FOCUS LCDs
LCDs MADE SIMPLE®
Ph. 480-503-4295 | NOPP@FocusLCDs.com

TFT | OLED | GRAPHIC | CHARACTER | UWVD | SEGMENT | CUSTOM

Application Note FAN4223

Working with the E43RB1-FW405-C (Part 3)

This application note will present the firmware to drive the E43RB1-FW405-C TFT Display Module with an STM32H747I-DISCO microcontroller development board.

Contents

List of Figures

Working with the E43RB1-FW405-C MIPI Display (Part 2)

This series of application notes will discuss the hardware and software requirements of driving the E43RB1-FW405-C MIPI DSI TFT Display with an STM32H747I-DISCO microcontroller board from ST Microelectronics. Driving a MIPI DSI display with a microcontroller presents a few challenges. The STM32H747 has the required bandwidth and I/O pins but lacks enough internal SRAM for a full frame buffer. The DISCO board presented has an external SDRAM chip used for implementing the display frame buffer.

Previously, Part 1 (FAN4221) walked through the hardware requirements and interfacing the display while Part 2 (FAN4222) presented an overview of the firmware. Part 3 (FAN4223) will discuss the firmware modifications required for using a different Focus LCDs MIPI display, the E50RA-I-MW490-C.

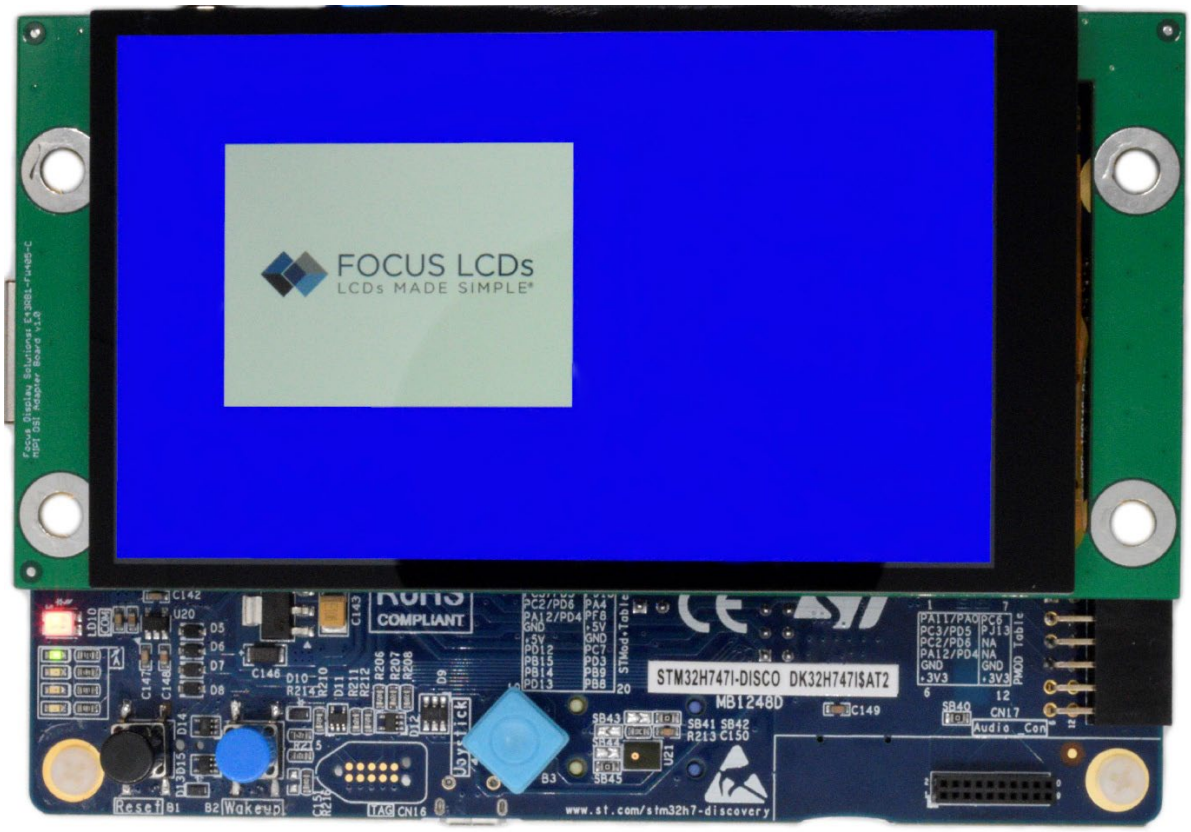

Figure 1: E43RB1-FW405-C, Adapter PCB, and STM32H7 Disco Running Demo

Introduction

This is the continuation of the application note series started in FAN4221 where the hardware aspects of this project were discussed. Part 2 focused on the demonstration application, project structure, and firmware overview. Part 3, presented here, will show how to modify the code to suit the Focus LCDs E50RA-I-MW490-C TFT MIPI Display.

It is expected that the E50RA-I-MW490-C will be connected to the Adapter Board and plugged into the STM32H747I-DISCO Development Board. Contact Focus LCDs for more information on the Adapter Board and firmware.

E50RA-I-MW490-C TFT MIPI Display

