Freescale Semiconductor
Application Note

Document Number: AN4452
Rev. 0, 1/2012

Display (eGUI) using TWR-LCD

Working with the Freescale Tower System

by: Ju Yingyi
Application Engineer, AISG

This application note describes how to use the TWR-LCD board to display eGUI or other graphic user interfaces. This document covers all Freescale ColdFire and Kinetis 32-bit MCU and MPU tower cards that have Flexbus or SPI interfaces.

Contents

1 Introduction 1
2 Hardware interface 2
2.1 TWR-LCD board 2
2.2 Flexbus/mini-Flexbus interface for LCD display . . . 4
2.3 SPI for LCD display 8
2.3 Touch screen interface 9
3 Display eGUI on TWR-LCD board. 11
3.1 CW10.1 and IAR project 12
3.2 eGUI configurations. 14
3.3 Baremetal project 17
3.4 MQX project. 19
3.5 Screen shots 19
4 Display microWindows with TWR-MCF5441X 20
5 Summary 21

1 Introduction

Freescale continues to introduce new 32-bit parts, each of them with Flexbus or mini-Flexbus and SPI/QSPI/DSPI interfaces that can be used to connect with the Freescale TWR-LCD board. This document describes how to use these new tower boards to display eGUI or other GUI (microWindows) working with the TWR-LCD board. The following contents show the hardware connections from the MCU board to the TWR-LCD board and the eGUI software package that drives the LCD module, including the baremetal eGUI version and eGUI running with MQX™. It covers most existing ColdFire V1, V2, and V4 MCU/MPU tower

© Freescale Semiconductor, Inc., 2012. All rights reserved.

cards, Coldfire+ MCU tower cards, and Kinetis tower cards.

For more detailed information regarding the Freescale Tower System, refer to www.freescale.com/tower.

For more detailed information on the Freescale Embedded Graphical User Interface (eGUI), please refer to www.freescale.com/egui.

2 Hardware interface

2.1 TWR-LCD board

The TFT-LCD module populated on the TWR-LCD board is integrated with Solomon Systech TFT-LCD controller driver SSD1289. This TFT-LCD controller driver integrates the graphic display data RAM, power circuits, gate driver, and source driver into a single chip. On this board, the graphic display data RAM is interfaced with the common MCU/MPU through a 16-bit 6800-series / 8080-series compatible parallel interface or SPI. Please refer to AN4153, “Using Freescale eGUI with TWR-LCD on MCF51MM Family” for more information on SSD1289.

Download the schematics of TWR-LCD board from http://cache.freescale.com/files/soft_dev_tools/hardware_tools/schematics/TWRLCDSCH.pdf?fpsp=1.

2.1.1 SW1 settings

SW1 settings for the 16-bit FlexBus/mini-Flexbus interface:

SW1[1]: on, PS2 = 0

SW1[2]: off, PS0 = 1

SW1[3]: on, disables the control of the MCF51JM128 on board

SW1[4]: for micro SD card interface, doesn’t matter

SW1[5]: select SPI channel, doesn’t matter in this mode

SW1[6]: TP_SEL, doesn’t matter in this mode, better on

SW1[7]: on/off, backlight on/off

SW1[8]: option, user can let MCU/MPU tower board output PWM wave to control the buzzer on board

SW1 settings for SPI interface:

SW1[1]: off, PS2 = 1

SW1[2]: on, PS0 = 0

SW1[3]: on, disables the control of the MCF51JM128 on board

SW1[4]: for micro SD card interface, doesn’t matter

SW1[5]: select SPI channel, on is for SPI channel 0, off selects SPI channel 1

SW1[6]: TP_SEL, doesn’t matter in this mode, better on

SW1[7]: on/off, backlight on/off

SW1[8]: option, you can let MCU/MPU tower board output PWM wave to control the buzzer on board

Because there is an MCF51JM128 soldered on the TWR-LCD board, it can work standalone with the MCF51JM128.

SW1 settings for standalone (MCF51JM128, SPI):

SW1[1]: off, PS2 = 1

SW1[2]: on, PS0 = 0

SW1[3]: MUST be off, controlled by MCF51JM128

SW1[4]: for micro SD card interface, doesn't matter

SW1[5]: select SPI channel, doesn't matter in this mode

SW1[6]: MUST be off

SW1[7]: on/off, backlight on/off

SW1[8]: doesn't matter in this mode

2.1.2 SW5 settings

This DIP switch is for the touch screen interface. Turn all switches on when you want to connect any MCU/MPU board. It is recommended that all switches be turned off when working alone.

2.1.3 SW3 settings

Reset button. Not only does it reset the MCF51JM128 on board, it also resets the whole tower system.

2.1.4 SW4 settings

This push button is used for the USB bootloader of the MCF51JM128 on board. To enter USB bootloader mode, set SW1 as "SW1 settings for standalone (MCF51JM128, SPI)" first, then connect a USB cable to J7 with PC, keep pressing SW4, and press SW3 to reset the board. If entering bootloader mode successfully, the LCD will display "Bootloader mode: waiting for S19 file…" on the upper left corner of the screen and there will be a USB mass storage device installed on your PC as in Figure 1 below.

