SmartFusion2 based Serial Display Solution - Using OpenGL SC Graphics Library and SPI interface User Guide

Table of Contents

Purpose

This application note describes how to interface a SPI based serial display device to SmartFusion®2 device and how to run customized Open Graphic Library Safety Critical (OpenGL SC) graphics library based application using built-in ARM® Cortex®-M3 processor.

References

The following references are used in this document:

http://www.khronos.org/registry/glsc/

http://www.ftdichip.com/Products/ICs/FT800.html

Microsemi Publications

UG0331: SmartFusion2 Microcontroller Subsystem User Guide

UG0594: M2S090TS-EVAL-KIT SmartFusion2 Security Evaluation Kit User Guide

Introduction

The SPI based FTDI 5.0 inch LCD display unit is used to demonstrate a SmartFusion2 device executing OpenGL based application, as shown in Figure 1.

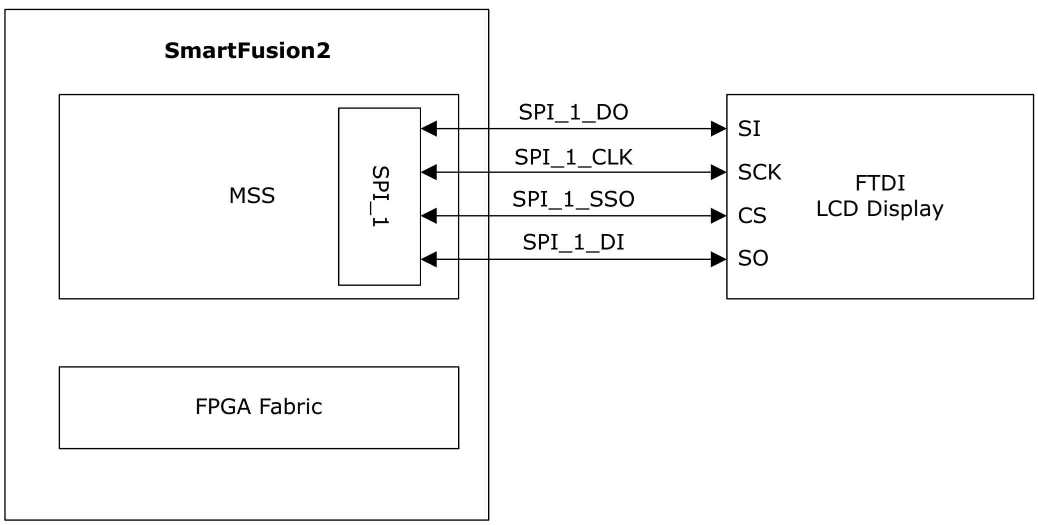

***Figure 1* • Device Executing OpenGL Based Application**

The demo design displays a real-time clock and a set of images on the display unit. The images (bit map) to be displayed are first transferred to the SmartFusion2 on board SPI Flash memory using the UART interface. The Cortex-M3 processor executes OpenGL SC graphics library APIs and creates the display frame buffer using the bit map images and transfers the frame buffer to FTDI display unit to display the graphical images.

Figure 2 shows the block diagram for the display solution using the SmartFusion2.

