SmartFusion cSoC Display Solutions

Using OpenGL SC Graphics Library and Customizable Display Controller in FPGA Fabric User's Guide

Microsemi

Table of Contents

Introduction

The SmartFusion® customizable system-on-chip (cSoC) integrates FPGA technology with the hardened ARM® Cortex™-M3 processor based microcontroller subsystem (MSS) and programmable high-performance analog blocks built on a low power flash semiconductor process. The MSS consists of hardened blocks, such as a 100 MHz ARM Cortex-M3 processor, peripheral DMA (PDMA), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), embedded FlashROM (eFROM), external memory controller (EMC), watchdog timer, the Philips Inter-Integrated Circuit (I^2C), serial peripheral interface (SPI), 10/100 Ethernet controller, real-time counter (RTC), GPIO block, fabric interface controller (FIC), in-application programming (IAP), and system registers. The programmable analog block contains the analog compute engine (ACE) and analog front-end (AFE), consisting of ADCs, DACs, active bipolar prescalers (ABPS), comparators, current monitors, and temperature monitors.

This design example describes the display reference design using the SmartFusion cSoC device. In this design, Open Graphic Library Safety Critical (OpenGL SC) graphics library is running on the Cortex-M3 processor to create the graphic images for a graphical user interface (GUI) based application and FPGA fabric in SmartFusion cSoC is running the customizable display controller driver for driving the following LCDs:

1. 5.5 inch NEC TFT Color LCD Module NL3224BC35-20.
2. 7 inch Toshiba Color TFT-EL Module LTA070A320F displays.

The limitations of fixed-function devices for driving LCD panels can be overcome by replacing them with a combination of FPGAs and hard-core processor solutions. Programmable logic platforms, due to the inherent parallelism of the FPGA, are able to implement at low cost what would otherwise have required an expensive processor.

The flexible nature of FPGAs supports a variety of LCD panel requirements. For example, many different memory standards are supported by using the appropriate memory controllers. FPGA configurability also allows designers to change the LCD driver functionality after the product has been manufactured, enabling easy in-field product upgrades or enhancements.

Following are some of the key features of display solution using the SmartFusion cSoC:

- Fully customizable LCD controller. It drives different display panels regardless of interface, resolution, and manufacturer.
- Complex LCD controller with features such as color space conversion and Alpha blending.
- LCD backlight control
- Hardware acceleration using FPGA fabric: Tight integration of processor system and programmable logic.
- Hardware/software partitioning
- Advanced graphics with low processor horse power: Cortex-M3 processor at 75 MHz.
- Scalable: Interfacing with a camera sensor to capture data.

1 – Display Solutions

Block Diagram of the Display Solution Using the SmartFusion cSoC

Figure 1-1 shows the block diagram for a display solution using the SmartFusion cSoC.

