TFT User Manual

Sean Kent, Eric Ponce

August 2019

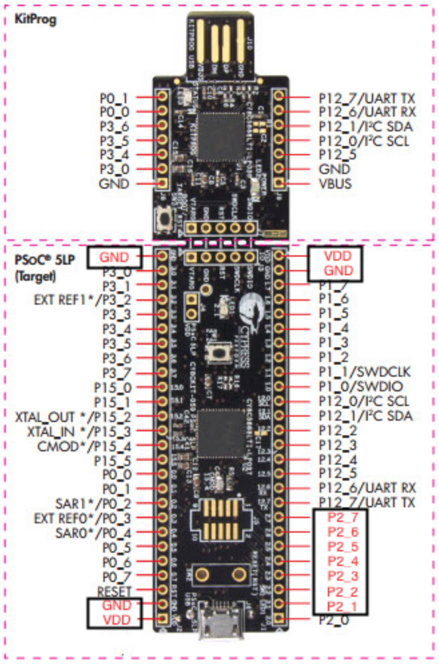
1 Overview

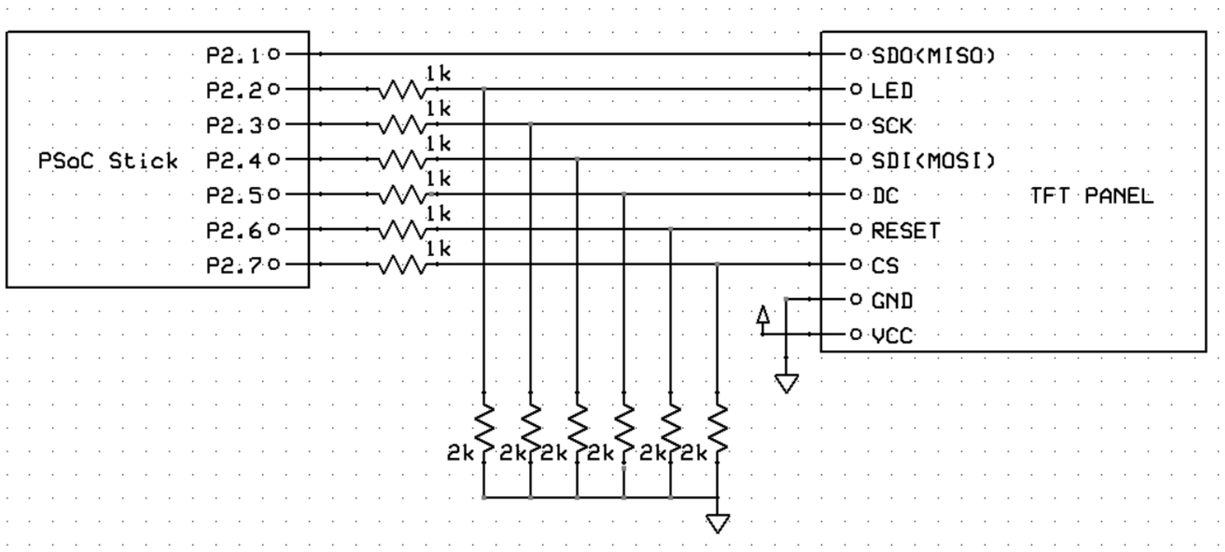
This tutorial shows how to configure the PSoC to communicate with an ILI9341 TFT display using 8-bit SPI (Serial Peripheral Interface). This tutorial also explains the basic steps for writing pixels to the display as well as how to include emWin, a graphic library which provides functions for drawing and displaying text on the LCD.