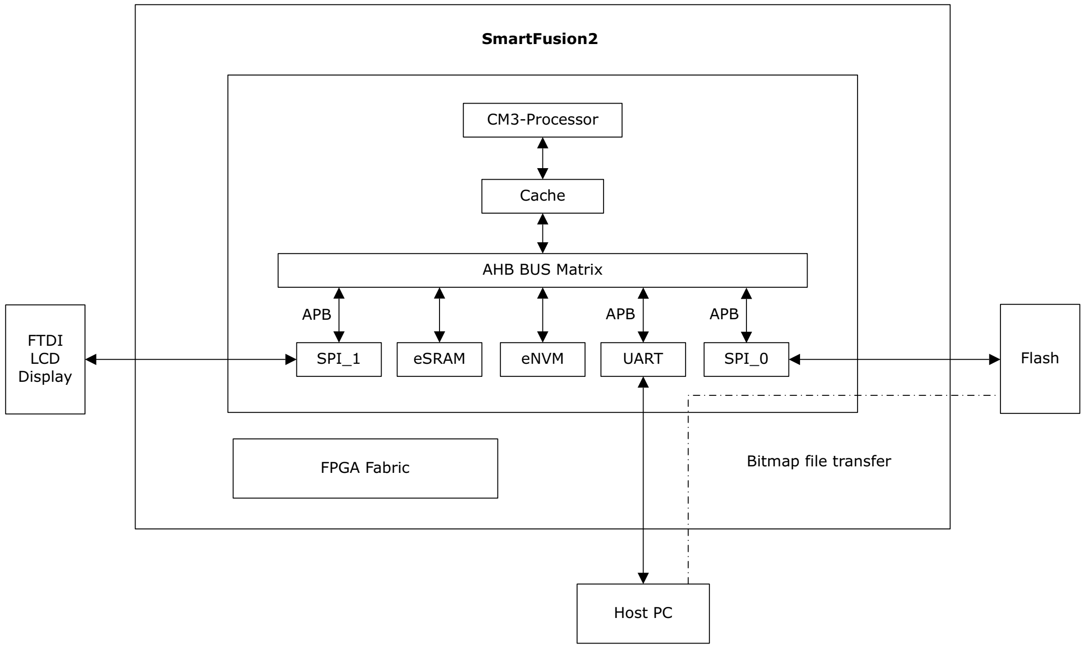

Figure 2 • **Display Solution Using the SmartFusion2**

Design Requirements

Table 1 **• Design Requirements**

Hardware Requirements	**Description**
SmartFusion2 Security Evaluation Kit: • Power adapter 12 V • FlashPro programmer • Two USB A to mini-B cables	Rev D
Host PC or Laptop	Any 64-bit Windows Operating System
Note: The following units are to be purchased separately and they are not part of the kit provided by Microsemi.	
FTDI LCD display 5.0 inch (VM800B50A)	Serial display unit
USB A to micro-B cable	To power up the display unit
Software Requirements	
Libero® System-on-Chip (SoC)	v11.7
Bit map Image loader	Software utility run on host PC
Operating System	64-bit Windows 7
Host PC Drivers	USB to UART drivers
SoftConsole	v3.4 SP1* *Note: *For this application note, SoftConsole v3.4 SP1 is used. For using SoftConsole v4.0, see the TU0546: SoftConsole v4.0 and Libero SoC v11.7 Tutorial.*

Design Description

This section describes the hardware and software implementation details of the demo. The demo design files are available for download from the Microsemi website:
http://soc.microsemi.com/download/rsc/?f=m2s_ac451_liberov11p7_df

The demo design files include:

- Libero SoC hardware project with SoftConsole firmware project
- STAPL programming file
- Image loader host PC utility
- Readme.txt file

Refer to the `readme.txt` file for complete directory structure of the design files.

Features

The demo has the following options to select using LCD display touch buttons:

Photo Album

The Cortex-M3 processor retrieves preloaded images from LPDDR and creates the frame buffer using OpenGL SC. The contents of the frame buffer transferred to the display unit to show the graphical images.

Analog Clock

The Cortex-M3 processor receives the Host PC system time through UART interface. The application adjusts the hour, minute, and second hands of the clock using Open GL primitives and creates the frame buffer using OpenGL SC. The contents of the frame buffer transferred to the display unit to show the analog clock. The clock is updated for every second to show the exact position of the hour, minute, and second hands.

FTDI Display Module

In this design, The FTDI LCD display 5.0 inch touch screen module (model: VM800B50A) is used. The FTDI display module supports SPI interface with maximum operating frequency of 30 MHz. It also supports 16 bpp with resolution of 480X272. For more information on display unit, refer to http://www.ftdichip.com/Products/ICs/FT800.html.

Hardware Implementation

The Libero hardware project configures the following SmartFusion2 MSS resources:

- **MMUART_0**: The MSS UART is configured for serial communication with Host PC. The image files are transferred from the Host PC to SmartFusion2 device using the UART interface.
- **SPI_0**: The MSS SPI_0 is interfaced to SmartFusion2 Security Evaluation Kit on board SPI flash. The images that are received using UART interface are stored in the SPI flash.
- **SPI_1**: The FTDI LCD display unit interfaces with MSS SPI_1 as a SPI slave device. The Cortex-M3 processor sends commands and data to the FTDI display unit using the SPI interface.
- **MDDR**: The microcontroller subsystem (MSS) DDR (MDDR) is configured for storing the display frame buffer contents and image files.

The FTDI display unit supports the SPI interface with maximum clock frequency of 30 MHz. In this demo, the MSS clock is configured for 120 MHz. MSS SPI_1 is configured to operate at 30 MHz to get the maximum data transfer speed between SmartFusion2 device and FTDI display unit.

Software Implementation

The OpenGL SC profile graphics application program interface (API) has been ported and modified for some of the features of the graphics library onto the Cortex-M3 processor. The graphics rendering operations are completely performed by the software implementation. The images to be displayed are created and drawn onto the frame buffer in the external RAM. The Cortex-M3 processor reads the frame buffer contents from the external RAM and sends it to the LCD module.

Introduction to OpenGL

OpenGL is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. This API is designed and controlled by Khronos Group (www.khronos.org).

OpenGL for Embedded Systems (OpenGL ES) is a subset of the desktop OpenGL graphic API designed for embedded systems such as mobile phones, PDAs, and video game consoles.

OpenGL SC applications are a subset of the OpenGL graphic API, designed to meet the needs of the safety critical market for avionics, industrial, military, medical, and automotive applications. OpenGL SC removes some functionality from OpenGL ES to minimize implementation and safety critical costs. It also adds functionality such as display lists.