The E50RA-I-MW490-C display used in this application is a 5.0” TFT with a 480 x 854 RGB pixel resolution from Focus LCDs. This display is interfaced over a 2-lane MIPI DSI protocol with a 20-pin FPC cable. The display is connected to an adapter board and to the STM32H7I Discovery development board.

Figure 2: E50RA-I-MW490-C TFT MIPI Display

The main features of the E50RA-I-MW490-C are:

- 5.0-inch diagonal display, 480 x 854 RGB pixel resolution
- Up to 65K/262K/16.7M (24-bit) colors
- 2-Lane MIPI DSI interface with 20-pin FPC cable
- Transmissive/Normally Black display mode
- White LED Backlight
- ILI9806E Display Controller
- Capacitive Touch Interface (GT911) – Touch Modes: 5-Point and Gestures
- Typical Operating Voltage 3.3V

Adapting the Firmware for the E50RA-I-MW490-C

Adapting the E43RB1-FW405-C firmware for the E50RA-I-MW490-C display will be presented. This guide will go through the necessary steps of the configuration for the E50RA display. Changes to the code will be shown in the STM32CubeIDE and in a plain text editor.

These are the basic steps to adapting the firmware:

1. Copy the existing .c and .h files and change the names of the display specific files to the new display name.
2. Edit the header file #ifndef to match the new file name.
3. Edit the .c file with the new header file name.
4. Check the LCD controller chip and adjust the .c files as necessary.
5. Continue to edit the .c file with the new display parameters.
6. Make the initialization sequence changes of the .c file in **`runInitSeqLCDConfig()`** function.
7. Configure the Makefile by adding the .c file to the sources section.

These steps will be discussed in the following sections.

Configure the Header File and Names

Using the e43rb1_fw405_c.h and .c files, copy them into the project and change their names to the new display part number. In this example, e43rb1_fw405_c.h would be changed to e50ra_i_mw490_c.h. Once they are copied and the names changed an adjustment in the header file is required.

In the header file, change the **`#ifndef __E43RB1_FW405_C_H__`** to match the new header file name. See the image below.

