#### **Overview**

In this session we will implement the touch screen IO and discover how to display (x, y) points detected whenever pressure is applied – sending them to the Terminal instrument in Altium Designer.

## **Prerequisites**

This tutorial assumes you have an elementary understanding of how to create a System-on-FPGA design in Altium Designer using OpenBus and Schematic Documents, how to configure peripherals and break-out their connections, as well as create a linked embedded project containing the Software Platform Builder. No additional information is required.

### **Design detail**

This exercise uses the following components:

Component	Library	Name in Library	
TSK3000A TSK-3000A 32-bit RISC Processor	OpenBus Palette	TSK3000A	
WB_INTERCON			
	OpenBus Palette	Interconnect	
WB_MULTIMASTER			
	OpenBus Palette	Arbiter	
WB_MEM_CTRL_SRAM			
	OpenBus Palette	SRAM Controller	
TERMINAL			
	OpenBus Palette	Terminal instrument	



WB_ILI9320	OpenBus Palette	VGA 32-Bit ILI9320	
WB_SPI_1	OpenBus Palette	SPI	
WB_TSPENDOWN	OpenBus Palette	Touchscreen Pen Control	
	FPGA Generic.IntLib	IOBUF16B	
*	FPGA Generic.IntLib	INV	
(ILK_BRD)>>-	FPGA NB3000 Port-Plugin.IntLib	CLOCK_BOARD	
TFT_DB[15.0] TFT_RESET TFT_RESET TFT_RS TFT_RS TFT_RS TFT_RS TFT_CS TFT_BLIGHT TFT_BLIGHT	FPGA NB3000 Port-Plugin.IntLib	TFT_LCD	
TFT_TSC_BUSY >>- TFT_TSC_CI.K <- TFT_TSC_CS_N <- TFT_TSC_DOUT <- TFT_TSC_DIN >>- TFT_TSC_IRQ_N >>-	FPGA NB3000 Port-Plugin.IntLib	TFT_PEN	
SRAM0_D[15.0]     SRAM0_A[17.0]     SRAM0_A[17.0]     SRAM0_W     SRAM0_OE     SRAM0_UB     SRAM0_UB     SRAM0_LB     SRAM1_LB     SRAM1_E     SRAM1_E     SRAM1_CF	FPGA NB3000 Port-Plugin.IntLib	SRAM0	
-> SRAMI_UB		SKAM1	
TEST_BUTTON >	FPGA NB3000 Port-Plugin.IntLib	TEST_BUTTON	

Table 1. List of components required by the design

## Tutorial steps – preparing the hardware

- 1. Create a new FPGA project in a new folder and name it **TFT\_TOUCH.PrjFpg**. Add a new OpenBus document and a Schematic document, saving them respectively as **TFT\_TOUCH\_OB.OpenBus** and **TFT\_TOUCH\_TOP.SchDoc**.
- 2. In the OpenBus document, create a system design with the following components from the OpenBus Palette:
  - a. TSK-3000A CPU
  - b. Wishbone Interconnect (x2)
  - c. Wishbone Arbiter
  - d. Terminal Instrument
  - e. Touchscreen Pen Control
  - f. SPI Controller
  - g. VGA 32-Bit ILI9320
  - h. SRAM Controller



Figure 1. OpenBus System for the TFT\_TOUCH design.

- 3. Join up the components as shown in Figure 1.
- 4. Configure the SRAM controller (called XRAM in Figure 1) to be Asynchronous SRAM, 1 MB (256K x 32-bit), 2 x 16-bit Wide Devices.
- 5. Configure the SPI device to use 32-bit transfers and uncheck the Enable Mode Pin option.
- 6. Make sure the TFT controller (designator WB\_ILI9320\_1) is the high-priority master (0) to the Arbiter (designator WB\_MULTIMASTER\_1), then save your work.
- Select Tools»OpenBus Signal Manager from the menus, to open the OpenBus Signal Manager dialog. Select the Interrupts tab and assign the ILI9320 Controller (TFT controller) interrupt to INT\_I0 of the TSK3000A.
- 8. In the schematic document, add all the components listed in Table 1.
- 9. Add the necessary Sheet Symbol for the OpenBus system document using **Design**»**Create Sheet Symbol** from Sheet of HDL.



	TFT DE[15.0] = 1 TFT RESET TFT WR TFT RS TFT RS TFT RS TFT RS TFT SC USY TFT TSC USY TFT TSC CS N TFT TSC CS N TFT TSC DOUT	VI IOBUF16B SPI_W_NB SPI_CLK SPI_CLK SPI_CUK	rGA32_ILI9320 BUF_[115.0] BUF_[0[15.0] BUF_TRI nRESET nWR RS nRD nCS	U_TFT_LEDs_OB TFT_TOUCH_OB.OpenBus XRAM_MEM( WB_IL19320_1 XRAM_MEMI	WB MEM	L_CTRL_SRAMI6_A D[15.0] (CCE UB UB UB LB LB LCTRL_SRAMI6_A D[15.0] (CTRL_SRAMI6_A D[15.0] CE WE UE	18         → SRAM0 D[15.0]         → SRAM0 E         → SRAM0 W         → SRAM0 OE         → SRAM0 UB         → SRAM0 LB         → SRAM1 D[15.0]         → SRAM1 A[17.0]         → SRAM1 E         → SRAM1 OE	MSING SAMSUNG
<b>2</b> @	TEST_BUTTON			- TOUCHSCR_PENDOWN		LB ——		<u> </u>

Figure 2. Top-level schematic for the TFT\_TOUCH design.