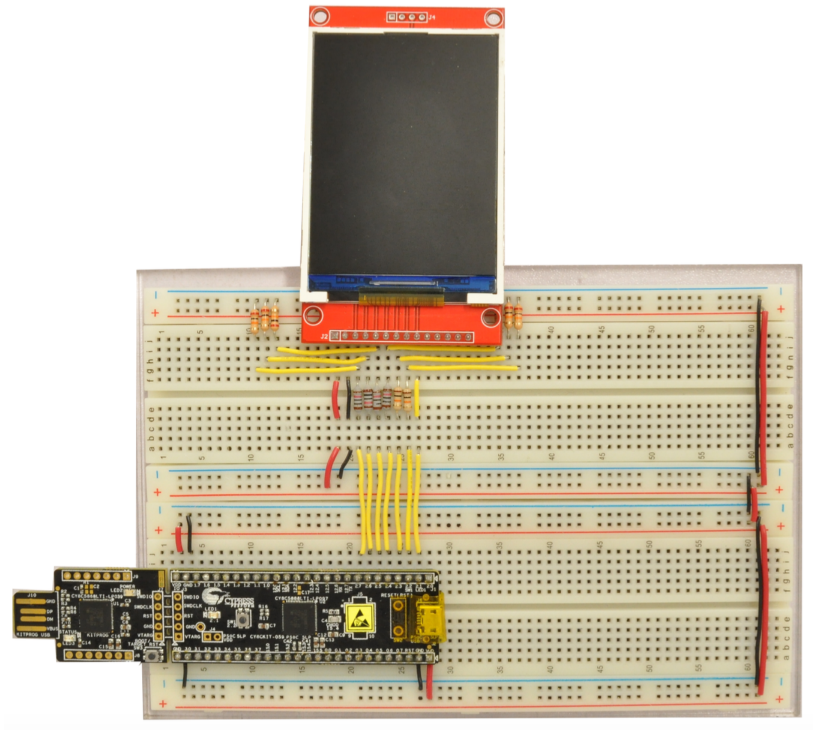
This tutorial is accompanied by two Creator projects, TFT and TFT_with_emWin, which can be found on the course website. We will start by looking at the TFT project, which is intended to get you comfortable communicating with the display.

2 Hardware

The pins colored red (also boxed) are used in this demo. Everything else can be left unconnected. The pin assingments can be found in the projects .cydwr file.







Although the TFT display module can be powered off of 5V, it uses 3.3V logic. If the PSoC is powered with 5V, all its outputs going to the TFT must be passed though a voltage divider to drop the 5V to 3.3V. Outputs of the TFT, such as SDI, will be at 3.3V and can be connected directly to an input pin on the PSoC.