Figure 3 shows the OpenGL pipeline for graphics rendering:

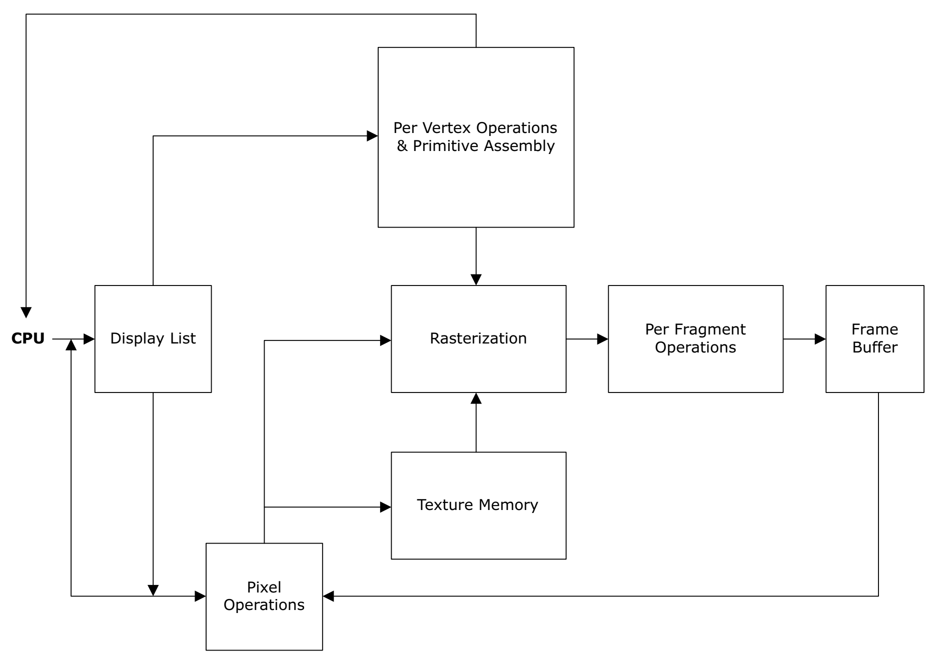

***Figure 3* • OpenGL Graphics Library Pipeline**

Figure 3 represents the flow of graphical information as it is processed from CPU to the frame buffer.

There are two pipelines of data flow. The upper pipeline is for geometric and vertex-based primitives. The lower pipeline is for pixel-based image primitives. Texturing combines these two types of primitives or flows together.

The OpenGL SC profile ported on the SmartFusion2 device supports both the flows of the OpenGL architecture.

Following are some of the graphical features supported by the OpenGL SC profile:

1. OpenGL SC v1.0 is derived from the OpenGL 1.3 spec. There are total 101 APIs for the graphics rendering are defined for this profiles. For more information about this API, refer to the following website: http://www.khronos.org/registry/glsc/.
2. Geometric primitives: POINTS, LINES, LINE STRIP, LINE LOOP, TRIANGLE, TRIANGLE FAN, and TRIANGLE STRIP
3. Texturing
4. Pixel operations: Read pixel, copy pixel, and draw pixel
5. Bitmaps
6. Display lists
7. Projections: Orthogonal and perspective projections
8. Advanced Vertex operations:
   - Scaling
   - Rotation
   - Translation
9. Scissoring
10. Multiple view ports
11. Coloring
12. Drawing multiple elements

The OpenGL SC port needs a surface to render the graphics; in this implementation we are using some of the features from libSDL for the surface creation. This surface will be used by the OpenGL SC engine.

Benefits:

- OpenGL SC minimizes implementation and safety critical costs.
- OpenGL SC requires low memory foot print.

Operational Details

The following procedure describe the operational flow of the application:

1. The Cortex-M3 processor configures the SPI interface and the display unit and sends command to the display unit to turn ON the back light of the display.
2. The application receives the `.bmp` image files and system current time using UART interface from Host PC.
3. The application stores all the images to SPI flash and copies the same to the LPDDR memory to access them easily to construct the frame buffer using OpenGL SC APIs.

Note: The image files are copied to the LPDDR memory from SPI flash memory to reduce the image data access time.

4. The application sends the frame buffer data to the display unit and enables the display RAM to show the graphical image on the LCD screen.
5. The application modifies the frame buffer contents depending on the user touch inputs using the OpenGL SC graphics APIs.

Figure 4 shows the operation flow of the design.

