

LCD Component

REV A

Publication Date: 2013/2/19
XMOS © 2013, All Rights Reserved.



Table of Contents

1	Overview	3
1.1	LCD component	3
1.1.1	Features	3
1.1.2	Memory requirements	3
1.1.3	Resource requirements	4
1.1.4	Performance	4
1.2	Touch screen component	4
1.2.1	Features	4
1.2.2	Memory requirements	4
1.2.3	Resource requirements	5
2	Hardware Requirements	6
2.1	Recommended Hardware	6
2.1.1	SliceKit	6
2.2	Demonstration Applications	6
2.2.1	LCD Demo Application	6
2.2.2	Text Display Application	6
2.2.3	Display Controller Application	6
2.2.4	Touch Screen Demo Application	7
3	API	8
3.1	module_lcd	8
3.1.1	Configuration Defines	8
3.1.2	Implementation Specific Defines	8
3.1.3	API	9
3.2	module_touch_controller_lib	10
3.2.1	Configuration Defines	10
3.2.2	User Define	10
3.2.3	API	11
3.3	module_touch_controller_server	12
3.3.1	Configuration Defines	13
3.3.2	User Define	13
3.3.3	API	13
4	Programming Guide	16
4.1	Source code structure	16
4.2	Additional Files	16
4.3	How to select the LCD target	16
4.4	Executing The Project	17
4.5	Software Requirements	17
5	Touch Controller Programming Guide	18
5.1	Source code structure	18
5.2	How to develop an application	18
5.3	Executing The Project	20
5.4	Software Requirements	20
6	Example Applications	21
6.1	app_lcd_demo	21

- 6.2 Application Notes 21
 - 6.2.1 Getting Started 21
- 6.3 app_touch_controller_lib_demo 22
- 6.4 Application Notes 22
 - 6.4.1 Getting Started 22
- 6.5 app_touch_controller_server_demo 22
- 6.6 Application Notes 22
 - 6.6.1 Getting Started 23

1 Overview

IN THIS CHAPTER

- ▶ LCD component
 - ▶ Touch screen component
-

1.1 LCD component

The LCD component is used to drive a single graphics LCD module up to 800 * 600 pixels with pixel clocks of up to 25MHz.

1.1.1 Features

- ▶ Standard component to support different LCD displays with RGB 565.
- ▶ Different color depths 32 bpp, 16 bpp, etc. based on user configuration.
- ▶ Resolution of up to 800 * 600 pixels. See table below for different screen configurations.
- ▶ Outputs to a CMOS interface.
- ▶ Configurability of * LCD pixel dimensions, * clock rate, * horizontal and vertical timing requirements, * port mapping of the LCD.
- ▶ Requires a single core for the server. * The function `lcd_server` requires just one core, the client functions, located in `lcd.h` are very low overhead and are called from the application.

1.1.2 Memory requirements

Resource	Usage
Stack	92 bytes
Program	2168 bytes

1.1.3 Resource requirements

Resource	Usage
Channels	1
Timers	0
Clocks	1
Logical Cores	1

1.1.4 Performance

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

1.2 Touch screen component

The touch screen component is used to read the touch coordinates from the touch screen controller AD7879-1.

1.2.1 Features

- ▶ Standard components to support touch screen controller with I2C serial interface
- ▶ Supports 4-wire resistive touch screens of different sizes
- ▶ Resolution of 4096 * 4096 points
- ▶ Pen-down interrupt signal supported
- ▶ Outputs touch coordinates with time information
- ▶ `module_touch_controller_lib` requires a single core while `module_touch_controller_server` requires an additional core for the server.

1.2.2 Memory requirements

`app_touch_controller_lib_demo`

Resource	Usage
Stack	304 bytes
Program	3160 bytes

`app_touch_controller_server_demo`

Resource	Usage
Stack	420 bytes
Program	3576 bytes

1.2.3 Resource requirements

app_touch_controller_lib_demo

Resource	Usage
Channels	0
Timers	3
Clocks	1
Logical Cores	1

app_touch_controller_server_demo

Resource	Usage
Channels	1
Timers	3
Clocks	1
Logical Cores	2

2 Hardware Requirements

IN THIS CHAPTER

- ▶ Recommended Hardware
 - ▶ Demonstration Applications
-

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) plus XA-SK-SCR480 plus XA-SK-XTAG2 (Slicekit XTAG adaptor)

2.2 Demonstration Applications

2.2.1 LCD Demo Application

The LCD demo application shows how a buffer of image data can be written to the 480x272 LCD screen that is supplied with the XA-SK-SCR480 Slice Card.

