Display Controller Component

REV A

Publication Date: 2013/9/30
XMOS © 2013, All Rights Reserved.

XMOS

Table of Contents

1 Overview

IN THIS CHAPTER

The display controller module is used to drive a single graphics LCD screen up to 800 * 600 pixels incorporating a managed double buffer.

1.1 Features

- Non-blocking SDRAM management.
- Real time servicing of the LCD.
- Touch interactive display
- Image memory manager to simplify handling of images.
- No real time constraints on the application.

1.2 Memory requirements

Resource	Usage
Stack	6198 bytes
Program	11306 bytes

1.3 Resource requirements

Resource	Usage
Channels	3
Timers	0
Clocks	0
Threads	1

1.4 Performance

The achievable effective bandwidth varies according to the avaliable XCore MIPS. The maximum pixel clock supported is 25MHz.

2 Hardware Requirements

IN THIS CHAPTER

2.1 Recommended Hardware

2.1.1 Slicekit

This module may be evaluated using the Slicekit Modular Development Platform, available from digikey. Required board SKUs are:

- XP-SKC-L2 (Slicekit L2 Core Board)
- XA-SK-SCR480 plus XA-SK-XTAG2 (Slicekit XTAG adaptor)

2.2 Demonstration Applications

2.2.1 Display Controller Application

- Package: sw_display_controller
- Application: app_display_controller

This combination demo employs the `module_lcd` along with the `module_sdram`, `module_touch_controller_lib`, `module_i2c_master` and the `module_display_controller` framebuffer framework component to implement a 480x272 display controller.

Required board SKUs for this demo are:

- XP-SKC-L2 (Slicekit L2 Core Board) plus XA-SK-XTAG2 (Slicekit XTAG adaptor)
- XA-SK-SDRAM
- XA-SK-SCR480 (which includes a 480x272 color touch screen)

3 API

IN THIS CHAPTER

- component: sc_sdram_burst which handles the SDRAM
- component: sc_lcd which handles the LCD

The below section details the APIs in the application. For details about the LCD and SDRAM APIs please refer to the respective repositories.

3.1 Configuration Defines

The `module_display_controller` can be configured via the header `display_controller_conf.h`. The module requires nothing to be additionally defined however any of the defines can be overridden by adding the header `display_controller_conf.h` to the application project and adding the define that needs overridding. The possible defines are:

DISPLAY_CONTROLLER_MAX_IMAGES
This defines the storage space allocated to the display controller for it to store image metadata. When an image is registered with the display controller its dimensions and location in SDRAM address space are stored in a table. The define specifies how many entries are allowed in that table. Note, there is no overflow checking by default.

DISPLAY_CONTROLLER_VERBOSE
This define switches on the error checking for memory overflows and causes verbose error warnings to be emitted in the event of an error.

3.2 API

- `display_controller_client.xc`
- `display_controller_internal.h`
- `display_controller.xc`
- `display_controller.h`
- `transitions.h`
- `transitions.xc`

The display controller handles the double buffering of the image data to the LCD as a real time service and manages the I/O to the SDRAM as a non-real time service.

The display controller API is as follows: .. doxygenfunction:: display_controller .. doxygenfunction:: display_controller_image_read_line .. doxygenfunction:: display_controller_image_read_line_p .. doxygenfunction:: display_controller_image_write_line .. doxygenfunction:: display_controller_image_write_line_p .. doxygenfunction:: display_controller_image_read_partial_line .. doxygenfunction:: display_controller_image_read_partial_line_p .. doxygenfunction:: display_controller_register_image .. doxygenfunction:: display_controller_wait_until_idle .. doxygenfunction:: display_controller_wait_until_idle_p .. doxygenfunction:: display_controller_frame_buffer_commit .. doxygenfunction:: display_controller_frame_buffer_init

The transition API is as follows: .. doxygenfunction:: transition_wipe .. doxygenfunction:: transition_slide .. doxygenfunction:: transition_roll .. doxygenfunction:: transition_dither .. doxygenfunction:: transition_alpha_blend

The transitions use the display controller API.

4 Programming Guide

IN THIS CHAPTER

4.1 Shared Memory Interface

The display controller uses a shared memory interface to move the large amount of data around from tile to tile efficiently. This means that the `display_controller`, `sdram_server` and `lcd_server` must be one the same tile.

4.2 Source code structure

Project	File	Description
module_display_controller	`display_controller.h`	Header file containing the APIs for the display controller component.
	`display_controller.xc`	File containing the implementation of the display controller component.
	`display_controller_client.xc`	File containing the implementation of the display controller client functions.
	`display_controller_internal.h`	Header file containing the user configurable defines for the display controller component.
	`transitions.h`	Header file containing the APIs for the display controller transitions.
	`transitions.xc`	File containing the implementation of the display controller transitions.

Figure 1: Project structure

4.3 Executing The Project

The module by itself cannot be built or executed separately - it must be linked in to an application. Once the module is linked to the application, the application can be built and tested for driving a LCD screen.

1. `module_display_controller`
2. `module_lcd`
3. `module_sdram`
1. `module_touch_controller_lib` or `module_touch_controller_server`
2. `module_i2c_master`

should be added to the list of MODULES.

4.4 Software Requirements

The module is built on XDE Tool version 12.0 The module can be used in version 12.0 or any higher version of xTIMEcomposer.

5 Example Applications

IN THIS CHAPTER

This tutorial describes a demo application that uses the display controller module. §2.1 describes the required hardware setup to run the demos.

5.1 app_display_controller_demo

This application demonstrates how the `lcd_module` is used to write image data to the LCD screen whilst imposing no real time constraints on the application. The purpose of this demonstration is to show how data is passed to the `display_controller`. This application also demonstrates an interactive display using `touch_controller_lib` module.

5.2 Application Notes

5.2.1 Getting Started

1. Plug the XA-SK-LCD Slice Card into the 'TRIANGLE' slot of the Slicekit Core Board
2. Plug the XA-SK-SDRAM Slice Card into the 'STAR' slot of the Slicekit Core Board
3. Open `app_display_controller_demo.xc` and build the project.
4. run the program ensuring that it is run from the project directory where the tga images are.

The output produced should look like a series of images transitioning on the LCD when the screen is touched.

Copyright © 2013, 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.

XMOS and the XMOS logo are registered trademarks of Xmos Ltd. in the United Kingdom and other countries, and may not be used without written permission. All other trademarks are property of their respective owners. Where those designations appear in this book, and XMOS was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.