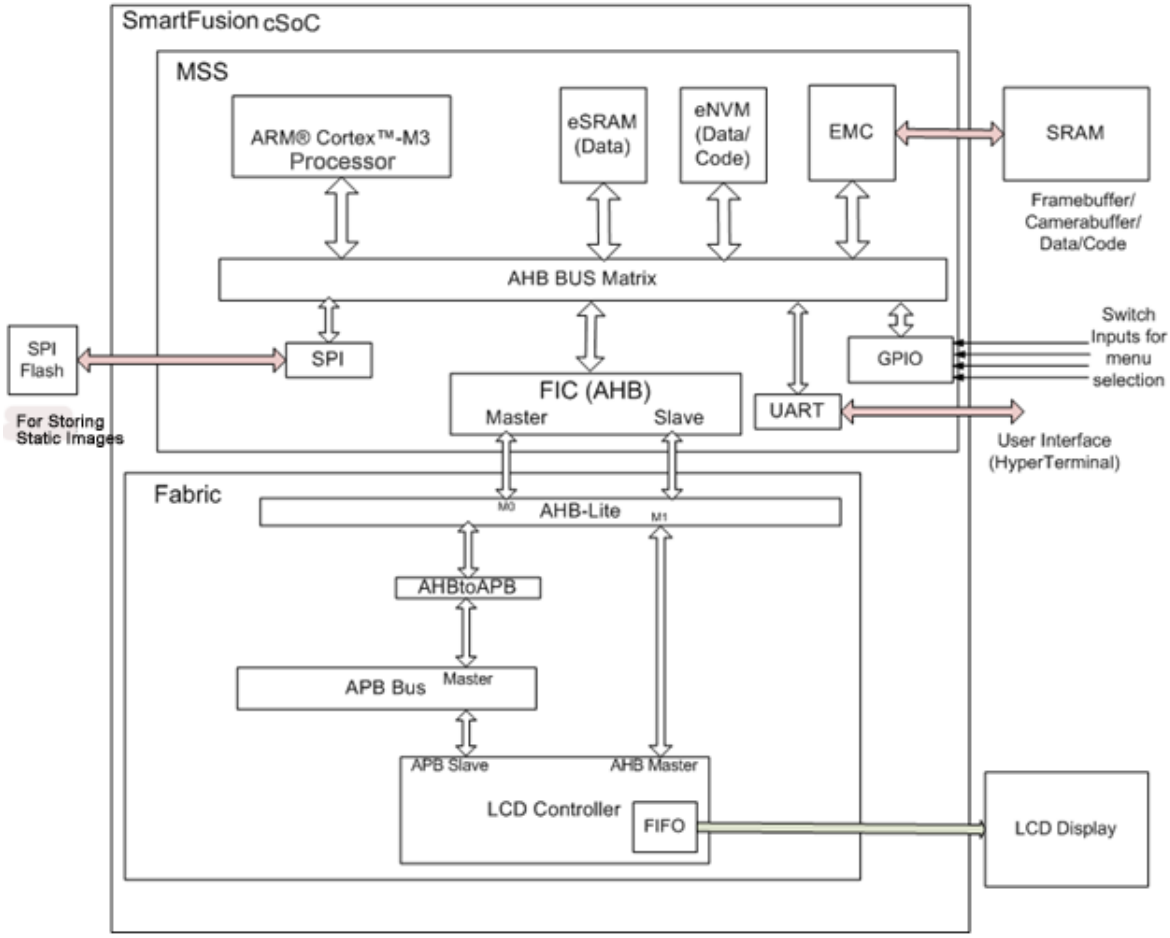

***Figure 1-1* • Display Solution Using the SmartFusion cSoC**

Hardware Implementation Details

The LCD controller implemented in the FPGA fabric supports both the 7 inch WVGA Toshiba TFT LCD module and 5.5 inch QVGA NEC TFT LCD module by setting appropriate generic parameters in the top level.

The LCD controller's top level module integrates the timing generation module, frame buffer interface, and a FIFO. The LCD controller in the fabric acts as an advanced high performance bus lite (AHB-Lite) master for fetching the pixel information from the framebuffer residing in external SRAM.

The Advanced Peripheral Bus (APB) slave interface available on the LCD controller is used to configure it using Cortex-M3 processor.

The LCD controller's top level module integrates the timing generation module, frame buffer interface,and a FIFO. The LCD controller in the fabric acts as an advanced high performance bus lite (AHB-Lite) master for fetching the pixel information from the framebuffer residing in external SRAM. The pixel data read from the framebuffer is stored in a FIFO buffer.

Pixel data read from the framebuffer by the LCD controller may not arrive in timely manner. These delays can occur due to latency or bus contention issues experienced by the LCD controller while accessing the frame buffer memory. The FIFO memory helps prevent a data underflow condition by acting as buffer. The timing module generates necessary timing information, pixel clock, and pixel information in RGB565 format to the attached LCD display panel.

Software Implementation Details

The OpenGL SC profile graphics Application Program Interface (API) implementation from Vincent 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 SRAM. The LCD controller implemented within the FPGA fabric reads from the external RAM and sends it to the LCD module. Graphics rendering is done through dual frame buffers with the ping-pong mode of operation and hence there will not be artifacts on the LCD display for frame changes in the display.

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 the functionality from OpenGL ES to minimize implementation and safety critical costs. It also adds functionality such as display lists.

Figure 1-2 depicts the OpenGL pipeline for graphics rendering:

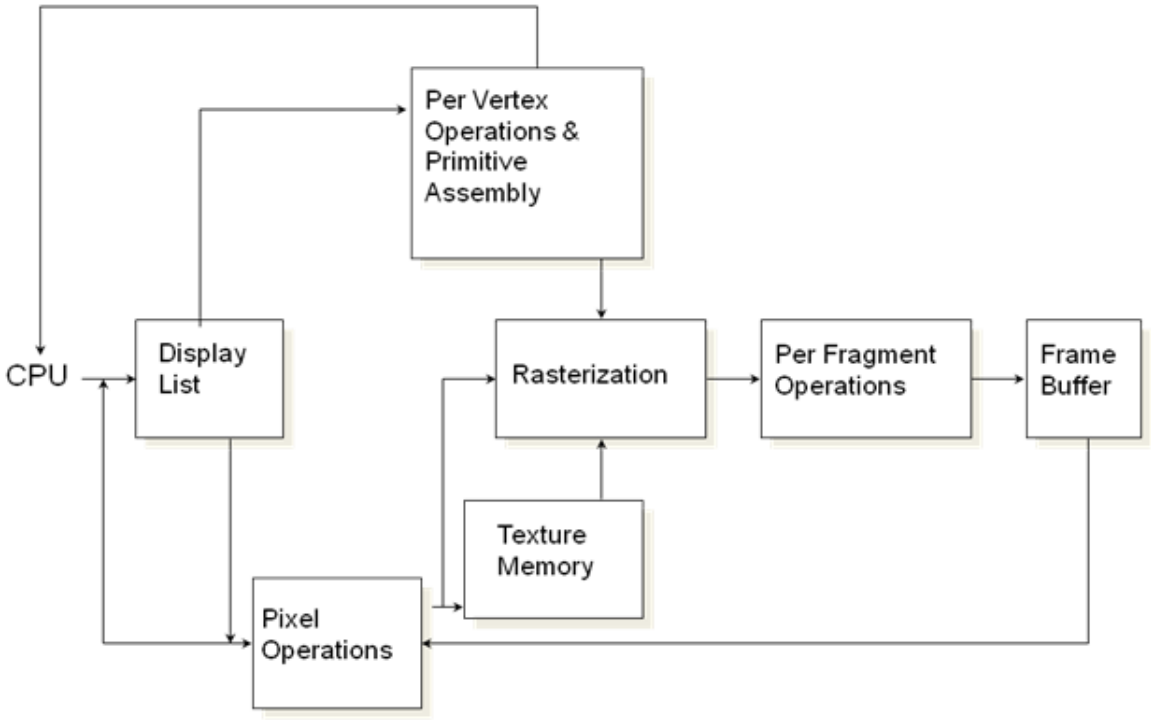

Figure 1-2 • **OpenGL Graphics Library Pipeline**

Figure 1-2 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/flows together.