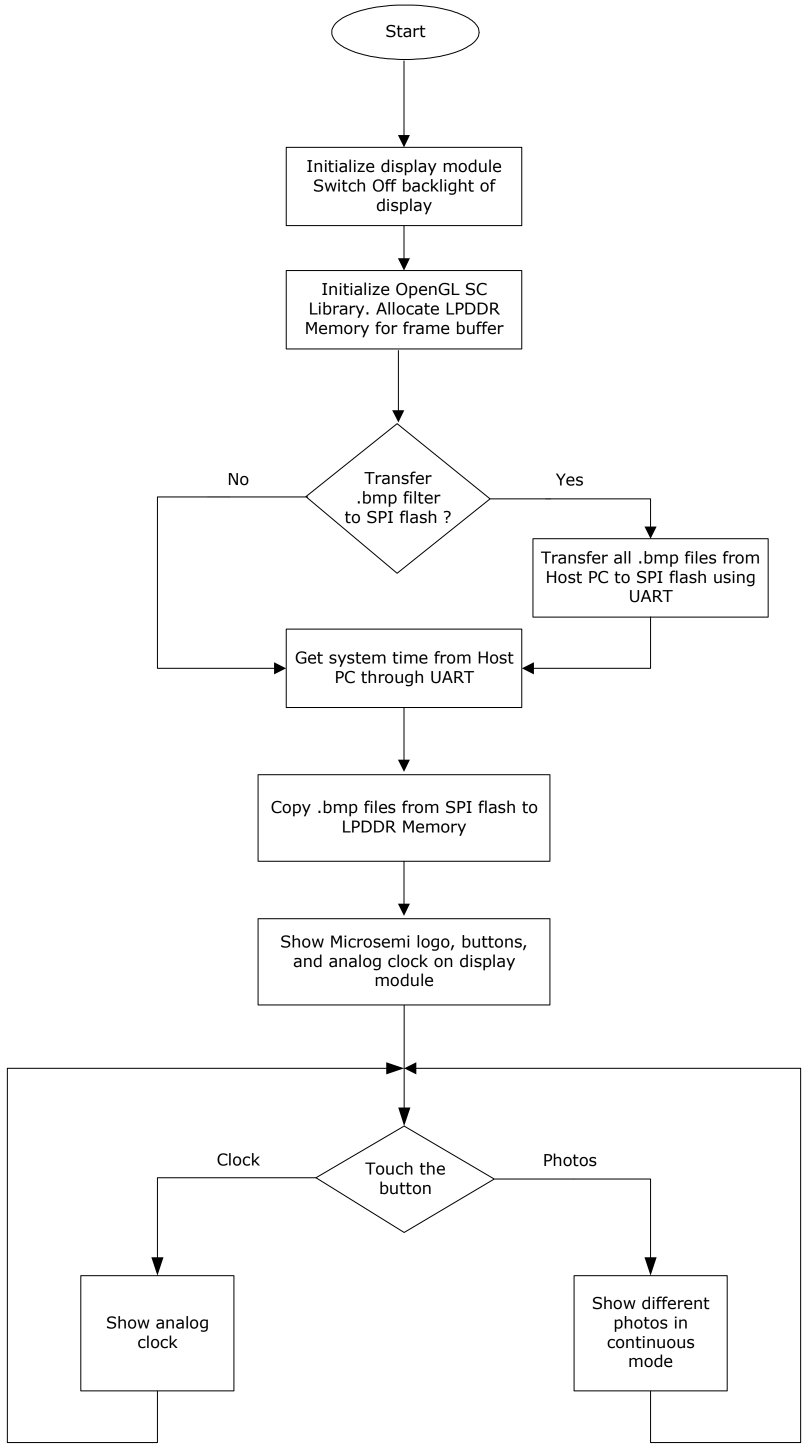

***Figure 4* • Operation Flow of the Design**

Graphics APIs for Surface and Windows Creations

Creating a surface

Function: vglCreateSurface

```
VGL_Surface vglCreateSurface(GLsizei width, GLsizei height, GLenum format, GLenum type,
GLenum depthStencilType)
```

Description

This API is used to create a surface for the OpenGL SC graphics rendering. OpenGL SC needs a surface instance for the graphics rendering. This API uses the minimal libSDL library for creating the surface.

Arguments:

-width

- height

The width and height of the LCD resolution. In the demo for the 5 inch LCD these values are 320, 240.

-format

Represents pixel format. This demo supports 16 BPP.

-type

Represents RGB information. This demo supports the RGB 565 format.

-depthStencilType

Represents depth and stencil information of the surface. This demo uses default value 0.

Enabling the current surface

Function: vglMakeCurrent;

```
GLboolean  vglMakeCurrent(VGL_Surface draw, VGL_Surface read)
```

Description

If we have multiple surfaces in the application, you need to enable the surface on which the subsequent drawing functions are to be reflected.

Arguments:

-draw

Pointer to the draw surface.

- read

Pointer to the read surface.

Setting the viewport/window

Function: glViewport;

```
void glViewport(GLint x, GLint y, GLsizei width, GLsizei height)
```

Description

You can have multiple independent viewports defined in a single surface and make the drawings using OpenGL SC API on those viewports.

Arguments:

-x

-y

Window coordinates of the viewports lower left corner.

- width

- height

Viewport’s width and height.

For more information about OpenGL SC APIs, refer to: "Appendix: OpenGL SC APIs" on page 16.

Memory Requirements for OpenGL SC Application using SmartFusion2

The minimum memory requirements for the graphics application are as follows:

- RAM of ~128 KB for the stack, heap, and data sections
- ROM of ~200 KB for the code and constant data sections
- RAM of ~255 KB for frame buffer storage of 5.0 inch LCD display (resolution: 480X272)

The SmartFusion2 Security Evaluation Kit Board has 64 MB of external LPDDR memory, which can be used for storing the bitmap images and frame buffer. The CODE and DATA memory size requirements may vary according to the application memory requirements. The 128 KB of LSRAM is used for the stack and data sections in this design.