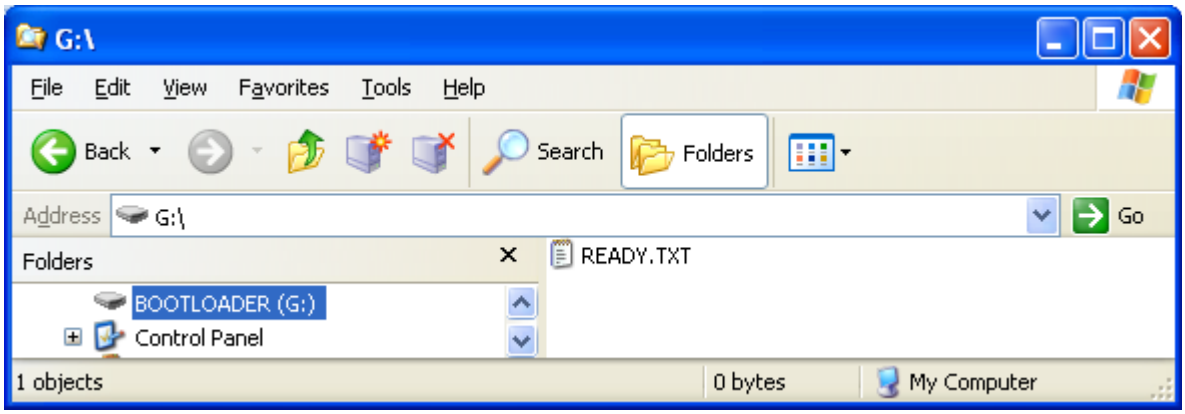

Figure 1. Bootloader drive

If you want to update the firmware of the MCF51JM128 on board, drag and drop the .S19 file into this drive just like copying a file into it. You can find more information on the USB bootloader and its source code on the TWR-LCD board's *Getting Started* CD.

2.1.1 J3 settings

This jumper is used for programming/debugging the MCF51JM128 on board. However, if you erase the original firmware or fail to boot from USB bootloader, you could reprogram the firmware by using P&E USB BDM Multilink or P&E USB Multilink Universal.

2.2 Flexbus/mini-Flexbus interface for LCD display

2.2.1 TWR-LCD side

The Flexbus/mini-Flexbus interface on the TWR-LCD board is configured to 16-bit data bus mode.

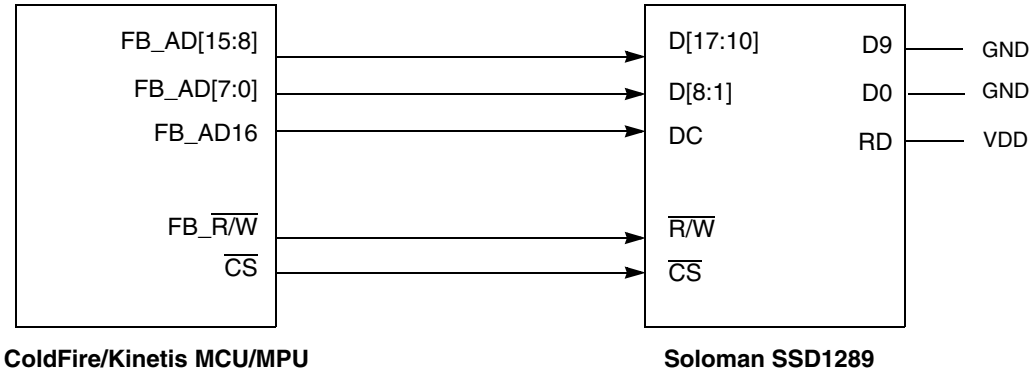

Figure 2. Flexbus/mini-Flexbus interface on TWR-LCD board

Because the DC signal of SSD1289 is connected to FB_AD16 in this design, when FB_AD16 is driven low, the index register of SSD1289 can be accessed. While FB_AD16 is driven high, the control register of SSD1289 or the display data can be accessed.

For example, assume that $\overline{CS0}$ of the MCU is connected to $\overline{CS}$ of the SSD1289, and that CSAR0 on the MCU side has been set to 0x400000. In this case, address 0x400000 is used to access the index register of the SSD1289, while address 0x410000 could be used to access control register or display data of the SSD1289.

2.2.2 Controller module side

There are several important things when using Flexbus/mini-Flexbus interface to display something on the LCD. Please read the following contents before going deeply into LCD low-level driver.

2.2.1.1 Address/Data Bus Multiplexing or Non-multiplexing mode

First, let's compare schematics of TWR-MCF51CN with TWR-K60N512. Refer to the figures below.