- 10. Wire up the top-level schematic as shown in Figure 2, using the pre-defined Harness Connectors as done in prior sessions.
- 11. Save your work using File»Save All.
- 12. Compile the FPGA project to ensure there are no errors in your wiring.

#### **Tutorial Steps – Setting up the Embedded Project**

- 13. Create a new embedded project in the workspace and save it as Embedded\_TFT\_TOUCH.PrjEmb.
- 14. Use the **Structure Editor** view in **Projects** panel to link the new embedded project to your FPGA project, as done in earlier sessions.
- 15. Add a new Software Platform Builder document to the project and name it TFT\_TOUCH.SwPlatform.
- 16. In the TFT\_TOUCH.SwPlatform document, import the C code wrappers from the FPGA project, and grow the software stacks up to contain the Graphics, Pointer and Serial IO services. It should look like Figure 3.



Pointer Context	4	1			ID SERIAL_1
POINTER_1	R				
Fouchscreen Contex	t	Ĩ			
TOUCHSCREEN_1					
AD7843 Touchscree	n Digitizer Driver	Graphics Context	Serial Device I/O	4	
DRV_AD7843_1		GRAPHICS_1	Context SERIAL_1	10101	
5PI Driver	Touchscreen Pen	YGA Module Driver	Virtual Terminal D	)river	
DRV_SPI_1	Pl	DRV_VGA_ILI9320_1	DRV_TERMINAL_1		
5PI Master Controlle	r	VGA TFT-Module	Virtual Terminal		
Import from FPG/	TOUCHSCR	WB_ILI9320_1	Instrument TERMINAL_1	ack Down	S <sup>®</sup> Link to Existing S
Import from FPG/	A OUCHSCR A Add New Wrapper.	Grow Stack Up	Instrument TERMINAL_1	ack Down	Contractions of the second sec
Import from FPG/	TOUCHSCR       A       B <td>Description</td> <td>TERMINAL_1</td> <td>ack Down</td> <td>Coptions: Generic Device</td>	Description	TERMINAL_1	ack Down	Coptions: Generic Device
Import from FPG/	A COUCHSCR A Add New Wrapper.	Controller     WB_ILI9320_1     Grow Stack Up      Description     Jupp Support for POSIX message co     Basic support for POSIX signals	Instrument TERMINAL_1	ack Down	Content of the second s
Import from FPG/	A Currently Ir Multithreading Support	Description Descr	Instrument TERMINAL_1	ack Down	Content of the second
Import from FPG/	A COUCHSCR A Add New Wrapper. A Add New Wrapper. Vices Included Service Not Currently Ir Message Queues S Not Currently Ir Signalling Support Not Currently Ir Multithreading Sup Always Includec Software Platform	Description Descr	mmunications ftware Platform	ack Down Related Stand stdin to Stand	Content of the second
Import from FPG/ Diftware Ser Optional Optional Optional System Automatic	A COUCHSCR A Add New Wrapper. A Add New Wrapper. A Service Not Currently Ir Message Queues S Not Currently Ir Signalling Support Not Currently Ir Multithreading Sup Always Includec Software Platform Included Graphics Services	Description Description Description Description Description Basic support for POSIX message co Basic support for POSIX multi-t Cor Global configuration for the Sol Graphics routines for line, recta	mmunications hreading ftware Platform angle, triangle, circ	Cack Down Related Stand stdin t Stand stdout	Content of the second s
Import from FPG/ Diftware Ser Optional Optional Optional System Automatic Automatic	TOUCHSCR         A	Description iup: Support for POSIX message co Basic support for POSIX multi-t Cor Global configuration for the Sol Graphics routines for line, recta Generic pointer routines	mmunications hreading ftware Platform angle, triangle, circ	ack Down Related Stand Stand Stand Stand Stand Stand Stand Stand Stand	Continue of the second
Import from FPG/ Type Optional Optional Optional System Automatic Automatic Automatic	TOUCHSCR         A	Description Descr	mmunications hreading ftware Platform inluding calibration	E Related Stand Stand Stand Stand Stand Stand	Contract of the second

Figure 3. Software Platform for the TFT\_TOUCH design.