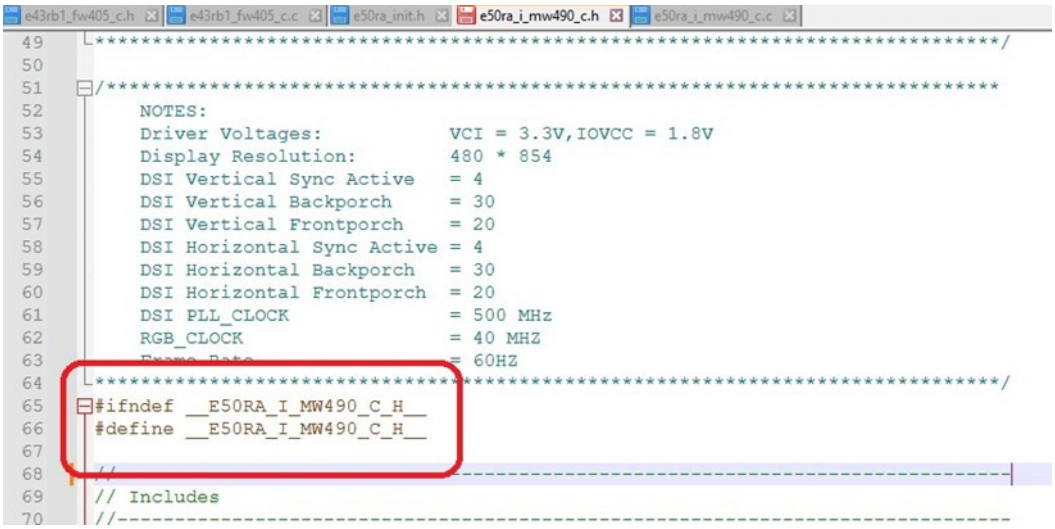

Figure 3: Header File Changes

Editing the Source File

First, looking at the datasheet for the E50RA-I-MW490-C, the TFT controller chip needs to be identified. In this case the controller chip is the same as the E43RB1-FW405-C, the ILI9806E. If the controller chip was different, more changes to the source code would be necessary.

General Information Items	Specification Main Panel	Unit	Note
TFT Display Area (AA)	61.63(H) x 109.65(V) (5.0 inch)	mm	-
Driver Element	TFT active matrix	-	-
Display Colors	16.7M	colors	-
Number of pixels	480(RGB)x854	dots	-
TFT Pixel arrangement	RGB vertical stripe	-	-
Pixel Pitch	0.1284(H)x0.1284(V)	mm	-
Viewing angle	All	o'clock	-
Display mode	Transmissive, Normally Black	-	-
TFT Controller	ILI9806E	-	-
Operating temperature	-20C to 70C	°C	-
Storage temperature	-30C to 80C	°C	-