Setting Up the Demo

Ensure that power supply switch SW7 is switched OFF before setting up the SmartFusion2 Security Evaluation Kit, then proceed with the following steps:

Programming the SmartFusion2 Device

The following procedure describes how to program the SmartFusion2 device with the provided programming file:

1. Connect the FlashPro4 programmer to the J5 connector of the SmartFusion2 Security Evaluation Kit board.
2. Launch the FlashPro software.
3. Click **New Project**.
4. In the New Project window, enter a **Project Name**.
5. Click **Browse** and navigate to the location where you want to save the project.
6. Select Single Device as the **Programming Mode**.
7. Click **OK** to save the project.
8. Click **Configure Device**.
9. Click **Browse** and navigate to the location where `display_demo.stp` file is located. The file is located in `<design files folder>\display_demo_df\programming_file` and the file name is `display_demo.stp`.
10. The required programming file is selected and is ready to be programmed in the device.
11. Click **PROGRAM** to start programming the device. An orange LED blinks as programming is in the progress. Wait until a message is displayed, which indicates that the program is passed.
12. Close the FlashPro software.

Board Setup

The following procedure describe how to interface the FTDI LCD display and SmartFusion2 Security Evaluation Kit board.

1. Power OFF the SmartFusion2 Security Evaluation Kit by changing SW7 switch position to OFF Connect the J18 connector (USB mini-B port) of the SmartFusion2 Security Evaluation Kit to the Host PC USB port using the USB A to mini-B cable. This connection establishes the UART communication between the SmartFusion2 device and the Host PC. Install the USB to UART bridge drivers on the Host PC, if the USB to UART bridge drivers are not detected automatically. The drivers are available for download at: http://www.ftdichip.com/Drivers/VCP.htm.
2. Ensure that the USB to UART bridge drivers are detected (verify in the Device Manager).

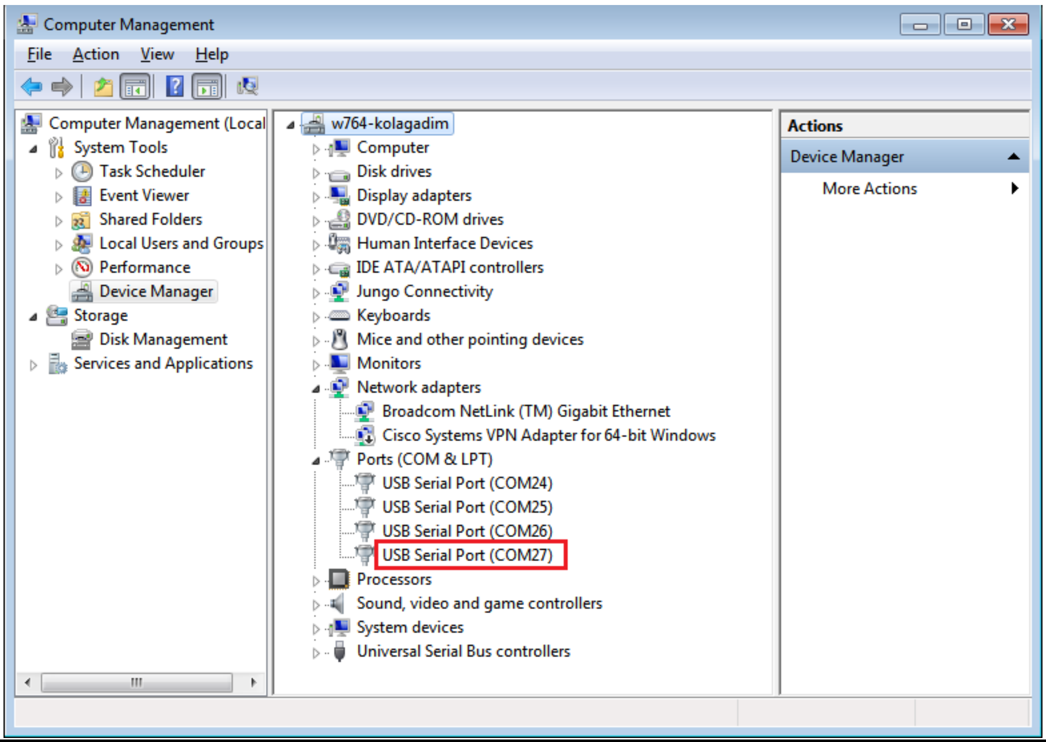

Figure 5 • **USB to UART Bridge Drivers**

3. From the Device Manager window, identify the COM port that is assigned to the SmartFusion2 UART interface. Figure 5 shows four COM ports with COM27 highlighted. Make a note of the highest numbered COM port to run demo. The COM port numbers vary from system to system.
4. Edit the batch file command of `load_images_thru_uart_Yes.bat` and `load_images_thru_uart_No.bat` files located in `<design files folder>\display_demo_df\host_pc_tool` with highest numbered COM port as shown in Table 2.

Right-click on file and select Edit option.