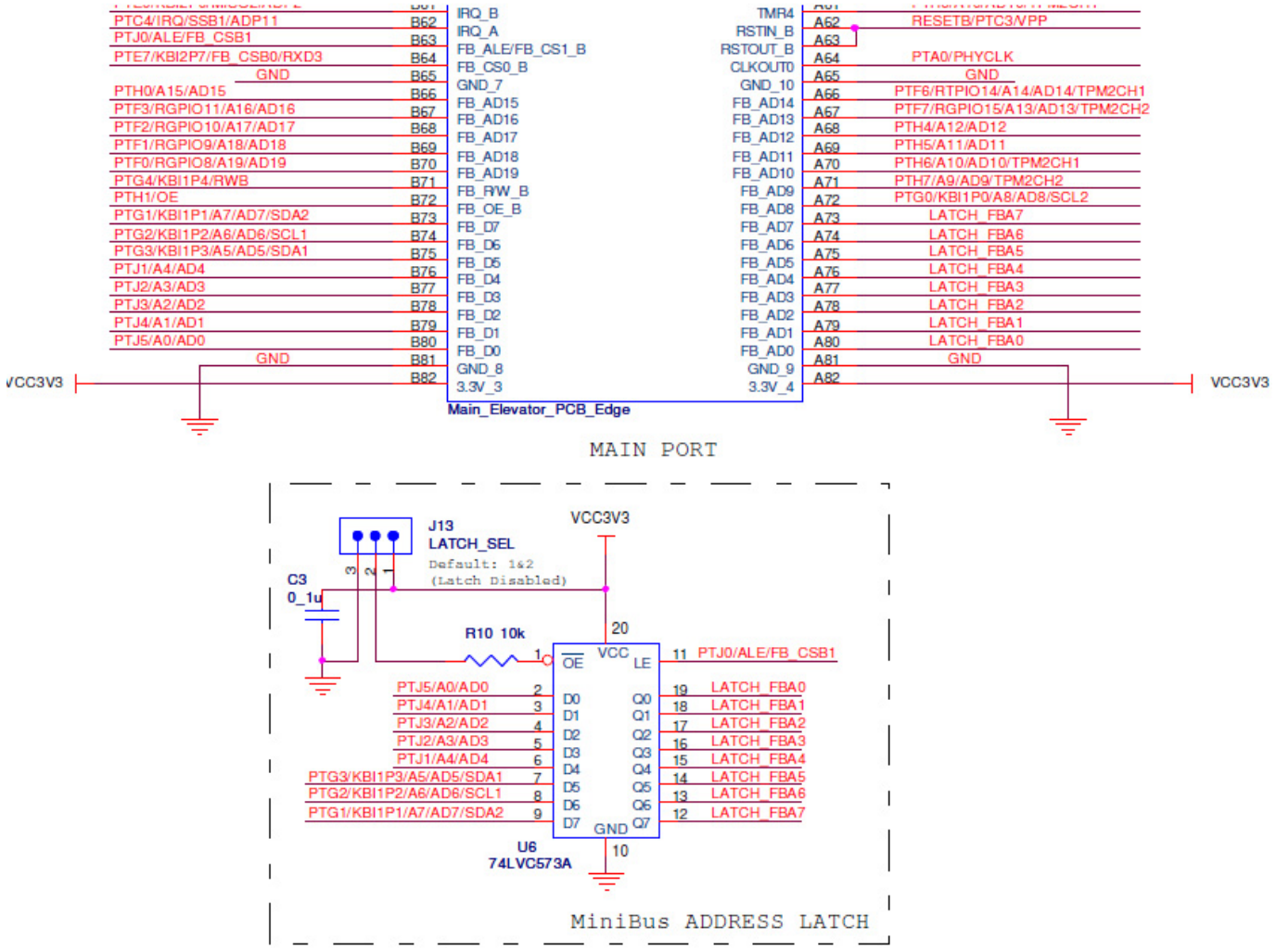

Figure 2. Mini-Flexbus interface of TWR-MCF51CN

There is a latch chip on the TWR-MCF51CN board. This chip is used because the mini-Flexbus interface of the TWR-MCF51CN is designed to work in Address/Data Bus Multiplexing mode.