The OpenGL SC profile ported on the SmartFusion cSoC 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 API for the graphics rendering are defined for this profiles. For more details on this API, please 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.

2 – API Usage Description

TFT Soft Controller Driver

Data Structures

TFT Instance Structure

This is a C data structure, struct TFT_instance_t, defined in the TFT Controller implementation. There should be one instance of this structure for each instance of TFTahbM in your system. TFT_init routine initializes this structure. It is used to identify the various TFTahbMs in your system. An initilized TFTahbM instance's structure should be passed as first parameter to TFTahbM functions to identify which TFTahbM should perform the requested operation. Software using this driver should only be used to create one single instance of this data structure for each GPIO instance in the system.

```
typedef struct tft_instance
{
addr_t address;
uint32_t VideoBufferStartAddress;
uint32_t VideoRdBufferStartAddress;
uint32_t VideoWrBufferStartAddress;
uint32_t VideoBufferSize;
uint32_t color_value;
uint32_t num_pixels;
uint32_t num_lines;
} TFT_instance_t;
```

Soft Core LCD Driver User APIs

Function: TFT_init

API:

TFT_init (TFT_instance_t * this_tft,

addr_t base_address,

uint32_t VideoBufferStartAddress,

uint32_t VideoBufferSize,

uint32_t color_value,

uint32_t num_pixels,

uint32_t num_lines)

Description: The TFT_init() function initializes a CoreTFTahbM hardware instance and the data structure associated with the CoreTFTahbM hardware instance.

Parameters:

1. **this_tft**: Pointer to the TFT_instance_t data structure instance holding all data regarding the CoreTFTahbM hardware instance is initialized. A pointer to the same data structure will be used in subsequent calls to the CoreTFTahbM driver functions in order to identify the CoreTFTahbM instance that should perform the operation implemented by the called driver function.
2. **base_addr**: The base_addr parameter is the base address in the processor's memory map for the registers of the CoreTFTahbM instance being initialized.
3. **VideoBufferStartAddress**: The VideoBufferStartAddress parameter informs the driver of the video buffer start address location in memory.
4. **color_value**: The color_value parameter informs the driver of the default color value to be used when the video buffer access is disabled.

5. **num_pixels**: The num_pixels parameter informs the driver, the number of pixels per line.This is specific to the TFT used. In the design files, the definition for symbol SCREEN_HOR_SIZE represents the num_pixels.
6. **num_lines**: The num_lines parameter informs the driver about the number of lines in the TFT. This is specific to the TFT used. In the design files, the definition for symbol SCREEN_VER_SIZE represents the num_lines.

Function: PowerOn_TFT

API: void PowerOn_TFT(TFT_instance_t * this_tft);

Description: This function performs the TFT specific power-on sequence.

Parameters: this_tft: Pointer to the TFT_instance_t data structure instance initialized for this port.

Function: PowerOff_TFT

API: void PowerOff_TFT(TFT_instance_t * this_tft);

Description: This function performs the TFT specific power-off sequence.

Parameters: this_tft: Pointer to the TFT_instance_t data structure instance initialized for this port.

Function: SetTFTDefaultColor

API: void SetTFTDefaultColor(TFT_instance_t * this_tft, uint32_t color_value)

Description: This function sets the TFT RGB color value used when video buffer is disabled.

Parameters:

- **this_tft**: Pointer to the TFT_instance_t data structure instance initialized for this port.
- RGB color value.

Function: SetTFTRdBuffer

API: void SetTFTRdBuffer(TFT_instance_t * this_tft, uint32_t ptr_videobuff, uint32_t videobuff_size)

Description: This function sets the TFT Video Read Buffer Start Address and End Address for this TFT instance.

Parameters:

- **this_tft**: Pointer to the TFT_instance_t data structure instance initialized for this port.
- **ptr_videobuff**: Video buffer address pointer.
- **videobuff_size**: Video buffer size.

Function: SetTFTWrBufferPage

API: uint32_t SetTFTWrBufferPage(TFT_instance_t * this_tft,uint8_t videobuff_page);

Description: This function sets TFT Video Write Buffer Start Address and End Address for this TFT instance.

Parameters:

- **this_tft**: Pointer to the TFT_instance_t data structure instance initialized for this port.
- videobuff_page: Video write buffer page.

Function: SetTFTRdBufferPage

API: uint32_t SetTFTRdBufferPage(TFT_instance_t * this_tft,uint8_t videobuff_page);

Description: This function sets the TFT Video Read Buffer Start Address and End Address for this TFT instance.

Parameters:

- **this_tft**: Pointer to the TFT_instance_t data structure instance initialized for this port.
- videobuff_page: Video write buffer page.

Function: GetTFTRdBuffer

API: void GetTFTRdBuffer(TFT_instance_t * this_tft)

Description: Get TFT Video Read Buffer Start Address.