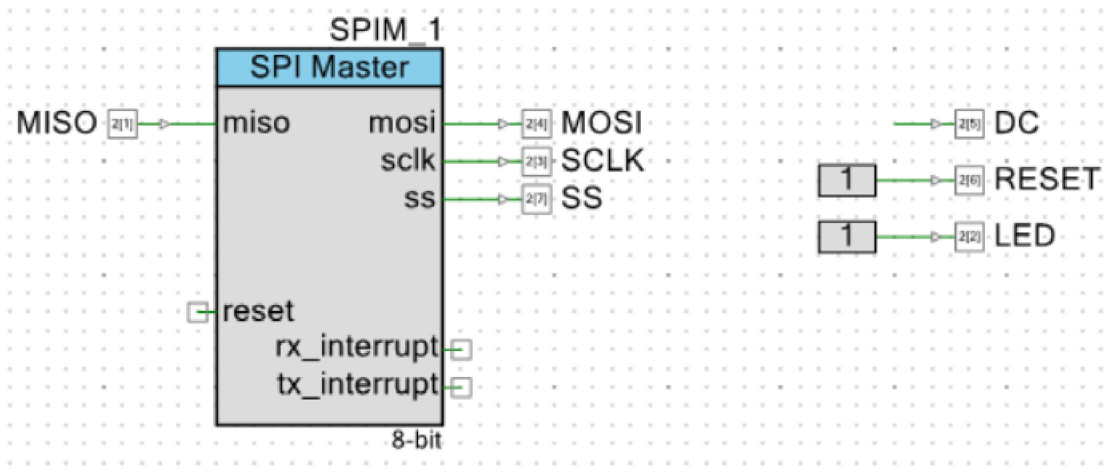
2.1 LCD Pin Descriptions

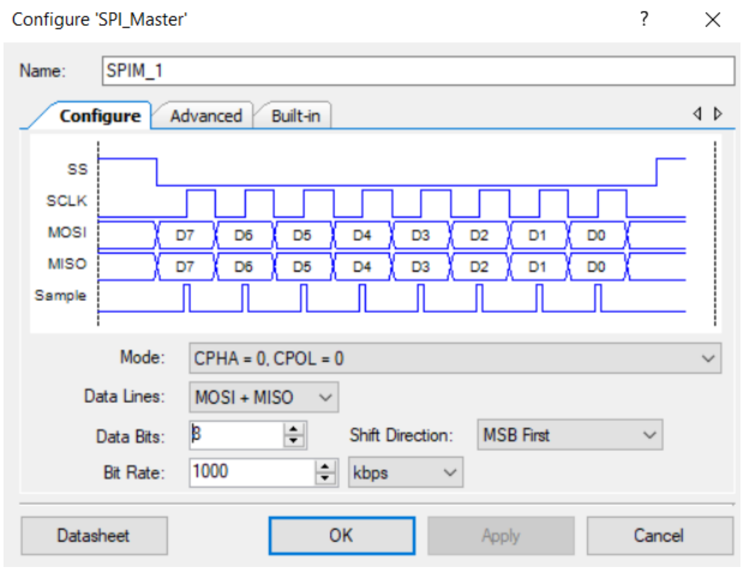
- VCC - 5V/3.3V power input

- GND - Ground
- CS - Chip select (active low)
- RESET - Reset (active low)
- DC - Data/Command select. When the DC line is low, data received by the LCD is interpreted as commands. When this DC line is high, data is interpreted as data (arguments to commands, pixel data, etc.)
- SDI - Serial data input (mosi)
- SDO - Serial data output (miso)
- SCK - Serial clock input
- LED - This pin is used to control the intensity of the backlight. If not controlled by an analog voltage, connecting this pin to 3.3V will set the display to full brightness.

2.2 Cypress Schematic

The PSoC communicates with the TFT using 8-bit SPI. To do this, we will use Creator's SPI Master (SPIM) component. The .cysch file and component configurations are shown below.




Configure 'SPI_Master'

Name: SPIM_1

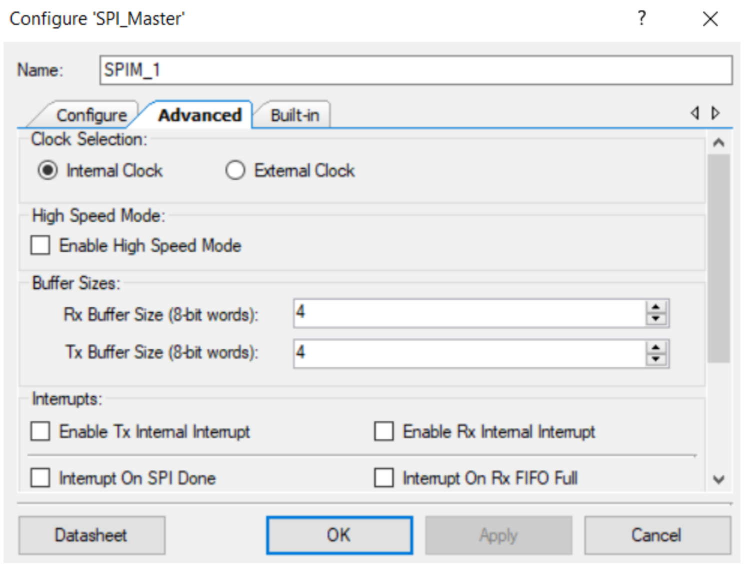
Mode: CPHA = 0, CPOL = 0

Data Lines: MOSI + MISO

Data Bits: 8

Shift Direction: MSB First

Bit Rate: 1000 kbps


Configure 'SPI_Master'

Name: SPIM_1

Clock Selection:

● Internal Clock ○ External Clock

High Speed Mode:

☐ Enable High Speed Mode

Buffer Sizes:

Rx Buffer Size (8-bit words): 4

Tx Buffer Size (8-bit words): 4

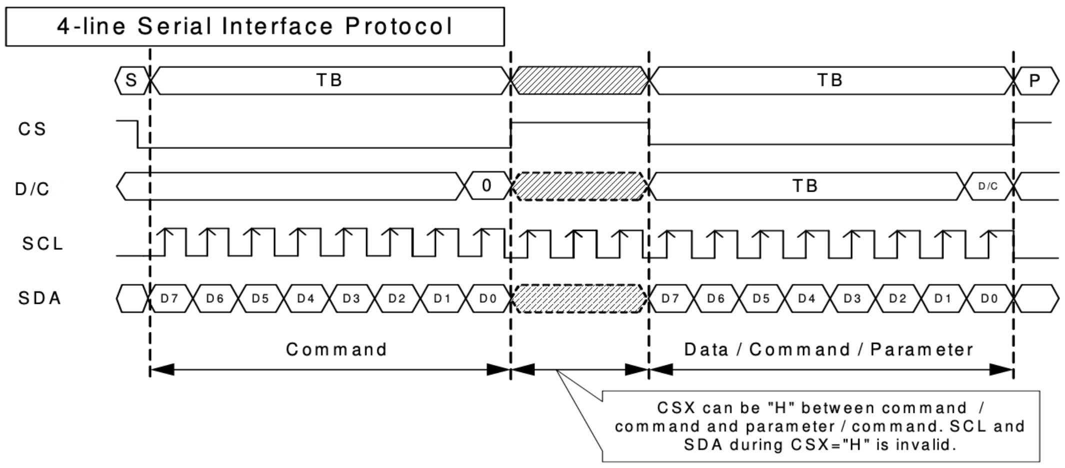
Interrupts:

☐ Enable Tx Internal Interrupt ☐ Enable Rx Internal Interrupt

☐ Interrupt On SPI Done ☐ Interrupt On Rx FIFO Full

Make sure the SPIM component is set up to send 8-bit data and that the shift direction is set to MSB first. The TFT can handle bit rates of up to around 3 Mbps.

In addition to the typical SPI connections, the TFT uses the DC line to specify whether the information being received should be interpreted as a command or as data. We will use the PSoC's output pin labeled "DC" to control the TFT's DC line. This pin must be set high or low in software depending on whether we want to send data or commands to the display. The other output pins, labeled "RESET" and "LED", should be tied to Vcc using the Logic High component. We will not use them in this tutorial. The diagram below shows the SPI write sequence.


4-line Serial Interface Protocol
S
TB
TB
P
CS
D/C
0
TB
D/C
SCL
SDA
D7
D6
D5
D4
D3
D2
D1
D0
Command
Data / Command / Parameter
CSX can be "H" between command / command and parameter / command. SCL and SDA during CSX="H" is invalid.

3 Firmware

3.1 Sending Commands/Data

If you navigate to the file tft.c, you will find two functions, `write8_a0()` and `write8_a1()`, which we will use to write commands and data to the TFT. `write8_a0()` sends an 8-bit value to the TFT (via the SPIM component) with the DC line set low. Data sent using this function are thus interpreted as commands. `write8_a1` sends 8-bit values with the DC line set high. Data sent using this function are interpreted as data.

Typically, communication with the TFT involves sending a command followed by zero or more parameters (sent to the TFT as data). A list of commands and their parameters can be found in the ILI9341 datasheet. For example, the command Column Address Set (0x2A) defines the range of frame memory columns the MCU can access. This command expects four 8-bit parameters, which specify the 16-bit addresses of the start column (SC) and end column (EC). The command and parameters would be sent as follows:

```
write8_a0(0x2A);        // send Column Address Set command (DC line low)
write8_a1(0x00);        // send 1st Parameter, SC[15:8] (DC line high)
write8_a1(0x0A);        // send 2st Parameter, SC[7:0]
write8_a1(0x00);        // send 3st Parameter, EC[15:8]
write8_a1(0x14);        // send 4st Parameter, EC[7:0]
```

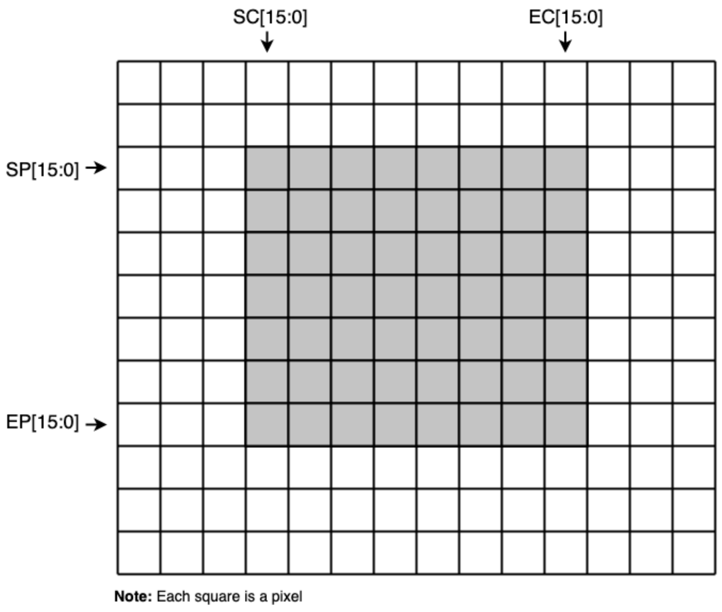
This block of code would set the start column to 10 and the end column to 20.