Table 2 • **File Name and Command**

File Name	Command
load_images_thru_uart_Yes.bat	spi_loader.exe 27 124558 y logo.bmp photos_but.bmp clk_but.bmp i1.bmp i2.bmp i3.bmp i4.bmp i5.bmp i6.bmp i7.bmp i8.bmp rose1.bmp rose2.bmp
load_images_thru_uart_No.bat	spi_loader.exe 27 124558 n logo.bmp photos_but.bmp clk_but.bmp i1.bmp i2.bmp i3.bmp i4.bmp i5.bmp i6.bmp i7.bmp i8.bmp rose1.bmp rose2.bmp

Save the files and close.

5. Connect the following pins of the SmartFusion2 J1 header to the FTDI display J5 header, as shown in Figure 6

Table 3 • **SmartFusion2 J1 Header to FTDI Display J5 Header**

SmartFusion2 J1 header	FTDI display J5 header
1	SCK
3	MISO
4	GND
6	MOSI
7	CS#

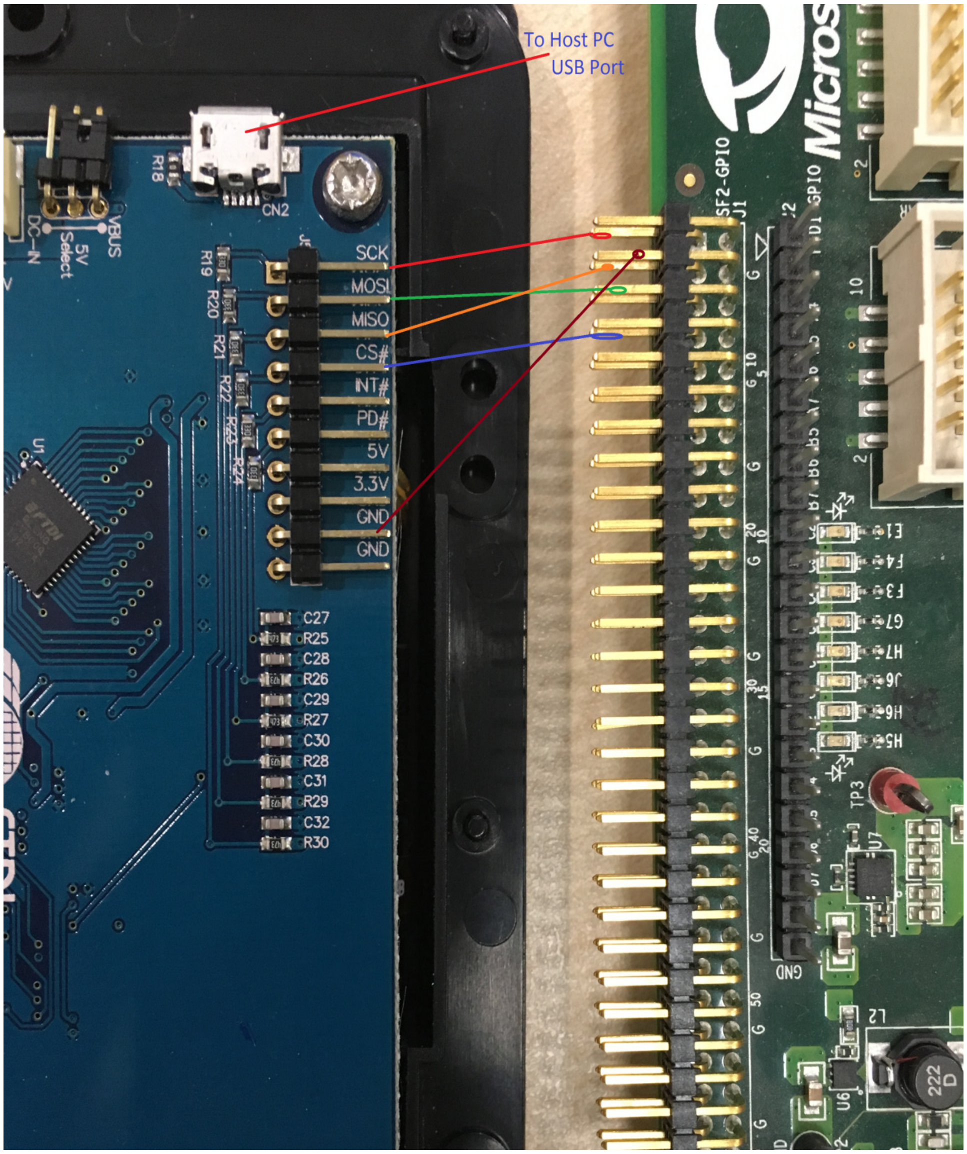

Figure 6 • SmartFusion2 J1 Header to FTDI Display J5 Header

Note: SmartFusion2 J1 header pin count starts from bottom where arrow head is pointing.

6. Connect the CN2 connector (micro USB port) of the FTDI display to the Host PC USB port using the USB A to micro-B cable. This connection powers up the FTDI display unit.