Parameters:

this_tft: Pointer to the TFT_instance_t data structure instance initialized for this port.

Graphics APIs for Surface and Windows Creations

Creating a surface

Function: vglCreateSurface

API: glSurface vglCreateSurface(width,height,GL_RGB,GL_UNSIGNED_SHORT_5_6_5,0);

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.

Parameters:

- The first and second parameters for this API are the width and height of the LCD resolution. In our demo for the 5 inch LCD these values are 320, 240 and for 7 inch LCD these values are 800 and 480 respectively.
- The third parameter is the pixel information for the 16 BPP it is GL_GRB and for the 24 bit it is GL_RGBA.
- The forth parameter is the RGB information. This demo supports the RGB 565 format and RGBA (24 bit) format.

Enabling the current surface

Function: vglMakeCurrent;

API: vglMakeCurrent(glSurface, glSurface);

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.

Parameters:

- The first parameter for this API is the pointer to the draw surface.
- Second parameter for this API is the pointer to the read surface.

Setting the viewport/window

Function: glViewport;

API: glViewport(0,0,SCREEN_HOR_SIZE,SCREEN_VER_SIZE);

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

Parameters: glViewport(x,y,w,h);

- The x and y are window coordinates of the viewport's lower left corner.
- w and h give the viewport's width and height.

OpenGL SC Graphics APIs Used in this Demo

Following are the OpenGL SC APIs used/verified in this port; refer to "Running the Design" section on page 17 and "Building the SoftConsole Project" section on page 35 for the demo design files links on how to run and configure the demo for 7 inch or 5 inch LCD:

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 details on the above mentioned API and their usage, please refer to the following websites:

http://www.khronos.org/openglsc

Following are some useful links related to the OpenGL & 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

3 – Memory Requirements for this OpenGL SC Application on SmartFusion cSoC

This reference design is the port of the OpenGL SC profile on the SmartFusion cSoC device. The minimum memory requirements for this port for the graphics application are as follows:

- RAM of ~44 KB for the stack, heap, and data sections.
- ROM of ~213 KB for the code and constant data sections.
- For Frame buffers:
  - 5.5 inch LCD: 320x240 (QVGA) with 16 bpp requires 307.2 KB of memory for the dual frame buffer.
  - 7 inch LCD: 800x480 (VGA) with 16 bpp requires 1.5 MB of memory for the dual frame buffer.

The SmartFusion Development Kit Board has 4 MB of external asynchronous SRAM connected through the EMC which can be used for the frame buffer. The CODE and DATA memory size requirements may vary according to the application memory requirements.

4 – Running the Design

Directory Structure of the Demo Files

Download the design files from the Microsemi SoC Products Group website:
www.microsemi.com/soc/download/rsc/?f=SmartFusionOpenGLSC_Demo

Figure 4-1 displays the directory structure of the demo files.

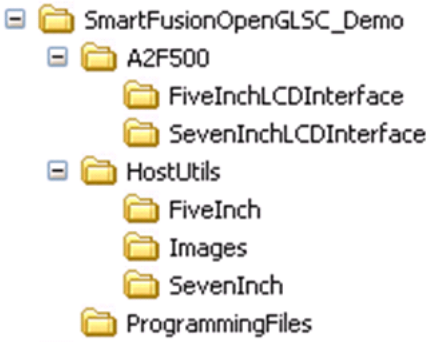

***Figure 4-1* • Directory Structure**

- HostUtils: This folder contains the images and host flash loader tools for both 5 inch and 7 inch LCD demos.
- A2F500: This folder contains the hardware Libero® System-on-Chip (SoC) software project files for the 5 inch and 7 inch LCD interfaces and also the softconsole project for both 5 inch and 7 inch LCD interfaces..
- ProgrammingFiles: This folder contains the STAPL files for both 5 inch and 7 inch LCD demos.

Hardware Required for the Demo Setup

- SmartFusion Development Kit Board (A2F500-DEV-KIT). We need to use the A2F500 Dev Kits for the exteranl RAM used for framebuffers and FPGA design size requirements.
- 5.5 inch LCD adaptor board or 7 inch LCD adapter board
- Mixed signal display adapter connector
- 12 V power supply adapter for LCD adaptor board
- Power adapter and 2 USB cables for the SmartFusion Development Kit Board

Steps to Run the Demo

1. Connect the 7 inch LCD adapter board with LCD (IGLOO-LCD-Adapter-LTA070A320F) or 5.5 inch LCD adapter board QVGA NEC TFT LCD to the mixed signal header SmartFusion Development Kit Board (Rev F, with SmartFusion 500 Device).
2. Connect all necessary cables such as USB and power (for both LCD and development board).
3. Make sure the jumper settings required for SPI flash and EMC are correct.

Figure 4-1 the jumper settings.