3.2 Initializing the Display

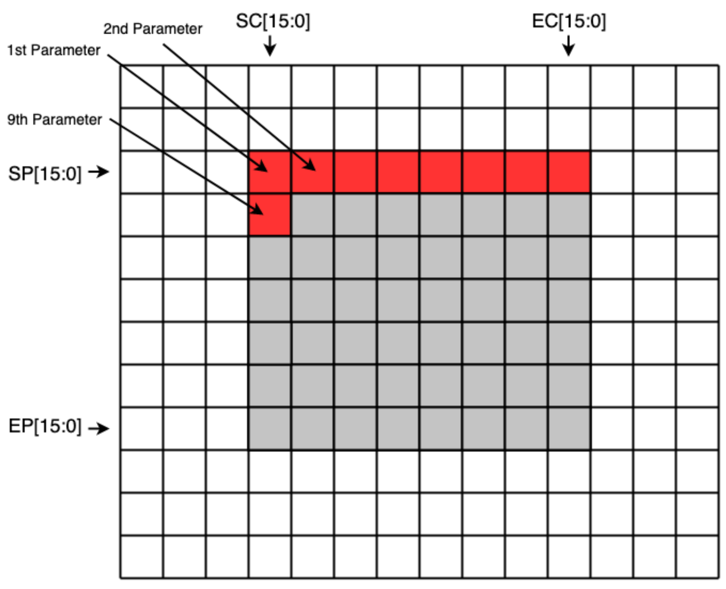
Before we can write pixels to the display, the initialization function `tftStart()` must be called. This function, which can also be found in tft.c, is responsible for setting important configuration parameters and turning the display on.

3.3 Writing Pixels

The process of writing pixels to the the TFT begins with defining a rectangular window of frame memory where you would like to write pixels. This is done using the Column Address Set (0x2A) and Page Address Set (0x2B) commands, which are responsible for setting the start column (SC), end column (EC), start page (SP), and end page (EP).



Once this window has been set, you can begin filling in pixels using the Memory Write (0x2C) command. This command takes an arbitrary number of 16-bit parameters, with each parameter specifying the color of a pixel (16-bit parameters should be sent MSB first). The Memory Write command begins filling in pixels starting in the top left corner. This means the first parameter will specify the color of the top left pixel. Successive parameters will fill in pixels moving left to right. When the end column is reached, the next row will be filled starting at the start column position. Once the Memory Write command has been sent, all successive data received by the driver will be interpreted as pixel data until another command is sent (i.e. a byte is sent with the DC line low).



Note: Each square is a pixel

The following is pseudo code outlining the typical pixel writing sequence described above:

```
write8_a0(0x2A);          // send Column Address Set command (DC line low)
write8_a1(SC[15:8]);      // set SC[15:0]
write8_a1(SC[7:0]);
write8_a1(EC[15:8]);      // set EC[15:0]
write8_a1(EC[7:0]);
write8_a0(0x2B);          // send Page Address Set command
write8_a1(SP[15:8]);      // set SP[15:0]
write8_a1(SP[7:0]);
write8_a1(EP[15:8]);      // set EP[15:0]
write8_a1(EP[7:0]);
write8_a0(0x2A);          // send Memory Write command
write8_a1(0xXX);          // send 1st Parameter (MSB first)
write8_a1(0xXX);
write8_a1(0xXX);          // send 2nd Parameter
write8_a1(0xXX);
.
.
.
write8_a1(0xXX);          // send Nth Parameter
write8_a1(0xXX);
write8_a0(0x00);          // send NOP command to end writing process
```