Figure 4: E50RA-I-MW490-C TFT Controller

Include File Change

In the source file, the #include directive must be changed to the e50ra_i_mw490_c.h header file. After the include, there are definitions for the for the controller chip Page access. Since the controllers are the same between both displays these do not have to be changed.

```
e43rb1_fw405_c.h | e43rb1_fw405_c.c | e50ra_init.h | e50ra_i_mw490_c.h | e50ra_i_mw490_c.c
64  *****************************************************************************/
65
66  //-----------------------------------------------------------------------------
67  // Inculdes
68  //-----------------------------------------------------------------------------
69  #include "e50ra_i_mw490_c.h"
70
71  //-----------------------------------------------------------------------------
72  // Private types
73  //-----------------------------------------------------------------------------
74
75  //-----------------------------------------------------------------------------
76  // Private constants
77  //-----------------------------------------------------------------------------
78
79  //-----------------------------------------------------------------------------
80  // Private macros
81  //-----------------------------------------------------------------------------
82  #define debugLCDCONF(type, ...)            printDEBUG(type, (char *)c_module_name, __V
83
84  static const uint8_t c_module_name[] = "E50RA-I-MW490-C";
85
86  //-----------------------------------------------------------------------------
87  // Private function prototypes
88  //-----------------------------------------------------------------------------
89
90  //-----------------------------------------------------------------------------
91  // Private variables
92  //-----------------------------------------------------------------------------
93  uint8_t Page0[] = { ILI9806E_EXTC_PARAM1, ILI9806E_EXTC_PARAM2,
94                      ILI9806E_EXTC_PARAM3, ILI9806E_EXTC_PARAM4, DSI_PAGE0 };
95
96  uint8_t Page1[] = { ILI9806E_EXTC_PARAM1, ILI9806E_EXTC_PARAM2,
97                      ILI9806E_EXTC_PARAM3, ILI9806E_EXTC_PARAM4, DSI_PAGE1 };
98
99  uint8_t Page2[] = { ILI9806E_EXTC_PARAM1, ILI9806E_EXTC_PARAM2,
00                      ILI9806E_EXTC_PARAM3, ILI9806E_EXTC_PARAM4, DSI_PAGE2 };
01  uint8_t Page3[] = { ILI9806E_EXTC_PARAM1, ILI9806E_EXTC_PARAM2,
02                      ILI9806E_EXTC_PARAM3, ILI9806E_EXTC_PARAM4, DSI_PAGE3 };
03  uint8_t Page4[] = { ILI9806E_EXTC_PARAM1, ILI9806E_EXTC_PARAM2,
04                      ILI9806E_EXTC_PARAM3, ILI9806E_EXTC_PARAM4, DSI_PAGE4 };
05  uint8_t Page5[] = { ILI9806E_EXTC_PARAM1, ILI9806E_EXTC_PARAM2,
06                      ILI9806E_EXTC_PARAM3, ILI9806E_EXTC_PARAM4, DSI_PAGE5 };
07  uint8_t Page6[] = { ILI9806E_EXTC_PARAM1, ILI9806E_EXTC_PARAM2,
08                      ILI9806E_EXTC_PARAM3, ILI9806E_EXTC_PARAM4, DSI_PAGE6 };
09  uint8_t Page7[] = { ILI9806E_EXTC_PARAM1, ILI9806E_EXTC_PARAM2,
10                      ILI9806E_EXTC_PARAM3, ILI9806E_EXTC_PARAM4, DSI_PAGE7 };
```

Figure 5: Includes and Page Variables

Initial Parameter Configuration

The parameters of the TFT LCD will be set and initialized in the `void` **`initLCDConfig(`**`LCDConfig` **`*lcdconfig)`** function. This function is found in the e50ra_i_mw490_c.c file. The argument **`lcdconfig`** is a structure containing all the essential information required by the DSI controller to configure the MIPI DSI interface. This will have all the parameters to set up the display, as an example it contains the horizontal and vertical back porch timings.

```
118
119 //-----------------------------------------------------------------------------
120 // Private functions
121 //-----------------------------------------------------------------------------
122 void initLCDConfig(LCDConfig* lcdconfig)
123 {
124
125     // set initial mode to video
126     lcdconfig->mode = DSI_MODE_VIDEO;
127
128     lcdconfig->virchid = 0;                         //  Virtual Channel ID
129     lcdconfig->virch = 0;                           //  Virtual Channel
130
131     lcdconfig->width = 480;
132     lcdconfig->height = 854;
133
134     lcdconfig->hact = lcdconfig->width;             //  horizontal address
135     lcdconfig->vact = lcdconfig->height;            //  vertical address
136     lcdconfig->hsw = 4;                             //  horizontal synchronization width
137     lcdconfig->hbp = 30;                            //  horizontal back porch
138     lcdconfig->hfp = 20;                            //  horizontal front porch
139     lcdconfig->vsh = 4;                             //  vertical synchronization height
140     lcdconfig->vbp = 30;                            //  vertical back porch
141     lcdconfig->vfp = 20;                            //  vertical front porch
142     lcdconfig->haa = lcdconfig->width;              //  horizontal active area
143     lcdconfig->vaa = lcdconfig->height;             //  vertical active area
144     lcdconfig->hsa = lcdconfig->hsw;                //  horizontal synchronization active
145     lcdconfig->vsa = lcdconfig->vsh;                //  vertical synchronization active
146
147     lcdconfig->ColorCoding = DSI_RGB888;
148     lcdconfig->CommandSize = 480;
149     lcdconfig->TearingEffectSource = DSI_TE_DSILINK;
150     lcdconfig->TearingEffectPolarity = DSI_TE_RISING_EDGE;
151
152     lcdconfig->HSPolarity = DSI_HSYNC_ACTIVE_LOW;
153     lcdconfig->VSPolarity = DSI_VSYNC_ACTIVE_LOW;
154     lcdconfig->DEPolarity = DSI_DATA_ENABLE_ACTIVE_HIGH;
155
156     lcdconfig->AutomaticRefresh = DSI_AR_ENABLE;
157     lcdconfig->TEAcknowledgeRequest = DSI_TE_ACKNOWLEDGE_ENABLE;
158
159     lcdconfig->VSyncPol = DSI_VSYNC_FALLING;
160
161
162
163     lcdconfig->LPVACTLargestPacketSize = 0;
164     lcdconfig->LPLargestPacketSize = 16;
165
166     lcdconfig->LcdClock_kHz = 25000; // HSE in kHz
167
168     debugLCDConfig(0, "init done\n");
169 }
```