- ▶ Package: `sc_lcd`
- ▶ Application: `app_lcd_demo`

2.2.2 Text Display Application

This application demonstrates how the `module_text_display` can be used to put text into the LCD image buffer for display to the Slice Card screen.

- ▶ Package: `sc_lcd`
- ▶ Application: `app_text_display`

2.2.3 Display Controller Application

This combination demo employs the `module_lcd` along with the `module_sdram` 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-SCR480 for the LCD
- ▶ XA-SK-SDRAM for the SDRAM
- ▶ Package: sw_display_controller
- ▶ Application: app_display_controller

2.2.4 Touch Screen Demo Application

The touch screen demo application shows how a touch event is processed and the touch coordinates are fetched from the touch screen controller chip fitted on the XA-SK-SCR480 Slice Card.

- ▶ Package: sc_lcd
- ▶ Applications: app_touch_controller_lib_demo, app_touch_controller_server_demo

3 API

IN THIS CHAPTER

- ▶ `module_lcd`
 - ▶ `module_touch_controller_lib`
 - ▶ `module_touch_controller_server`
-

The component `sc_lcd` includes the modules `module_lcd`, `module_text_display`, `module_touch_controller_lib` and the `module_touch_controller_server`.

3.1 `module_lcd`

3.1.1 Configuration Defines

The `module_lcd` includes device support defines, each support header, located in the `devices` directory defines a number of parameters. It is sufficient for the user to specify which device to support in the `lcd_conf.h` for the device to be correctly supported. To do this `lcd_conf.h` must include the define: `:: #define LCD_PART_NUMBER p`

- ▶ `AT043TN24V7`
- ▶ `K430WQAV4F`
- ▶ `K70DWN0V1F`

3.1.2 Implementation Specific Defines

It is possible to override the default defines when a part number is selected. The defines available are:

LCD_WIDTH

This define is used to represent the width of the LCD panel in pixels.

LCD_HEIGHT

This define is used to represent the height of the LCD panel in terms of lines.

LCD_BITS_PER_PIXEL

Count of bits used to set a pixels colour, i.e. if the screen was wired for `rgb565` then the `LCD_BITS_PER_PIXEL` would be 16, `rgb888` would be 24. This is independent of the actual bit depth of the lcd.

LCD_HOR_FRONT_PORCH

The horizontal front porch timing requirement given in pixel clocks.

LCD_HOR_BACK_PORCH

The horizontal back porch timing requirement given in pixel clocks.

LCD_VERT_FRONT_PORCH

The vertical front porch timing requirement given in horizontal time periods.

LCD_VERT_BACK_PORCH

The vertical back porch timing requirement given in horizontal time periods.

LCD_HOR_PULSE_WIDTH

The horizontal pulse width timing requirement given in pixel clocks. This is the duration that the hsync signal should go low to denote the start of the horizontal frame. Set to 0 when hsync is not necessary.

LCD_VERT_PULSE_WIDTH

The vertical pulse width timing requirement given in vertical time periods. This is the duration that the vsync signal should go low to denote the start of the vertical frame. Set to 0 when vsync is not necessary.

LCD_FREQ_DIVIDEND

The defines `FREQ_DIVIDEND` and `FREQ_DIVISOR` are used to calculate the frequency of the clock used for LCD. The frequency configured = $(\text{FREQ_DIVIDEND} / \text{FREQ_DIVISOR})$ in MHz

LCD_FREQ_DIVISOR

The defines `FREQ_DIVIDEND` and `FREQ_DIVISOR` are used to calculate the frequency of the clock used for LCD. The frequency configured = $(\text{FREQ_DIVIDEND} / \text{FREQ_DIVISOR})$ in MHz

LCD_FAST_WRITE

The define enables a faster LCD write function, however, it produces more code. Use when a 25MHz pixel clock is required.