4 Adding emWin Graphic Library

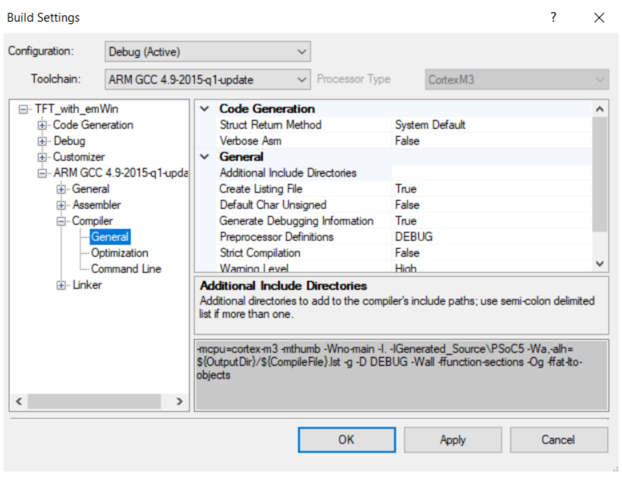
This section describes how to add the emWin library to a PSoC project. emWin is an embedded graphic library which has been licensed to Cypress by SEGGER and provides a wide range of functions for drawing and displaying text on an TFT. The following steps walk through how to add emWin to a project.

1. From the course website, download the PSoC project TFT_with_emWin. The emWin library is designed to support a large number of LCD/TFT controllers, including the ILI9341. However, the user must provide the appropriate hardware/software to communicate with display, as well as an initialization function. In addition to tft.c (which contains our functions for reading and writing to the display and our initialization function), you will find several other configuration files which have been provided by Cypress to set up the emWin library. The necessary changes have already been made to these files to facilitate communication with our TFT.

2. Download (and unzip) the emWin Graphic Library here[1]

3. Click on **Project** → **Build Settings**

   - In the left hand menu, navigate to **ARM GCC 4.9-2015-q1-update** → **Compiler** → **General** and click on **Additional Include Directories**. Add the following files:
     - ..\emWinGraphics_v5_46\Code\Include\PSoC4_5\nosnts
     - ..\emWinGraphics_v5_46\Code\Include\PSoC4_5



Build Settings

Configuration: Debug (Active)

Toolchain: ARM GCC 4.9-2015-q1-update

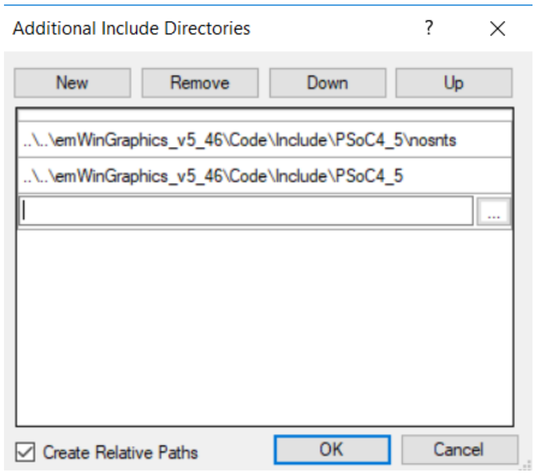
Processor Type: CortexM3

Code Generation	
Struct Return Method	System Default
Verbose Asm	False
General	
Additional Include Directories	
Create Listing File	True
Default Char Unsigned	False
Generate Debugging Information	True
Preprocessor Definitions	DEBUG
Strict Compilation	False
Warning Level	High

Additional Include Directories
Additional directories to add to the compiler's include paths; use semi-colon delimited list if more than one.

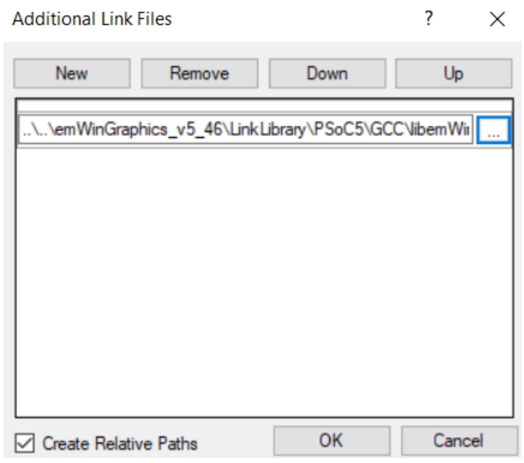
```
-mcpu=cortex-m3 -mthumb -Wno-main -I. -IGenerated_Source\PSoC5 -Wa,-alh=
${OutputDir}/${CompileFile}.lst -g -D DEBUG -Wall -ffunction-sections -Og -ffat-lto-
objects
```