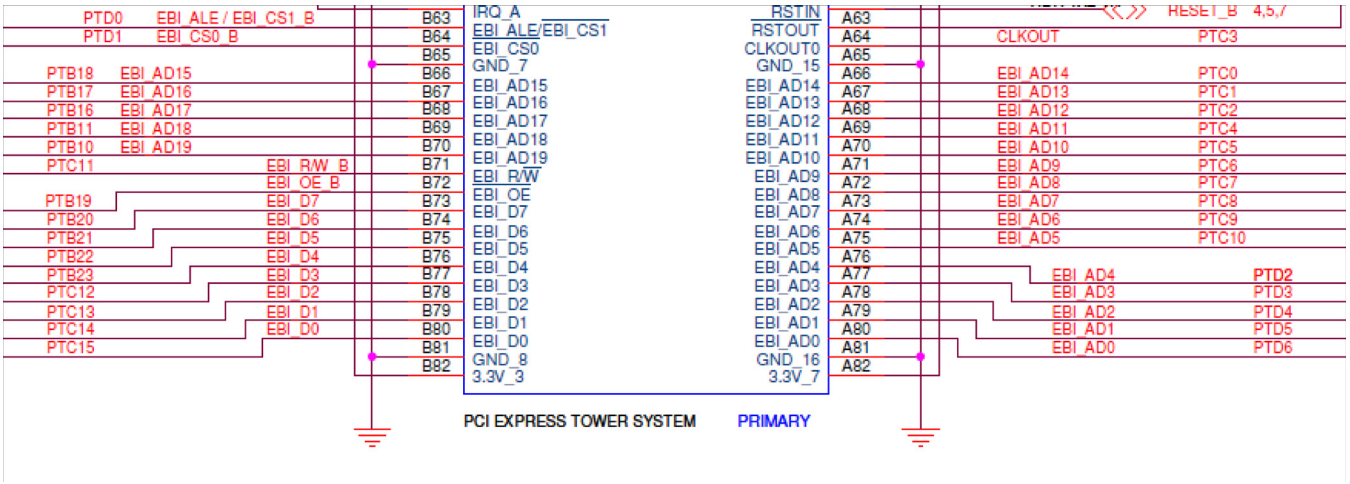

Figure 3. Flexbus interface of TWR-K60N512

There is no latch chip on the TWR-K60N512 board, because the Flexbus interface of TWR-K60N512 is designed to work in Address/Data Bus Non-multiplexing mode when connected to the TWR-MEM card.

Because the data bus on TWR-LCD board is connected to FB_AD[15:0] (or EBI_AD[15:0]), we don't need signals on B73–B80 (These signals are for Address/Data Bus Non-multiplexing mode only). But the user must pay attention to the FB_ALE signal when using TWR-MCF51CN. FB_ALE must keep driving high during the whole data access cycle. So ALE signal could be configured as a GPIO output pin and always output high. If FB_ALE works normally during a data access cycle, data on FB_AD[7:0] could not pass latch chip (74LVC573A) because FB_ALE is low when Flexbus/mini-Flexbus needs to latch data from LCD controller.

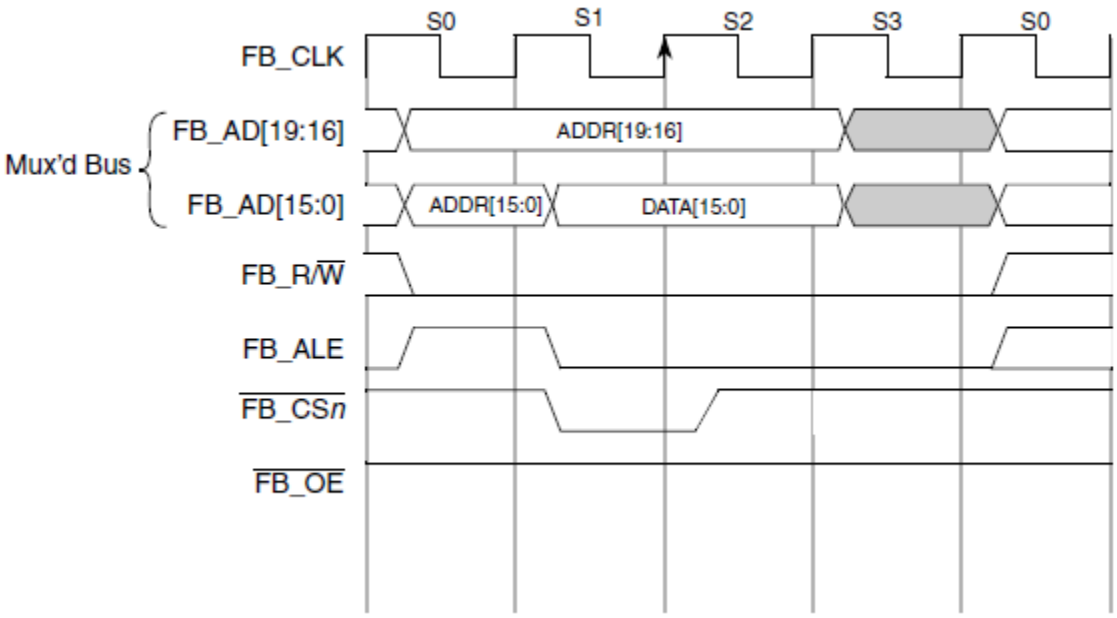

Figure 4. Flexbus/mini-Flexbus single word-write transfer

Until now, because of heavy pin multiplexing, Freescale has several tower cards whose Flexbus/mini-Flexbus interface is designed for Address/Data Bus Multiplexing mode and there is an address-latching chip on the board:

TWR-MCF51CN

TWR-MCF51JF

TWR-MCF51QM

TWR-K40X256

TWR-K53N512

2.2.2.1 Data byte alignment

When you want to display something on the TWR-LCD board with your controller module, remember to set proper Data Byte Alignment (if Flexbus/mini-Flexbus controller of MCU/MPU has CSCR*n*[BLS] bit) according to physical connections.