3.1.3 API

- ▶ `lcd.xc`
- ▶ `lcd.h`
- ▶ `lcd_defines.h`
- ▶ `lcd_assembly.S`
- ▶ `/devices`

where the following functions can be found:

```
void lcd_init(chanend c_lcd)
    LCD init function.
```

This sets the lcd into a state where it is ready to accept data.

This function has the following parameters:

`c_lcd` The channel end connecting to the lcd server.

```
static void lcd_req(chanend c_lcd)
    Receives the request for data from the LCD server.
```

This function has the following parameters:

`c_lcd` The channel end connecting to the lcd server.

```
static void lcd_update(chanend c_lcd, unsigned buffer[])
    LCD update function.
```

This sends a buffer of data to the lcd server to to sent to the lcd.

Note, no array bounds checking is performed.

This function has the following parameters:

`c_lcd` The channel end connecting to the lcd server.

`buffer[]` The data to to emitted to the lcd screen, stored in rgb565.

```
static void lcd_update_p(chanend c_lcd, unsigned buffer)
```

C interface for LCD update function.

This sends a buffer of data to the lcd server to to sent to the lcd.

Note, no array bounds checking is performed.

This function has the following parameters:

`c_lcd` The channel end connecting to the lcd server.

`buffer` A pointer to data to to emitted to the lcd screen, stored in rgb565.

```
void lcd_server(chanend client, lcd_ports &ports)
```

The LCD server thread.

This function has the following parameters:

`client` The channel end connecting to the client.

`ports` The structure carrying the LCD port details.

3.2 module_touch_controller_lib

The device-specific configuration defines and user defines are listed in `touch_lib_conf.h`.

3.2.1 Configuration Defines

TOUCH_LIB_LCD_WIDTH

This define is used to represent the width of the LCD panel in pixels.

TOUCH_LIB_LCD_HEIGHT

This define is used to represent the height of the LCD panel in terms of lines.

TOUCH_LIB_TS_WIDTH

This define is used to represent the width of the touch screen in points.

TOUCH_LIB_TS_HEIGHT

This define is used to represent the height of the touch screen in points.

3.2.2 User Define

TOUCH_LIB_TIME_OUT

This define is used to set the time in seconds for the time-out message to display.

3.2.3 API

- ▶ touch_controller_lib.xc
- ▶ touch_controller_lib.h
- ▶ /AD7879-1

where the following functions can be found:

```
void touch_lib_init(touch_controller_ports &ports)
```

The touch controller initialisation.

This function has the following parameters:

ports The structure containing the touch controller port details.

```
{unsigned,unsigned} get_touch_coordinates(r_i2c &i2c_ports)
```

Get touch coordinates from the result registers of touch controller.

This function has the following parameters:

i2c_ports The structure containing the I2C port details.

```
select touch_lib_req_next_coord(touch_controller_ports &ports,  
                                unsigned &ts_x,  
                                unsigned &ts_y)
```

The function to fetch the next touch coordinates from the touch screen controller.

This function has the following parameters:

ports The structure containing the touch controller port details.

ts_x The X coordinate of point of touch.

ts_y The Y coordinate of point of touch.

```
void touch_lib_req_next_coord_timed(touch_controller_ports &ports,  
                                    unsigned &ts_x,  
                                    unsigned &ts_y,  
                                    unsigned &time,  
                                    timer t)
```

The function to fetch the next touch coordinates from the touch screen controller.

The delay in touch event is also computed.

This function has the following parameters:

ports The structure carrying the LCD port details.

<code>ts_x</code>	The X coordinate of point of touch.
<code>ts_y</code>	The Y coordinate of point of touch.
<code>time</code>	The delay in touch event in seconds.
<code>t</code>	The timer used to compute the delay in touch event.

```
select touch_lib_next_coord_timed(touch_controller_ports &ports,
                                unsigned &ts_x,
                                unsigned &ts_y,
                                unsigned &nSec,
                                timer t,
                                unsigned &timerCount,
                                unsigned &touched)
```

The function called by another function to fetch the next touch coordinates from the touch screen controller.

The delay in touch event is also computed.