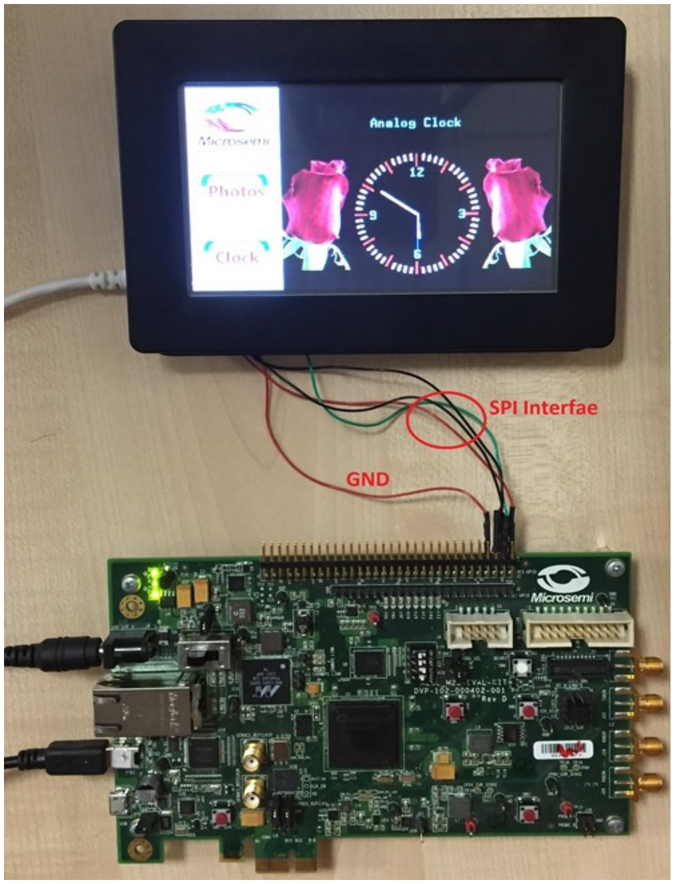

Figure 7 • **Board Setup**

Running the Demo

The following procedure describes how to run the demo on the SmartFusion2 Security Evaluation Kit board.

1. Switch ON the power supply switch, **SW7**.
2. Double click `load_images_thru_uart_Yes.bat` batch file to load the necessary .bmp images into SPI flash. This step needs to be done only once to store the images into SPI flash memory. This batch file executes the `spi_loader.exe` host PC application to transfer the Host PC system current time and .bmp images through UART. The `spi_loader.exe` host PC application runs for approximately 4 minutes. The next time you run the demo, the .bmp images are not necessary to transfer to SPI flash memory. Double click `load_images_thru_uart_No.bat` batch file to transfer only the Host PC system current time through UART.

3. Close the command window.

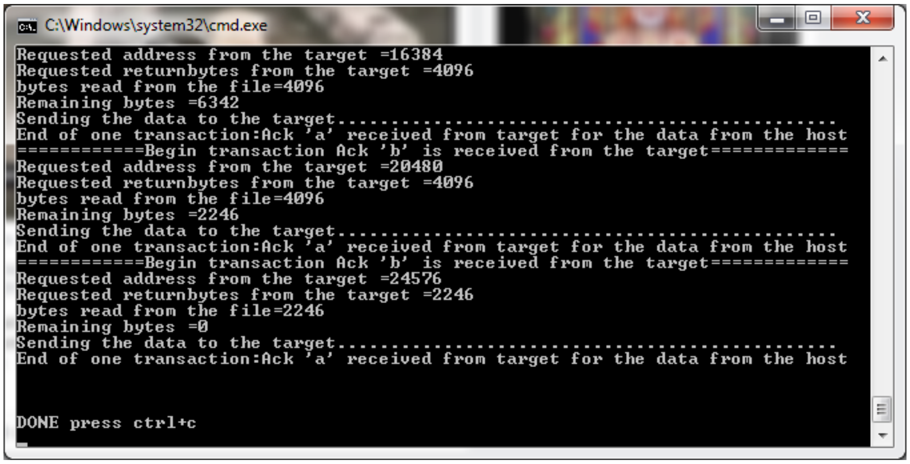

***Figure 8* • Running load_images_thru_uart_No.bat Batch File**

4. The FTDI display shows the analog clock with Microsemi logo and touch buttons, as shown in Figure 9.

***Figure 9* • FTDI Display Unit with Analog Clock**

Note: The demo application takes approximately two minutes to show the output on the display module.

5. Touch **Photos** should be used to switch the display unit to show a sequence of the photos in rotating mode where each image stays for 1.5 seconds.

Figure 10 **• FTDI Display Unit with Photos Album**

Conclusion

The application note explained the analog clock and continuous photos display features using OpenGL SC Graphics Library. This feature helps the users to develop the customized designs for their user applications.