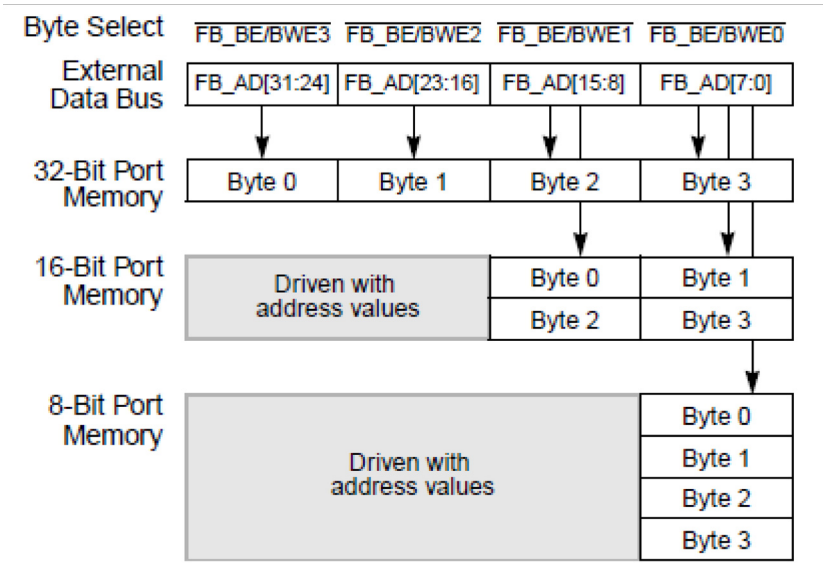

Figure 5. Connections for external memory port sizes (CSCR*n*[BLS] = 1)

Generally speaking, Coldfire V1 and Coldfire+ MCUs only have mini-Flexbus interface and it doesn't have this control bit. Those Coldfire V4 and Kinetis MCUs have Flexbus interface and do have this control bit. It must be set to 1 for the TWR-LCD board.

2.2.2.2 Wait states

According to the SSD1289 data sheet, the minimum write clock cycle time is 100 ns. And there are at least 4 FB_CLK cycles during one read/write flex bus cycle. So FB_CLK should not be higher than 40 MHz, or proper wait cycles must be inserted.

For example:

TWR-MCF51CN/TWR-MCF51JE/TWR-MCF51MM/TWR-MCF51JF/TWR-MCF51QM, when system clock = 50 MHz, FB_CLK = 25 MHz, no wait state needed.

TWR-K40X256/TWR-K53N512/TWR-K60N512, when system clock = 100 MHz, FB_CLK = 50 MHz, 1 wait state is needed.

TWR-MCF5441X, when system clock = 250 MHz, FB_CLK = 1/4 system clock = 62.5 MHz, 3 wait states are needed. You can change FB_CLK = 1/2 system clock by setting MISCCR2[FBHALF] = 0, in which case 9 wait states are needed.

2.2.2.3 Base address settings

The base address of TWR-LCD can be initialized by setting CSAR*n* and CSMR*n*. Please read the memory map overview in the corresponding MCU/MPU's reference manual first to determine the base address. Different devices have different memory areas for off-chip expansion.

- ColdFire V1 & ColdFire+ MCUs can use 0x0040_0000–0x007FF_FFFF and 0x00A0_0000–0x00BFF_FFFF
- Coldfire V2 MCU (MCF5225X) can use 0x8000_0000–0xFFFF_FFFF
- Coldfire V4 MPU (MCF5441X) can use 0x0000_0000–0x3FFF_FFFF() and 0xC000_0000–0xDFFF_FFFF
- Kinetis family MCUs can use 0x6000_0000–0xDFFF_FFFF;

Do NOT set address value out-of-range of the MCU/MPU on the controller module. And also remember to set CSMR0[V] = 1 to enable global chip-select even if you connect other $\overline{CS}$ to the TWR-LCD board.

2.3 SPI for LCD display

Because PS3 and PS1 are fixed to high on TWR-LCD board, the user could use 4-wire SPI mode (PS3 = PS2 = PS1 = HIGH, PS0 = LOW) or 3-wire SPI mode (PS3 = PS2 = PS1 = PS0 = HIGH):.

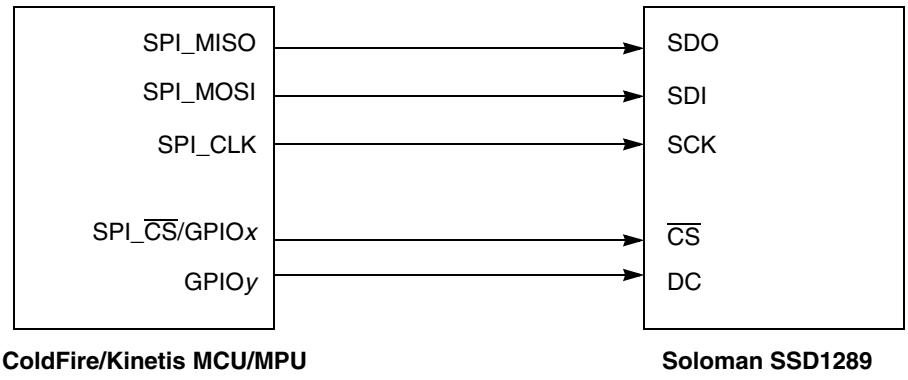

Figure 6. SPI interface on TWR-LCD

2.3.1 4-wire SPI mode

In this mode, on the TWR-LCD side, SDI, SCK, $\overline{CS}$, and DC are necessary. SDI is shifted into an 8-bit shift register on every rising edge of SCK in the order of MSB first. DC is sampled on every eighth clock to determine whether the data byte in the shift register is written to the command register or display data RAM at the same clock. However, on the MCU/MPU side, $\overline{CS}$ could be controlled by SPI_$\overline{CS}$ or a GPIO pin, while DC must be controlled by a GPIO pin.