Figure 6: Display Configuration Parameters

Video or Command Mode and Additional Configuration

Identifying whether the display operates in MIPI video or command mode can be verified by reviewing the controller datasheet for GRAM. If the controller has GRAM, it can operate in command mode but if there is no GRAM it must operate in video mode. Reviewing the ILI9806E datasheet shows that it has no internal GRAM and operates in video mode. In the source file the mode is set to **DSI_VIDEO_MODE** and does not need to be changed as both displays use the same controller.

```
118
119   //---------------------------------------------------------
120   // Private functions
121   //---------------------------------------------------------
122   void initLCDConfig(LCDConfig* lcdconfig)
123   {
124
125       // set initial mode to video
126       lcdconfig->mode = DSI_MODE_VIDEO;
127
```

Figure 7: Set the Mode to Video

Looking at the datasheet, the display resolution is 480 x 854. Modify the configuration height and width:

```
121   //---------------------------------------------
122   void initLCDConfig(LCDConfig* lcdconfig)
123   {
124
125       // set initial mode to video
126       lcdconfig->mode = DSI_MODE_VIDEO;
127
128       lcdconfig->virchid = 0;
129       lcdconfig->virch = 0;
130
131       lcdconfig->width = 480;
132       lcdconfig->height = 854;
133
134       lcdconfig->hact = lcdconfig->width;
```

Figure 8: Height and Width Configuration

The next set of parameters to consider are the display timings. These can sometimes be found in the display datasheet or from the comments section of the display initialization file. Both can be found on the Focus LCDs website or are available upon request.

```
e43rb1_fw405_c.h | e43rb1_fw405_c.c | e50ra_init.h | e50ra_i_mw490_c.h | e50ra_i_mw490_c.c
1  //*****************************************************
2  //
3  //*****     Focus LCDs      *****
4  //***** LCDs Made Simple   *****
5  //***** www.FocusLCDs.com *****
6  //
7  //*****************************************************
8  // NOTES:
9  // Driver Voltages         = VCI = 3.3V IOVCC = 1.8V
10 // Display Resolution      = 480 x 854
11 // Vertical Sync Active    = 4
12 // Vertical Backporch      = 30
13 // Vertical Frontporch     = 20
14 // Horizontal Sync Active  = 4
15 // Horizontal Backporch    = 30
16 // Horizontal Frontporch   = 18
17 // DCR PLL CLOCK           = 29 MHz
18 // Frame Rate              = 60 HZ
```