Table 4-1 • **Jumper Settings**

Jumper	Pin	Pin
JP17	2	3
JP19	2	3
JP24	1	2
JP16	2	3

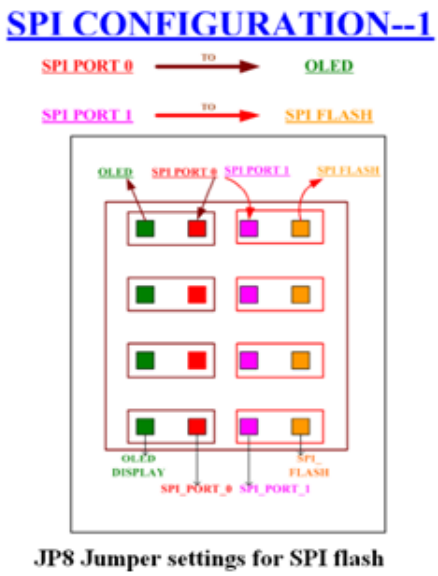

Figure 4-2 • **JP8 Jumper Settings For SPI Flash**

4. Power on. Make sure both Development board and LCD daughter card are powered on
5. There is a STAPL file available for the Smart Fusion OpenGL demo under the folder **SmartFusionOpenGLSC_Demo -> programming file.**

- Top_7TFT16bit_WVGA_EthMAC.stp for 7 inch LCD

  or

- Top_55TFT16bit_QVGA_EthMAC.stp for 5.5 inch LCD

6. Open standalone Flash pro3 application from the Libero installation folder.
7. Create a project.
8. Program the Top_70TFT16bit_WVGA_EthMAC.stp for 7 inch LCD or Top_55TFT16bit_QVGA_EthMAC.stp for 5.5 inch LCD on to the SmartFusion Development Kit Board.
9. Plug in the UART cable between the host PC and board.
10. From the Windows Start menu, select Programs > Accessories > Communications > HyperTerminal. This opens HyperTerminal. If your computer does not have HyperTerminal, use any free serial terminal emulation program like PuTTY or Tera Term. Refer to the Configuring Serial Terminal Emulation Programs tutorial for configuring the HyperTerminal, Tera Term, and PuTTY.
11. Set Up HyperTerminal with the following setting:
    – Baud rate is set to 57600
    – 8 bits data
    – 1 stop bit
    – No parity and no flow control
12. Connect HyperTerminal.
13. Perform power cycle the board.
14. Text will be displayed in the HyperTerminal window, as shown in Figure 4-3.

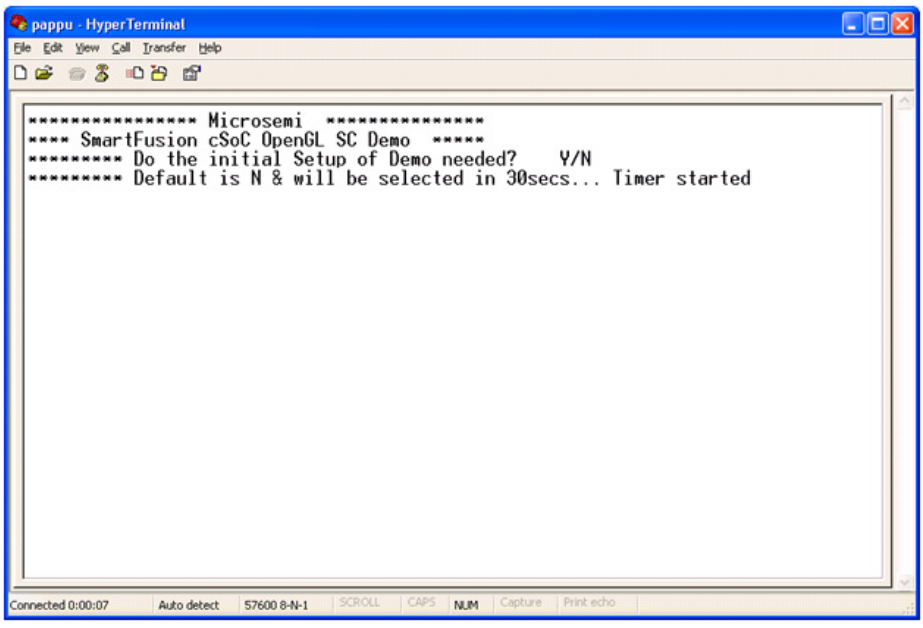

***Figure 4-3* • Demo 1**

15. First, type Y in the HyperTerminal window, then text will be displayed, on the HyperTerminal window as shown in Figure 4-4.