- 17. In the TFT\_TOUCH.SwPlatform file, select the **Touchscreen Context**. In the right-hand panel ensure that the **Settings** field is set to NB3000, by clicking in the field and selecting from the drop-down that appears.
- 18. Next, select the Pointer Context, and in the right-hand panel set the Left Boundary and Top Boundary fields to 0, the Right Boundary to 320 and the Bottom Boundary to 240.
- 19. Select the Serial Device I/O Context and note it's ID. Then in the Software Services (lower) pane of the TFT\_TOUCH.SwPlatform file select Serial Device I/O Services. Check the boxes and link Standard Input, Standard Output and Standard Error to the SERIAL\_1 device (Terminal Instrument). This allows you to use C Standard IO printf() to put text on the terminal in Altium Designer. It should look like Figure 3.
- 20. Add a source code file main.c to your embedded project, and add the code to it as shown in Table 2. Save your work.



```
#include <stdio.h>
#include <graphics.h>
#include <touchscreen.h>
#include <pointer.h>
#include "generic_devices.h"
#include "devices.h"
#define WIDTH 320
#define HEIGHT 240
// Function prototypes
static void draw_mark(int x, int y, int width, int height, void *vp);
char *cal1 = "Touch screen at marker";
char *cal2 = "Calibration done";
// Pointers for LCD, Touch panel, and graphics drivers
graphics_t * display;
canvas_t * canvas;
touchscreen_t * tft_touch;
touchscreen_data_t * position;
touchscreen_callback_t callback;
pointer_t * ptr;
pointer_state_t * pointer_state;
void main (void)
{
    // Open instances of drivers for LCD, Touch and Graphics
    tft_touch = touchscreen_open(TOUCHSCREEN_1);
    ptr = pointer_open(POINTER_1);
    display = graphics_open(GRAPHICS_1);
    canvas = graphics_get_canvas(display, GRAPHICS_1);
    // Tell Touch driver which function to call for display during calibration
    touchscreen_set_callback(tft_touch, draw_mark, canvas);
while(!touchscreen_calibrate(tft_touch, 320, 240));
    graphics_fill_canvas(canvas, 0x080008);
    graphics_set_visible_canvas(display, canvas);
    while(1)
    ł
        if (pointer_update(ptr, pointer_state))
         {
             // Send point to Terminal in Devices View.
            printf( "Touched screen at point (%d,%d)\n",
                     pointer_state->x,
                     pointer_state->y );
        }
    }
}
// Calibration call-back function - used to display a cross-hair at
// Calibration points on TFT panel.
static void draw_mark(int x, int y, int width, int height, void *vp)
{
     graphics_draw_circle(canvas, x, y, 10, 0xff00ff);
     graphics_draw_line(canvas, x - 15, y, x + 15, y, 0x00fff);
     graphics_draw_line(canvas, x, y - 15, x, y + 15, 0x00ffff);
graphics_draw_string(canvas, 50, 50, call, NULL, 0xffffff, 0);
     graphics_set_visible_canvas(display, canvas);
```

Table 2. Code for Main.c



21. In the Embedded Project Options, go to **Configure Memory** tab, and split the external SRAM into 512K of ROM, and 512K of RAM, as shown in Figure 4



Figure 4. Embedded project memory configuration.





## **Tutorial Steps – running the design**

- 22. Open Devices View 🕙 you should see your connection to the NB3000. Right-click on the NB3000 icon and Auto-Configure your project. At this point it's a good idea to do **File**»**Save All**.
- 23. Build and download the project.
- 24. You will need to click the **Up to Date Download** button under the TSK-3000A soft-chain icon in Devices View once the hardware design has been downloaded.
- 25. Follow the prompts on-screen on the NB3000 TFT Panel for calibration.
- 26. Open the TERMINAL instrument in Devices View. Note the coordinates are received wherever you press the TFT panel.

#### Some additional items for you to try

- Referring to the Graphics Context documentation in the Knowledge Center panel, modify the code to draw lines between each point where the stylus (pen) touches the TFT screen.
- Modify the code to print the co-ordinates to the TFT screen as a text string, as well as to the Terminal Instrument.
- Modify the call-back function draw\_mark to display some shapes other than the cross-hair and circle. Try squares, triangles etc.
- Remove the Terminal Instrument from the design and re-build it to work just with the TFT for visual feedback.
- Note that the graphics drivers used in this project make use of "double buffering" where you can change screens on the TFT rapidly by switching canvases (a canvas is a screen-sized pixel buffer in memory that the TFT panel reads it's pixel data from). See if you can find where in the SwPlatform file you can specify how many canvas buffers you want.



# **Revision History**

Date	Revision No.	Changes
29-Jul-2009	1.0	New document release

Software, hardware, documentation and related materials:

Copyright © 2009 Altium Limited.

All rights reserved. You are permitted to print this document provided that (1) the use of such is for personal use only and will not be copied or posted on any network computer or broadcast in any media, and (2) no modifications of the document is made. Unauthorized duplication, in whole or part, of this document by any means, mechanical or electronic, including translation into another language, except for brief excerpts in published reviews, is prohibited without the express written permission of Altium Limited. Unauthorized duplication of this work may also be prohibited by local statute. Violators may be subject to both criminal and civil penalties, including fines and/or imprisonment. Altium, Altium Designer, Board Insight, Design Explorer, DXP, LiveDesign, NanoBoard, NanoTalk, P-CAD, SimCode, Situs, TASKING, and Topological Autorouting and their respective logos are trademarks or registered trademarks of Altium Limited or its subsidiaries. All other registered or unregistered trademarks referenced herein are the property of their respective owners and no trademark rights to the same are claimed.