This function has the following parameters:

<code>ports</code>	The structure carrying the LCD port details.
<code>ts_x</code>	The X coordinate of point of touch.
<code>ts_y</code>	The Y coordinate of point of touch.
<code>time</code>	The delay in touch event in seconds.
<code>t</code>	The timer used to compute the delay in touch event.
<code>timerCount</code>	A counter variable.
<code>touched</code>	The flag that records the touch status.

```
void scale_coords(unsigned &x, unsigned &y)
```

The function to scale coordinate values (from the touch point coordinates to the LCD pixel coordinates).

This function has the following parameters:

<code>x</code>	The X coordinate value
<code>y</code>	The Y coordinate value

3.3 module_touch_controller_server

The device-specific configuration defines and user defines are listed in `touch_server_conf.h`.

3.3.1 Configuration Defines

TOUCH_SERVER_LCD_WIDTH

This define is used to represent the width of the LCD panel in pixels.

TOUCH_SERVER_LCD_HEIGHT

This define is used to represent the height of the LCD panel in terms of lines.

TOUCH_SERVER_TS_WIDTH

This define is used to represent the width of the touch screen in points.

TOUCH_SERVER_TS_HEIGHT

This define is used to represent the height of the touch screen in points.

3.3.2 User Define

TOUCH_SERVER_TIME_OUT

This define is used to set the time in seconds for the time-out message to display.

3.3.3 API

- ▶ touch_controller_server.xc
- ▶ touch_controller_server.h
- ▶ /AD7879-1

where the following functions can be found:



doxygenfunction: Cannot find function “touch_server_init” in doxygen xml output

```
{unsigned,unsigned} get_touch_coordinates(r_i2c &i2c_ports)
```

Get touch coordinates from the result registers of touch controller.

This function has the following parameters:

`i2c_ports` The structure containing the I2C port details.

```
void touch_controller_server(chanend c_server,
                           touch_controller_ports &ports)
```

The touch controller server function.

This function has the following parameters:

`c_server` The channel connecting the client API and the server.

`ports` The structure containing the touch controller port details.

```
select process_interrupt(touch_controller_ports &ports,
                        unsigned presentTimeSec,
                        t_status &touched,
                        unsigned &x,
```

```
    unsigned &y,  
    unsigned &touchTime)
```

The function that processes the pen-down interrupt generated by touching the screen.

This function has the following parameters:

ports	The structure containing the touch controller port details.
presentTimeSec	The current time in seconds relative to the call of touch controller server function.
touched	The flag that records the touch status.
x	The X coordinate value.
y	The Y coordinate value.
touchTime	The time of touch in seconds relative to the call of touch controller server function.

```
t_status touch_server_get_last_coord(chanend c_ts, unsigned &x, unsigned &y)
```

The function to get the last touch coordinates from the touch controller server function.

This function has the following parameters:

c_ts	The channel connecting this client API and the server.
x	The X coordinate value.
y	The Y coordinate value.

```
void touch_server_get_next_coord(chanend c_ts, unsigned &x, unsigned &y)
```

The function to get the next touch coordinates from the touch controller server function.

This function has the following parameters:

c_ts	The channel connecting this client API and the server.
x	The X coordinate value.
y	The Y coordinate value.

```
t_status touch_server_get_last_coord_timed(chanend c_ts,  
                                           unsigned &t,  
                                           unsigned &x,  
                                           unsigned &y)
```

The function to get the last touch coordinates from the touch controller server function.

The time elapsed from the last touch event is also computed.

This function has the following parameters:

<code>c_ts</code>	The channel connecting this client API and the server.
<code>t</code>	The time elapsed from the last touch event in seconds.
<code>x</code>	The X coordinate value.
<code>y</code>	The Y coordinate value.

```
void scale_coords(unsigned &x, unsigned &y)
```

The function to scale coordinate values (from the touch point coordinates to the LCD pixel coordinates).

This function has the following parameters:

<code>x</code>	The X coordinate value
<code>y</code>	The Y coordinate value

4 Programming Guide

IN THIS CHAPTER

- ▶ Source code structure
 - ▶ Additional Files
 - ▶ How to select the LCD target
 - ▶ Executing The Project
 - ▶ Software Requirements
-

This section provides information on how to program applications using the LCD module.