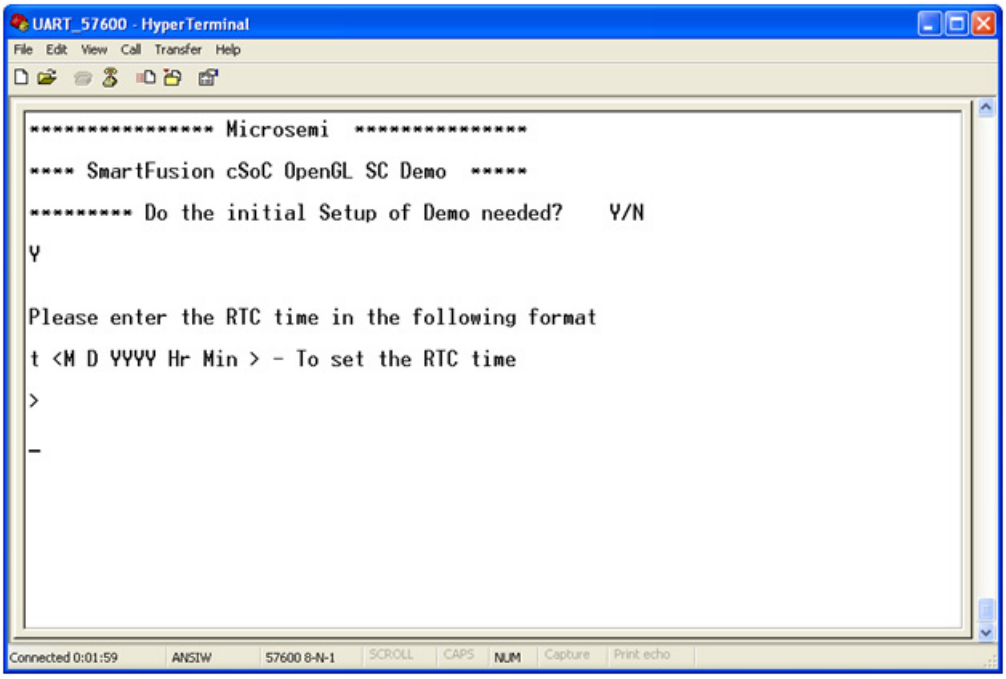

***Figure 4-4* • Demo 2**

16. Type the date as described for RTC configuration (Example: t 10 18 2011 19 40). The text will be displayed, as shown in Figure 4-5. (If you enter the date incorrectly, you will be required to start again from step 12)

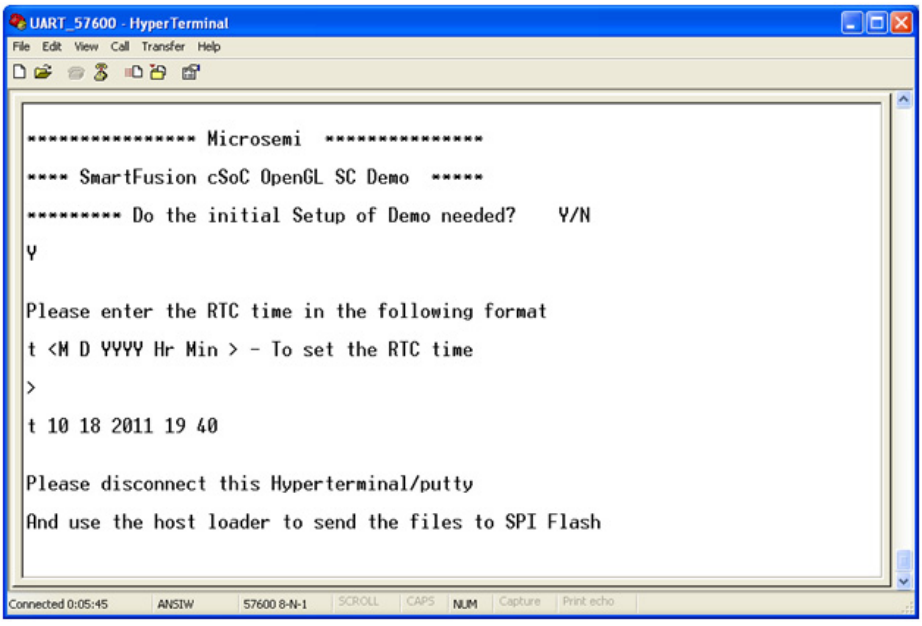

***Figure 4-5* • Demo 3**

17. Disconnect HyperTerminal.
18. Navigate to the directory..\SmartFusionOpenGLSC_Demo\HostUtils, open the command prompt window at this location, and run the batch file indicated below:
    - For 5 inch LCD: loadImages_5inch.bat COMPORT_NUMBER
    - For 7 inch LCD: loadImages_7inch.bat COMPORT_NUMBER

Example: if the com port number is 6, use the batch files as follows:

- For 5 inch LCD: loadImages_5inch.bat 6
- For 7 inch LCD: loadImages_7inch.bat 6

If you are using the same kit for both the 5 inch and 7 inch demos, you need to format the SPI flash before writing the new images, you have only 8 MB SPI flash, which will not be able to accommodate the images for both the resolutions. In this case, use the command **fm** in step 15 instead of time input; this command will format the SPI flash.

19. The following text will be displayed in the command prompt window:

```
Serial port COMPORT_NUMBER successfully reconfigured.

Please make sure smart fusion dev kit is running
target loader before launching host loader

Handshaking with Smartfusion F2 uart
HAND SHAKE is fine
ReadBack Number Of Files =0xa

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_1_5inch.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_2_5inch.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_3_5inch.bin

.....Ori File name length = 17.....
```

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_4_5inch.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_5_5inch.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_6_5inch.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_7_5inch.bin

.....Ori File name length = 15.....

.....File name length .....
read back file name length =15

.....Sent File Name .....
read back filename =logo_187x35.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =menu1_5_5inch.bin
.....Ori File name length = 22.....

```
.....File name length .....
read back file name length =22

.....Sent File Name .....
read back filename =sel_switch_5_5inch.bin
size = 81920

.....Erasing the Flash .....
read back size =81920
Wrting Data
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Bytes written =81920
.....Flash Programming is Successful.....
size = 81920

.....Erasing the Flash .....
read back size =81920
Wrting Data
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Bytes written =81920
.....Flash Programming is Successful.....
size = 81920

.....Erasing the Flash .....
read back size =81920
Wrting Data
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Bytes written =81920
.....Flash Programming is Successful.....
size = 81920

.....Erasing the Flash .....
read back size =81920
Wrting Data
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
Bytes written =81920
.....Flash Programming is Successful.....
size = 81920

.....Erasing the Flash .....
read back size =81920
Wrting Data
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Bytes written =81920
.....Flash Programming is Successful.....
size = 81920

.....Erasing the Flash .....
read back size =81920
Wrting Data
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Bytes written =81920
.....Flash Programming is Successful.....
size = 81920

.....Erasing the Flash .....
read back size =81920
Wrting Data
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Bytes written =81920
.....Flash Programming is Successful.....
size = 13090

.....Erasing the Flash .....
read back size =13090
Wrting Data
---------------------------------------------------Bytes written =13090
.....Flash Programming is Successful.....
size = 10240

.....Erasing the Flash .....
```