Appendix: OpenGL SC APIs

1. GLAPI void APIENTRY glBegin(GLenum mode);
2. GLAPI void APIENTRY glBitmap (GLsizei width, GLsizei height, GLfloat xorig, GLfloat yorig, GLfloat xmove, GLfloat ymove, const GLubyte *bitmap);
3. GLAPI void APIENTRY glCallLists (GLsizei n, GLenum type, const GLvoid *lists);
4. GLAPI void APIENTRY glClear (GLbitfield mask);
5. GLAPI void APIENTRY glClearColor (GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha);GLAPI void APIENTRY glColor4f (GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha);
6. GLAPI void APIENTRY glColor4fv (const GLfloat *v);
7. GLAPI void APIENTRY glColor4ub (GLubyte red, GLubyte green, GLubyte blue, GLubyte alpha);
8. GLAPI void APIENTRY glColorMask (GLboolean red, GLboolean green, GLboolean blue, GLboolean alpha);
9. GLAPI void APIENTRY glCopyPixels (GLint x, GLint y, GLsizei width, GLsizei height, GLenum type);
10. GLAPI void APIENTRY glCullFace (GLenum mode);
11. GLAPI void APIENTRY glDisable (GLenum cap);
12. GLAPI void APIENTRY glDisableClientState (GLenum array);
13. GLAPI void APIENTRY glDrawArrays (GLenum mode, GLint first, GLsizei count);
14. GLAPI void APIENTRY glDrawElements (GLenum mode, GLsizei count, GLenum type, const GLvoid *indices);
15. GLAPI void APIENTRY glDrawPixels (GLsizei width, GLsizei height, GLenum format, GLenum type, const GLvoid *pixels);
16. GLAPI void APIENTRY glEnable (GLenum cap);
17. GLAPI void APIENTRY glEnableClientState (GLenum array);
18. GLAPI void APIENTRY glEnd (void);
19. GLAPI void APIENTRY glEndList (void);
20. GLAPI void APIENTRY glFinish (void);
21. GLAPI void APIENTRY glFlush (void);
22. GLAPI GLuint APIENTRY glGenLists (GLsizei range);
23. GLAPI GLenum APIENTRY glGetError (void);
24. GLAPI void APIENTRY glHint (GLenum target, GLenum mode);
25. GLAPI GLboolean APIENTRY glIsEnabled (GLenum cap);
26. GLAPI void APIENTRY glLineStipple (GLint factor, GLushort pattern);
27. GLAPI void APIENTRY glLineWidth (GLfloat width);
28. GLAPI void APIENTRY glListBase (GLuint base);
29. GLAPI void APIENTRY glLoadIdentity (void);
30. GLAPI void APIENTRY glLoadMatrixf (const GLfloat *m);
31. GLAPI void APIENTRY glMatrixMode (GLenum mode);
32. GLAPI void APIENTRY glMultMatrixf (const GLfloat *m);
33. GLAPI void APIENTRY glNewList (GLuint list, GLenum mode);
34. GLAPI void APIENTRY glNormal3f (GLfloat nx, GLfloat ny, GLfloat nz);
35. GLAPI void APIENTRY glNormal3fv (const GLfloat *v);
36. GLAPI void APIENTRY glNormalPointer (GLenum type, GLsizei stride, const GLvoid *pointer);
37. GLAPI void APIENTRY glOrthof (GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat zNear, GLfloat zFar);
38. GLAPI void APIENTRY glPixelStorei (GLenum pname, GLint param);
39. GLAPI void APIENTRY glPointSize (GLfloat size);

40. GLAPI void APIENTRY glPopMatrix (void);
41. GLAPI void APIENTRY glPushMatrix (void);
42. GLAPI void APIENTRY glRasterPos3f (GLfloat x, GLfloat y, GLfloat z);
43. GLAPI void APIENTRY glReadPixels (GLint x, GLint y, GLsizei width, GLsizei height, GLenum format, GLenum type, GLvoid *pixels);
44. GLAPI void APIENTRY glRotatef (GLfloat angle, GLfloat x, GLfloat y, GLfloat z);
45. GLAPI void APIENTRY glScalef (GLfloat x, GLfloat y, GLfloat z);
46. GLAPI void APIENTRY glScissor (GLint x, GLint y, GLsizei width, GLsizei height);
47. GLAPI void APIENTRY glShadeModel (GLenum mode);
48. GLAPI void APIENTRY glTranslatef (GLfloat x, GLfloat y, GLfloat z);
49. GLAPI void APIENTRY glVertex2f (GLfloat x, GLfloat y);
50. GLAPI void APIENTRY glVertex2fv (const GLfloat *v);
51. GLAPI void APIENTRY glVertex3f (GLfloat x, GLfloat y, GLfloat z);
52. GLAPI void APIENTRY glVertex3fv (const GLfloat *v);
53. GLAPI void APIENTRY glVertexPointer (GLint size, GLenum type, GLsizei stride, const GLvoid *pointer);
54. GLAPI void APIENTRY glViewport (GLint x, GLint y, GLsizei width, GLsizei height)

For more information about this API and its usage, refer to the following websites:
http://www.khronos.org/openglsc

Following are some useful links related to the OpenGL and licensing details:

- http://www.opengl.org/
- http://www.khronos.org/opengl
- http://www.khronos.org/opengles/
- http://www.khronos.org/openglsc
- http://www.sgi.com/company_info/trademarks/downloads/opengl_es_trademarks.pdf

List of Changes

The following table shows important changes made in this document for each revision.

Revision	Changes	Pages
Revision 1 (April 2016)	Initial release.	NA

**The part number is located on the last page of the document.*

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

Microsemi®
Power Matters.™

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
E-mail: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.

51900451-1/04.16