4.1 Source code structure

Project	File	Description
module_lcd	lcd.h	Header file containing the APIs for the LCD component
	lcd.xc	File containing the implementation of the LCD component
	lcd_defines.xc	Header file containing the user configurable defines for the LCD
	lcd_assembly.S	Assembly file containing the fast_write functionality for the LCD.
	/devices	Folder containing header files of configurations for LCDs

Figure 1:
Project
structure

4.2 Additional Files

4.3 How to select the LCD target

The module has been designed to support multiple LCD targets. Each target has a specific configuration and have been provided with the component in the /devices directory. The module only supports a single LCD target per XCore.

To select the target the following should be done:

- ▶ Create a header in the application project called `lcd_conf.h`
- ▶ In the `lcd_conf.h` add the define `#define LCD_PART_NUMBER AT043TN24V7`. This will include the “`lcd_defines_AT043TN24V7.h`” required for the selected target.

- ▶ Any specific overrides should be added to the `lcd_conf.h`. For example, to override the `LCD_HEIGHT` to 600 pixels add the line `#define LCD_HEIGHT 600`.
- ▶ The application should also include the port mapping for the LCD as per the hardware used. A variable of the type structure `lcd_ports` should be created and must include the port information

Example: In the application file

```
struct lcd_ports lcd_ports = {
    XS1_PORT_1G,
    XS1_PORT_1F,
    XS1_PORT_16A,
    XS1_PORT_1B,
    XS1_PORT_1C,
    XS1_CLKBLK_1
};
```

The declared variable `lcd_ports` is used by the LCD server call to address these ports. A core should have the `lcd_server` running on it and it should be connected by a channel to the application, for example:

```
chan c_lcd;
par {
    lcd_server(c_lcd, lcd_ports);
    application(c_lcd);
}
```

4.4 Executing The Project

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

1. The module name `module_lcd` should be added to the list of `MODULES` in the application project build options.
2. Now the module is linked to the application and can be directly used

4.5 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 Touch Controller Programming Guide

IN THIS CHAPTER

- ▶ Source code structure
 - ▶ How to develop an application
 - ▶ Executing The Project
 - ▶ Software Requirements
-

This section provides information on how to program applications using the touch controller module.

5.1 Source code structure

5.2 How to develop an application

The modules have been designed to support two types of interfacing with the touch screen controller; one for direct interfacing and the other for interfacing through a server. Only one of these two modules should be used by the application program.

- ▶ Create a header file in the application project called `touch_lib_conf.h` or `touch_server_conf.h`.
- ▶ In the header file, add the defines for conditional compilation and device-specific parameters.
- ▶ The application should also include the port mapping for the touch screen controller. A variable of the type structure `touch_controller_ports` should be created and must include the port information.

Example: In the application file

```
struct touch_controller_ports ports = {
    XS1_PORT_1E,
    XS1_PORT_1H,
    1000,
    XS1_PORT_1D
};
```

When `module_touch_controller_server` is used, a core should have the `touch_controller_server` running on it and it should be connected by a channel to the application, for example:

```
chan c;
```

Project	File	Description
module_touch_controller_lib	touch_controller_lib.h	Header file containing the APIs for interfacing touch controller component
	touch_controller_lib.xc	File containing the implementation of APIs
	/AD7879-1	Folder containing files for the implementation of touch controller component
	touch_controller_impl.h	Header file containing the APIs for implementing touch controller component
	touch_controller_impl.xc	File containing the implementation of touch controller component
module_touch_controller_server	touch_controller_server.h	Header file containing the APIs for interfacing touch controller component
	touch_controller_server.xc	File containing the implementation of APIs
	/AD7879-1	Folder containing files for the implementation of touch controller component
	touch_controller_impl.h	Header file containing the APIs for implementing touch controller component
	touch_controller_impl.xc	File containing the implementation of touch controller component

Figure 2:
Project
structure

```
par {
    touch_controller_server(c, ports);
```

```
    app(c);  
}
```

5.3 Executing The Project

The touch controller module by itself cannot be built or executed separately. It must be linked into an application. The application also depends on I2C module. Once the modules are linked to the application, the application can be built and run.