read back size =10240
Wrting Data
----------Bytes written =10240
.....Flash Programming is Successful.....
size = 10240

.....Erasing the Flash .....
read back size =10240
Wrting Data
----------Bytes written =10240
.....Flash Programming is Successful.....

For 7 inch LCD:

Serial port COM6 successfully reconfigured.

Please make sure smart fusion dev kit is running
target loader before launching host loader

Handshaking with Smartfusion F2 uart
HAND SHAKE is fine
ReadBack Number Of Files =0xa

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_1_7inch.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_2_7inch.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_3_7inch.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_4_7inch.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_5_7inch.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_6_7inch.bin

.....Ori File name length = 17.....

.....File name length .....
read back file name length =17

.....Sent File Name .....
read back filename =image_7_7inch.bin

.....Ori File name length = 15.....

.....File name length .....
read back file name length =15

.....Sent File Name .....
read back filename =logo_256x48.bin

.....Ori File name length = 15.....

.....File name length .....
read back file name length =15

.....Sent File Name .....

read back filename =menu1_7inch.bin

.....Ori File name length = 20.....

.....File name length .....
read back file name length =20

.....Sent File Name .....
read back filename =sel_switch_7inch.bin
size = 368640

.....Erasing the Flash .....
read back size =368640
Wrting Data

Bytes written =368640
.....Flash Programming is Successful.....
size = 368640

.....Erasing the Flash .....
read back size =368640
Wrting Data


```
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
Bytes written =368640
.....Flash Programming is Successful.....
size = 368640

.....Erasing the Flash .....
read back size =368640
Wrting Data
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
Bytes written =368640
.....Flash Programming is Successful.....
size = 368640

.....Erasing the Flash .....
read back size =368640
Wrting Data
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
```


Bytes written =368640
.....Flash Programming is Successful.....
size = 368640

.....Erasing the Flash .....
read back size =368640
Wrting Data

Bytes written =368640
.....Flash Programming is Successful.....
size = 368640

.....Erasing the Flash .....
read back size =368640

```
Wrting Data
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
Bytes written =368640
.....Flash Programming is Successful.....
size = 368640

.....Erasing the Flash .....
read back size =368640
Wrting Data
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
Bytes written =368640
.....Flash Programming is Successful.....
```

size = 24576

.....Erasing the Flash .....
read back size =24576
Wrting Data

----------Bytes written =24576
.....Flash Programming is Successful.....
size = 40960

.....Erasing the Flash .....
read back size =40960
Wrting Data

Bytes written =40960
.....Flash Programming is Successful.....
size = 40960

.....Erasing the Flash .....
read back size =40960
Wrting Data

Bytes written =40960
.....Flash Programming is Successful.....

20. This process will take a couple of minutes as it is writing images to the SPI flash. When this process is complete, reconnect HyperTerminal and power cycle the board.
21. The HyperTerminal display will look similar to Figure 4-3 on page 19. Type N.