Figure 9: Display Initialization Timings

Due to slight variations in timing, some parameters need to be adjusted to ensure proper functioning of the display. Shown below the display initialization file calls for the HFP (horizontal front porch) to be 18, but through testing it was found that 20 displayed the image properly.

```
121 //--------------------------------------------------------------------------------
122 void initLCDConfig(LCDConfig* lcdconfig)
123 {
124
125     // set initial mode to video
126     lcdconfig->mode = DSI_MODE_VIDEO;
127
128     lcdconfig->virchid = 0;                     //  Virtual Channel ID
129     lcdconfig->virch = 0;                       //  Virtual Channel
130
131     lcdconfig->width = 480;
132     lcdconfig->height = 854;
133
134     lcdconfig->hact = lcdconfig->width;         //  horizontal address
135     lcdconfig->vact = lcdconfig->height;        //  vertical address
136     lcdconfig->hsw = 4;                         //  horizontal synchronization width
137     lcdconfig->hbp = 30;                        //  horizontal back porch
138     lcdconfig->hfp = 20;                        //  horizontal front porch
139     lcdconfig->vsh = 4;                         //  vertical synchronization height
140     lcdconfig->vbp = 30;                        //  vertical back porch
141     lcdconfig->vfp = 20;                        //  vertical front porch
142     lcdconfig->haa = lcdconfig->width;          //  horizontal active area
143     lcdconfig->vaa = lcdconfig->height;         //  vertical active area
144     lcdconfig->hsa = lcdconfig->hsw;            //  horizontal synchronization active
145     lcdconfig->vsa = lcdconfig->vsh;            //  vertical synchronization active
146
```

Figure 10: LCDConfig Timings

Configure the Initialization Sequence

Having adjusted the timings for the display, the initialization sequence will be modified for the new display. In the previous section, the display timings were retrieved from the display initialization file. The display initialization is included in that file.