2.3.2 3-wire SPI mode

Compared with 4-wire SPI mode, on the TWR-LCD side, the DC signal is not used. There are altogether 9 bits that will be shifted into the shift register on every ninth clock in sequence: DC bit, D7 to D0 bit. The DC bit will determine whether the following data byte in the shift register is written to the command register or display data RAM. On the MCU/MPU side, $\overline{CS}$ could be controlled by SPI_$\overline{CS}$ or a GPIO pin.

NOTE

The current released eGUI package does NOT support 3-wire SPI mode.

If using 16-bit SPI mode on the MCU/MPU side, it is recommended that the data be transferred twice. For example, if you want to send data 0x9290 to register 0x03, the data format is 0303 (index of command register), 9292 (high byte), 9090 (low byte). For more details, please refer to the SSD1289 Application Note available at www.solomon-systech.com.

SSD1289 has a limitation for SPI clock speed. The maximum frequency is 13 MHz. So the user must set proper baud rate (≤13MHz) for the SPI module of tower controller module.

2.3 Touch screen interface

There is a touch screen covered LCD screen on the TWR-LCD board. It is a 4-wire resistive touch screen. Figure 7 shows a basic diagram of the construction and operation of a touch screen.

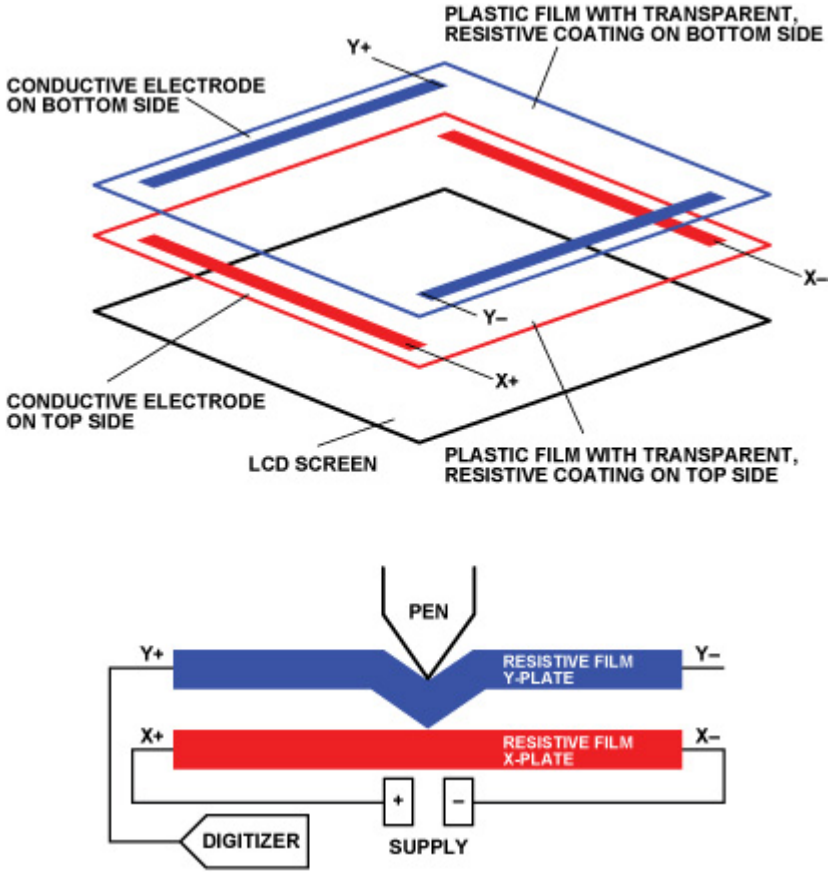

Figure 7. Basic touch screen diagram of the construction and operation

The touch screen is formed by two plastic films, each coated with a conductive layer of metal (usually indium tin oxide, ITO) that are separated by an air gap. One plate, X-plate in the diagram above, is excited by the supply voltage. When the screen is touched, the two conductive plates come together, creating a resistor divider along the X-plate. The voltage at the point of contact, which represents the position on the X-plate, is sensed through the Y+ electrode, as shown in Figure 8. The process is then repeated by exciting the Y-plate and sensing the Y position through the X+ electrode.