[1] http://www.cypress.com/go/comp_emWin



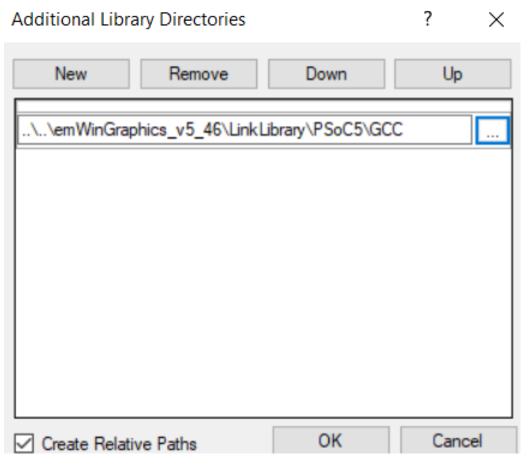
```
..\..\emWinGraphics_v5_46\Code\Include\PSoC4_5\nosnts
..\..\emWinGraphics_v5_46\Code\Include\PSoC4_5
```

- Next, navigate to **ARM GCC 4.9-2015-q1-update** → **Linker** → **General** and click on **Additional Link Files**. Add the following file:
  - ..\emWinGraphics_v5_46\LinkLibrary\PSoC5\GCC\libemWin_nosnts_cm3_gcc.a



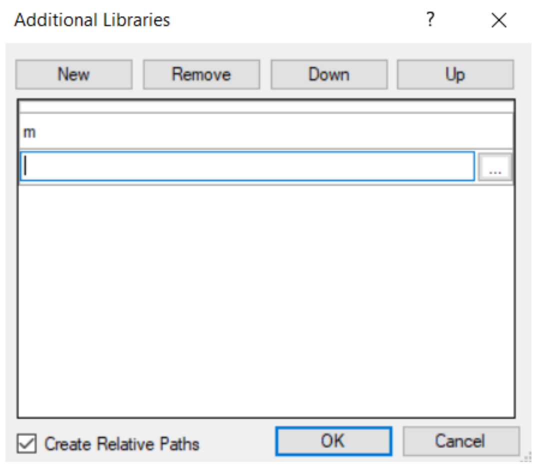
```
..\..\emWinGraphics_v5_46\LinkLibrary\PSoC5\GCC\libemWi
```

- Click on **Additional Library Directories** and add:
  - ..\emWinGraphics_v5_46\LinkLibrary\PSoC5\GCC



```
..\..\emWinGraphics_v5_46\LinkLibrary\PSoC5\GCC
```

- Click on **Additional Libraries** and add **m** (the math.h library)



m

4. The emWin should now be added to your project. In main.c, make sure "GUI.h" is included. The file should look something like this:

```
#include <project.h>
#include "GUI.h"

void MainTask(void);

int main()
{
    CyGlobalIntEnable;
    SPIM_1_Start();
    MainTask();
    for(;;){}
}

void MainTask(void)
{
    GUI_Init();     // must be called before any other GUI function
    .
    .
    .
}
```

It is typical to use the function `MainTask()` as the entry point for the emWin functions. Also note that `GUI_Init()` must be called before any other GUI functions can be used.

5. Program the PSoC with the TFT_with_emWin project. The code should fill the screen black and then print "6.115 Rocks!".

A complete description of the emWin setup process can be found here here[2]. The emWin user manual can be found here [3].

4.1 Helpful Links

- ILI9341 datasheet - https://cdn-shop.adafruit.com/datasheets/ILI9341.pdf
- Adding emWin to PSoC project - https://www.cypress.com/documentation/component-datasheets/segger-emwin-graphic-library-emwingraphics
- emWin User Manual - https://www.segger.com/downloads/emwin/UM03001
- 2.8inch SPI Module - http://www.lcdwiki.com/2.8inch_SPI_Module_ILI9341_SKU:MSP2807

[2]https://www.cypress.com/file/440296/download
[3]https://www.segger.com/downloads/emwin/UM03001