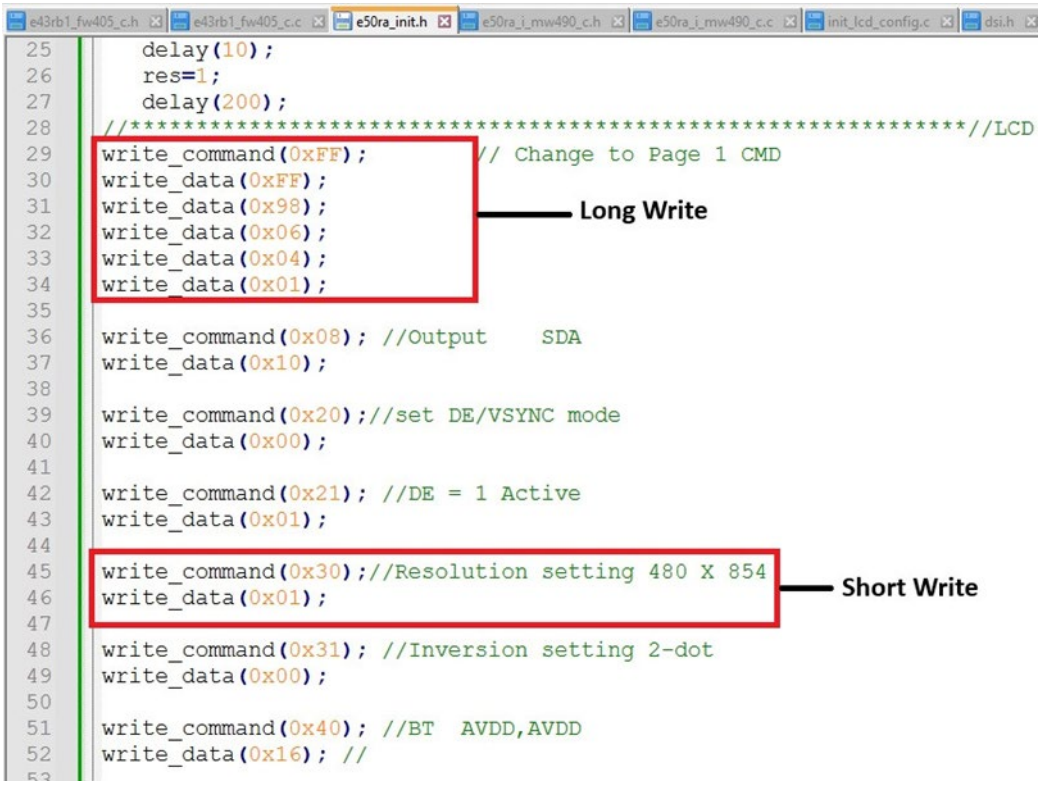

Figure 11: Initialization Sequence

In the figure above, there are two types of data writes, a long and short. The **`write_command(0xFF)`** is the address of the register where the data will be written, in this instance the **`0xFF`** register. The **`write_data()`** is the data that will be written to the register. This command takes 5 parameters and requires the use of a long write function.

The short write function is used when there is only 1 data parameter.

There is also a write command function that has no corresponding write data function. In these instances, a short write function is used and **`0x00`** is placed in the data parameter argument.

Edit the `void` **`runInitSeqLCDConfig(`**`LCDConfig` **`*lcdconfig)`** function for the new initialization sequence. This is required for all the write commands in the display initialization file. Shown below is a section of the modified **`runInitSeqLCDConfig()`** function.

```
void runInitSeqLCDConfig(LCDConfig* lcdconfig)
{

    // DISPLAY POWER ON SEQUENCE
    {

        uint32_t long_pkt_write_type = (DSI_DCS_LONG_PKT_WRITE);
        uint32_t short_pkt_write_type = (DSI_DCS_SHORT_PKT_WRITE_P1);
        {
            longWriteDSI(lcdconfig->virch, long_pkt_write_type, 5, 0xFF, Page1);

            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x08, 0x10); //   Output SDA
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x20, 0x00); // set DE/VSYNC mode
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x30, 0x01); // resolution setting 480x854

            /***************************** POWER CONTROL *****************************/
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x40, 0x16); // BT AVDD
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x41, 0x33); //
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x42, 0x03); // VGL   = DDVDH + VCIP - DDVDL, VGH = 2DDVDL -
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x43, 0x09); // set VGH clamp level
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x44, 0x06); // set VGL clamp level
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x50, 0x88); // VREG1
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x51, 0x88); // VREG2
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x52, 0x00); // Flicker MSB
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x53, 0x49); // Flicker LSB, VCOM
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x55, 0x40); // Flicker


            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x60, 0x07); //
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x61, 0x00); //
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x62, 0x07); //
            shortWriteDSI(lcdconfig->virch, short_pkt_write_type, 0x63, 0x00); //
```

Long Write

Short Write

Figure 12: LCD Configuration Sequence

In the figure above, the **`longWriteDSI()`** and **`shortWriteDSI()`** functions are used to send commands to the display. Looking back at Figure 14, there are 5 parameters to the page change command. In Figure 15, the long write is the DSI version of the page change command. The last parameter, Page1, is the array from Figure 8 which stores the 5 parameters.

Following the long write, the short write for setting the resolution takes the command **`0x30`** and the data parameter **`0x01`**. This sets the resolution to 480 x 854.

Checking the LCD controller datasheet can confirm that writing **`0x01`** to register **`0x30`** will set the resolution to 480 x 854. The controller datasheets can be found on Focus LCDs website. The specific ILI9806E datasheet can be found here.

4.4.12. Resolution Control (30h)

Page 1 Command Set		30h : RESCTRL (Resolution Control)								
	Write / Read	D7	D6	D5	D4	D3	D2	D1	D0	Default (Hex)
Command	Write	0	0	1	1	0	0	0	0	30h
1st Parameter	Write / Read	0	0	0	0	0	RES[2:0]			02h

Description

RES[2:0]: These bits are used to select panel resolution.

The relation between bit setting and resolution is illustrated as below table.

RES[2:0]	Resolution
000	480X864
001	480X854
010	480X800
011	480X640
100	480X720
Other setting are inhibited	

Figure 13: Resolution Control

Modify the Makefile

The final step in the process is to modify the Makefile with the new display source file. In the original Makefile, under SRCS, the e43rb1_fw405_c.c file was listed.