22. After a few seconds, the display will look similar to Figure 4-6.

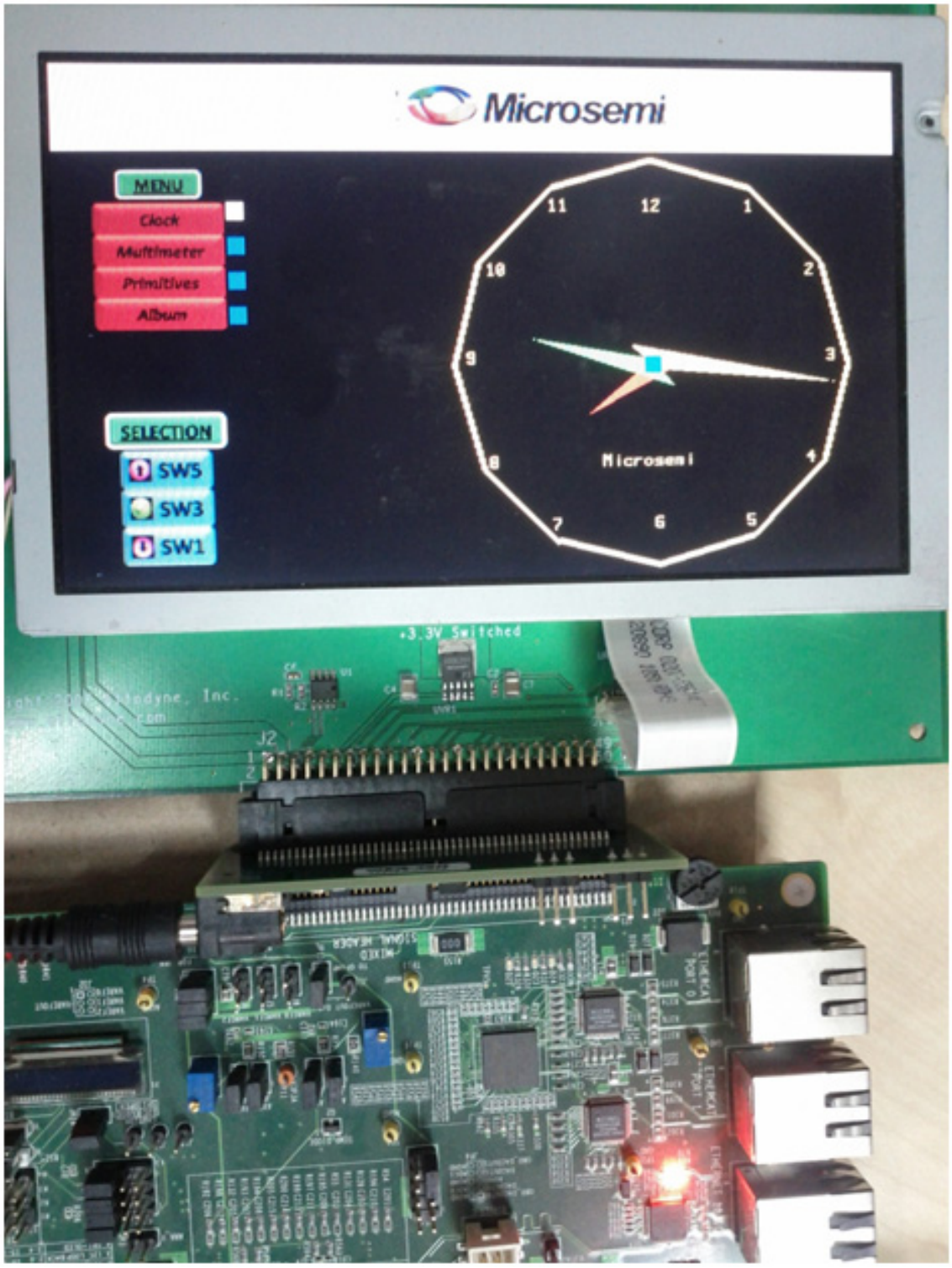

Figure 4-6 • **Menu**

23. The Clock option from the menu is active by default. To select menu options, navigate using SW5, SW3, and SW1, which are the switches on the development kit board. For example, to select the multimeter, press SW1 gently once to scroll and then press SW3 gently once to select.
24. The selected option will display a small white square next to the option. All others will have a blue square.
25. While navigating from one option to other, give sufficient time.
26. When the album option is selected, allow some time to load the pictures from SPI flash.

5 – Building the SoftConsole Project

The SoftConsole project is made for the both the 5 inch and 7 inch LCD demos. If any changes are needed, you must select the compiler option to choose the LCD and then recompile the project. The default option is set for the 7 inch LCD demo. If you want to change to the 5 inch LCD, set the symbol VGA_NQVGA to 0 in the project properties by using the edit option in the same window (Figure 5-1).

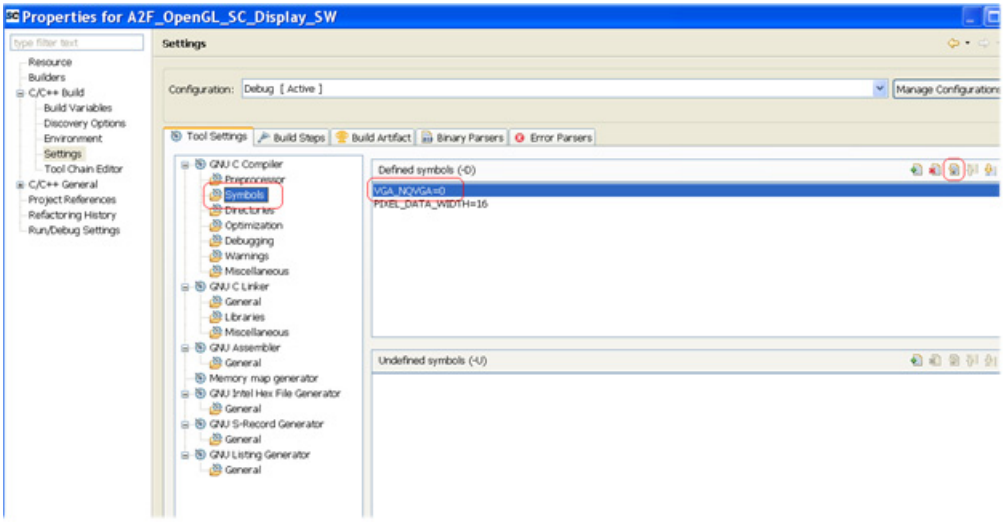

***Figure 5-1* • Properties Window**

A – List of Changes

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
Revision 1 (May 2012)	Modified "Running the Design" section (SAR 38468)	17
	Replaced Figure 4-1 (SAR 38468)	17
	Modified "Steps to Run the Demo" (SAR 38468)	18
	Replaced Figure 4-3 (SAR 38468)	19

*Note: *The part number is located on the last page of the document. The digits following the slash indicate the month and year of publication.*

B – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 650.318.8044

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

50200303-1/05.12