigned *unsafe buffer)`
Parameters	`c_lcd` The channel to the LCD server `buffer` This is a pointer to the data to be written to the LCD

Function	lcd_update
Description	Passes a buffer to be rendered by the LCD.
Type	`void lcd_update(streaming chanend c_lcd, unsigned *unsafe buffer)`
Parameters	`c_lcd` The channel to the LCD server `buffer` This is a pointer to the data to be written to the LCD

Function	lcd_req
Description	Returns the movable pointer from the LCD server to the client for reuse This is a blocking call that may be used as a select handler.
Type	`void lcd_req(streaming chanend c_lcd)`
Parameters	`c_lcd` The channel to the LCD server

Function	lcd_sync_update
Description	Adds or removes n pixel clock periods from the vertical back porch.
Type	`void lcd_sync_update(streaming chanend c_sync, int n)`
Parameters	`c_sync` The synchronisation channel to the LCD server `n` The number of pixel clocks to add to to veritcal back porch

APPENDIX A - Known Issues

There are no known issues with this library.

APPENDIX B - LCD library change log

B.1 3.0.0

- Consolidated version, major rework from previous LCD components

Copyright © 2015, All Rights Reserved.

Xmos Ltd. is the owner or licensee of this design, code, or Information (collectively, the "Information") and is providing it to you "AS IS" with no warranty of any kind, express or implied and shall have no liability in relation to its use. Xmos Ltd. makes no representation that the Information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.