Figure 14:E43RB1 Makefile

The source file must be replaced with the new source file e50ra_i_mw490_c.c for the code to compile correctly.

```
e43rb1_fw405_c.h   e43rb1_fw405_c.c   e50ra_init.h   e50ra_i_mw490_c.h   e50ra_i_mw490_c.c

#wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww
# Add your source files here
#wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww
# firmware output file name
PROJ_NAME=main


# *.c file sources
SRCS = main.c ./sdk/system_stm32h7xx.c
SRCS += clock.c delay.c
SRCS += fwver.c misc.c debug.c usart.c

SRCS += fmc.c dsi.c
SRCS += touch.c i2cs.c
SRCS += focusimage.c
SRCS += e50ra_i_mw490_c.c
```

Figure 15: E50RA Makefile

Once all the changes have been made to the various headers, source, and Makefiles the demo will build and can be downloaded into the STM32H7 Discovery board with the appropriate display attached.

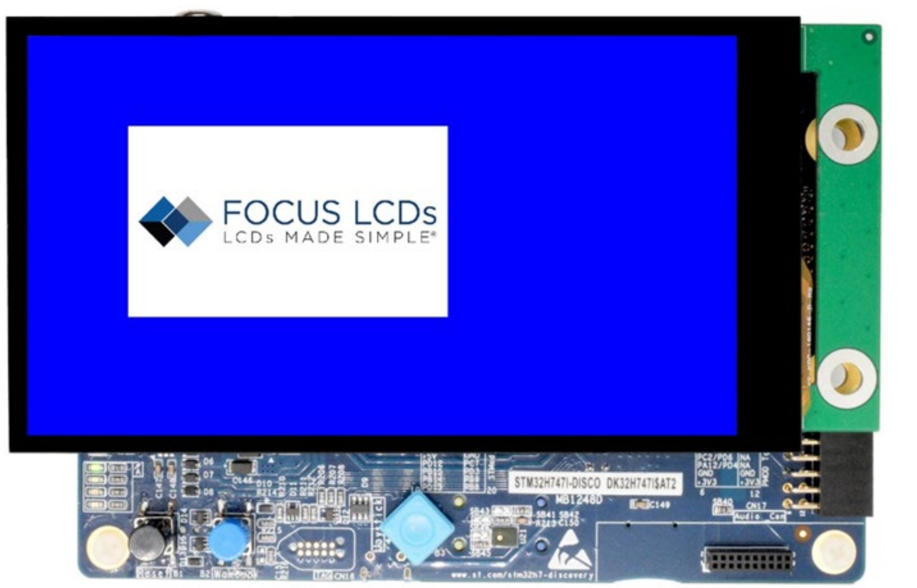

Figure 16 E50RA-I-MW490-C Running Demo Firmware

Summary

In Part 1, the hardware requirements for a MIPI display demo were presented. How to assemble the hardware was shown. Finally, a brief overview of the MIPI DSI interface and the STM32 DSI Host peripheral were discussed.

In Part 2 presented here, the development tools and frame buffer consideration were briefly discussed. The demonstration firmware operation was mentioned along with the touch interface. The following sections went through the operation of modifying the firmware for the E50RA-I-MW490-C display. Through the modification of the firmware, the structure and basic layout of the code is shown.

LCD Handling Precautions

- Do not store the TFT-LCD module in direct sunlight, best stored in a dark place.
- Do not leave it exposed to high temperature and high humidity for a long period of time.
- Recommended temperature range is 0 to 35 °C, relative humidity should be less than 70%.
- Stored modules away from condensation as formation of dewdrops may cause an abnormal operation or failure of the module.
- Protect the module from static discharge.
- Do not press or scratch the surface and protect it from physical shock or any force.

Disclaimer