1. The module name `module_touch_controller_lib` or `module_touch_controller_server` should be added to the list of MODULES in the application project build options.
2. The module name `module_i2c_master` should also be added.
3. Now the modules are linked to the application and can be directly used

5.4 Software Requirements

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

6 Example Applications

IN THIS CHAPTER

- ▶ `app_lcd_demo`
 - ▶ Application Notes
 - ▶ `app_touch_controller_lib_demo`
 - ▶ Application Notes
 - ▶ `app_touch_controller_server_demo`
 - ▶ Application Notes
-

This tutorial describes the demo applications included in the XMOS LCD software component. §2.1 describes the required hardware setups to run the demos.

6.1 `app_lcd_demo`

This application demonstrates how the module is used write image data to the LCD screen. The purpose of this application is to show how data is passed to the `lcd_server`

6.2 Application Notes

1. `lcd_server` requires a single logical core.
2. `lcd_init` must be called before any of `lcd_update`, `lcd_update_p` or `lcd_update` are called. This puts the LCD server into a state ready to accept data.
3. `lcd_update` and `lcd_update_p` are used to send an array of pixel data to the LCD server. There is a real-time requirement that this function is called often enough to maintain the display. `lcd_update_p` is the C interface to the LCD server, it takes a pointer to an array rather than the array itself.
4. `lcd_req` is a function (also a select handler) that acknowledges the LCDs request for the next line of pixel data.
5. The LCD server does no buffering of pixel line arrays, therefore, for every `lcd_req` there must be only one `lcd_update` or `lcd_update_p`. Likewise for every `lcd_update` or `lcd_update_p` there must be only one `lcd_req`.
6. The pixel array must be on the same tile as the `lcd_server`.

6.2.1 Getting Started

1. Plug the XA-SK-LCD Slice Card into the 'STAR' slot of the Slicekit Core Board

2. Plug the XA-SK-XTAG2 Card into the SliceKit Core Board.
3. Ensure the XMOS LINK switch on the XA-SK-XTAG2 is set to “off”.
4. Ensure the jumper on the XA-SK-SCR480 is bridged if the back light is required.
5. Open `app_lcd_demo.xc` and build the project.
6. Run the program

The output produced should look like a bouncing “X” on the LCD screen.

6.3 `app_touch_controller_lib_demo`

This application demonstrates how the module `module_touch_controller_lib` is used to fetch the touch coordinates from the touch screen controller.

6.4 Application Notes

1. `touch_lib_init` must be called before calling `touch_lib_req_next_coord` or/and `touch_lib_req_next_coord_timed`.
2. `touch_lib_req_next_coord` and `touch_lib_req_next_coord_timed` wait for touch event and then read the touch coordinates stored in the result registers of touch screen controller. `touch_lib_req_next_coord_timed` computes the time delay in touch event from the function call.

6.4.1 Getting Started

1. Plug the XA-SK-LCD Slice Card into the ‘TRIANGLE’ slot of the SliceKit Core Board
2. Plug the XA-SK-XTAG2 Card into the SliceKit Core Board.
3. Click on the `app_touch_controller_lib_demo` and build the project.
4. Run the demo.

6.5 `app_touch_controller_server_demo`

This application demonstrates how the module `module_touch_controller_server` is used to fetch the touch coordinates from the touch screen controller. The purpose of this application is to show how commands are passed from the application client to the `touch_controller_server`.

6.6 Application Notes

1. `touch_controller_server` requires a single logical core.
2. `touch_server_get_next_coord` wait for touch event and then read the touch coordinates stored in the result registers of touch screen controller.

3. `touch_server_get_last_coord` and `touch_server_get_last_coord_timed` get the last touch coordinates from the server. `touch_server_get_last_coord_timed` gets the time elapsed from the last touch event.

6.6.1 Getting Started

1. Plug the XA-SK-LCD Slice Card into the 'TRIANGLE' slot of the Slicekit Core Board
2. Plug the XA-SK-XTAG2 Card into the Slicekit Core Board.
3. Click on the `app_touch_controller_server_demo` and build the project.
4. Run the demo.



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.