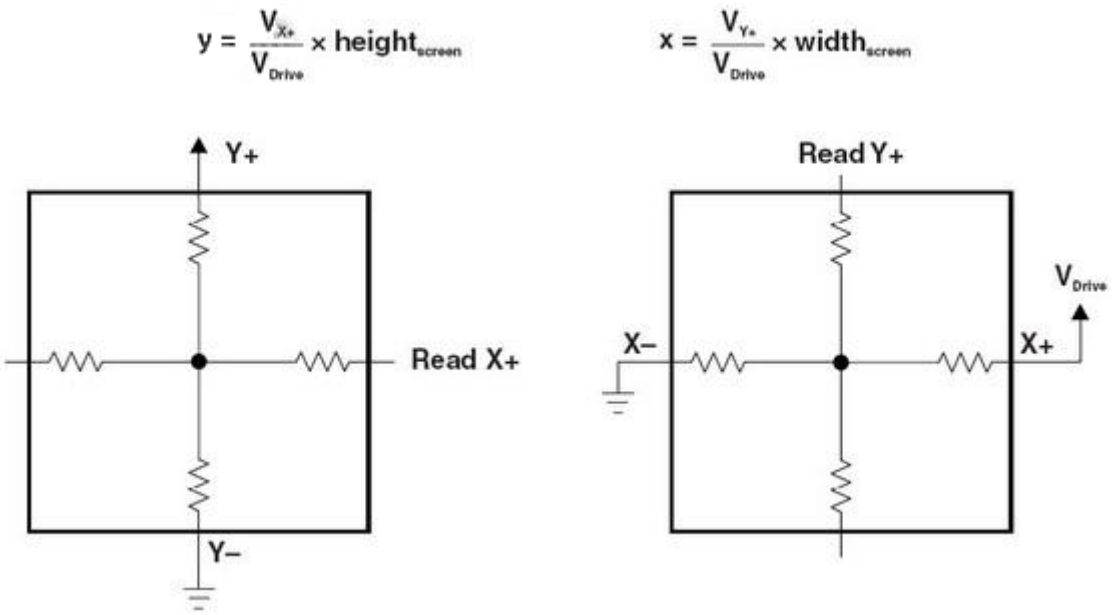

Figure 8. Position measurement

So, we could use two ADC channels with the GPIO pin multiplexed and two GPIO pins to simulate a touch screen controller.

According to Figure 8, to read X position:

1. Drive HIGH on X+ and drive low on X-, set Y- to HIGH-Z (may set it as input port).
2. Set Y+ as ADC channel, then, can read raw X voltage value from Y+.

To read Y position:

1. Drive HIGH on Y+ and drive low on Y-, set X- to HIGH-Z (may set it as input port).
2. Set X+ as ADC channel, then, can read raw Y voltage value from X+.

However, this is just an example. The user can also use Y-/X- to read X/Y position. But please remember to set another side (Y+/X+) to HIGH-Z to avoid reading the invalid value.

After getting the raw voltage value, the X/Y position can be calculated with the equation in Figure 8.

3 Display eGUI on TWR-LCD board

You can get the latest eGUI release from www.freescale.com/egui. The following table can be found in the eGUI release package (\Freescale_embedded_GUI_SW_Official_Demos\EGUI_D4D_Demo\):

Board Name	MCU Type	LCD Controller	Interface				Bare Metal			MQX 3.6			MQX 3.7		
			Serial	Parallel											
			SPI	FlexBus(6800)	Intel(8080)	RGB	CW Classic	IAR 6.1	CW 10.1	CW Classic	IAR 6.1	CW 10.1	CW Classic	IAR 6.1	CW 10.1
TWR-MCF51CN128	ColdFire V1	SSD1289													
TWR-MCF51MM256	ColdFire V1	SSD1289													
TWR-MCF51JE256	ColdFire V1	SSD1289													
TWR-LCD	ColdFire V1	SSD1289													
TWR-MCF52259	ColdFire V2	SSD1289													
M52277EVB	ColdFire V2	Frame Buffer													
TWR-K40X256	Kinetis	SSD1289													
TWR-K60N512	Kinetis	SSD1289													
TWR-MPC5125	MPC	Frame Buffer													
DEMOQE_HCS08QE128	HCS08	SSD1289													
DEMOQE_MCF51QE128	ColdFire V1	SSD1289													

Legend	
Option done (in rel. 2.1)	
Option possible	
Not supported by MQX	
Not applicable	

Table 1. eGUI Options

The following content focuses on how to display eGUI on TWR-LCD by configuring IAR or CW 10.1 project properly. TWR-K60N512 projects are shown as an example.

Before starting:

- Please read *2.1 TWR-LCD Board* or \Freescale_embedded_GUI_SW_Official_Demos\EGUI_D4D_Demo\readme.txt for jumper/switch settings.
- If you are using TWR-K60N512 Rev.C or the older version board, please remove C5 (Rev. A or Rev. B) or C2 (Rev. C). This capacitance will impact on Flexbus signal (FB_AD9). If you are using TWR-K60N512 Rev. D, remove the jumper on J16.

3.1 CW10.1 and IAR project

To open the CW 10.1 project, please run CW 10.1 first, then click File → Import…

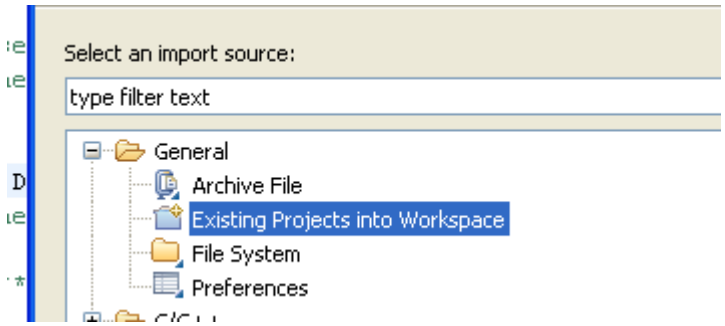

Figure 9. How to open CW10.1 Project 1

Select “Existing Projects into Workspace” and click “Next”:

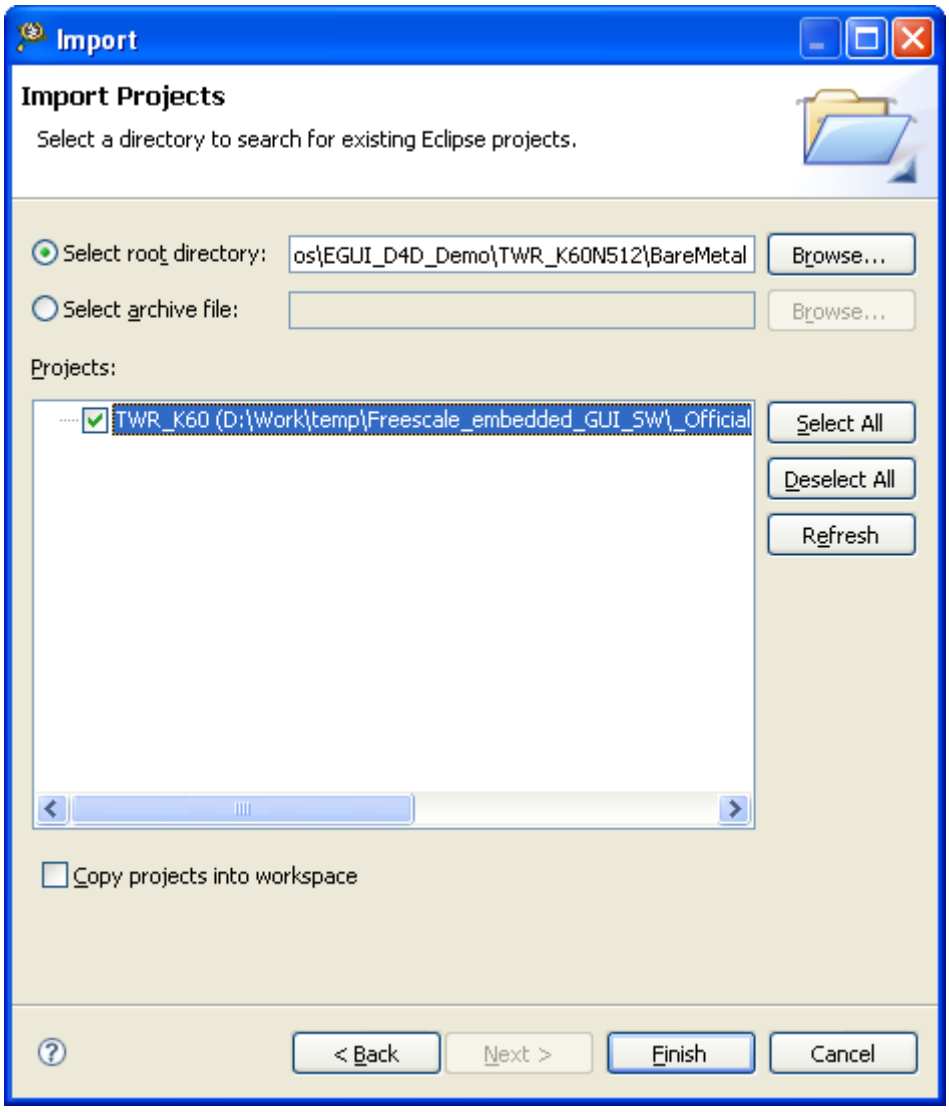

Figure 10. How to open CW10.1 Project 2

Select the root directory by “Browse…”, then you can find the project to be open. Click “Finish.”

To open the IAR project, just double click the “project.eww” right in the folder of IAR_6_1.

3.2 eGUI configurations

Before compiling, you should configure LCD and touch screen drivers according to the settings of your TWR-LCD board and controller module.

d4d_user_cfg.h:

```
// Please define a used low LCD driver
#define D4D_LLD_LCD d4dlcd_ssd1289   // the name of low level driver
                                     // descriptor structure

// Please (if it's needed) define a used LCD hw interface driver
#define D4D_LLD_LCD_HW d4dlcdhw_kinetis_spi
//#define D4D_LLD_LCD_HW d4dlcdhw_flexbus_16b

// Please define a used touch screen driver if touch screen is used in project
#define D4D_LLD_TCH  d4dtch_resistive

// Please (if it's needed) define a used touch screen hw interface driver
#define D4D_LLD_TCH_HW d4dtchhw_kinetis_adc
```

However, besides the low-level driver definitions, you may also modify other software configurations to meet your own requirement.

d4dlcdhw_flexbus_16b_cfg.h and d4dlcdhw_kinetis_spi_cfg.h:

These two header files are used to configure Flexbus & SPI hardware settings for LCD driver.

d4dlcdhw_flexbus_16b_cfg.h:

```
// Alternative function 5 = FB enable
#define ALT5 (PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK)
// FlexBus = Sysclk/2 = ~48MHz
#define FLEX_CLK_INIT (SIM_CLKDIV1 |= SIM_CLKDIV1_OUTDIV3(1))

#define D4DLCD_DISPLAY_MCU_USER_INIT SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK | SIM_SCGC5_PORTB_MASK
| SIM_SCGC5_PORTC_MASK | SIM_SCGC5_PORTD_MASK | SIM_SCGC5_PORTE_MASK;\
PORTC_PCR0=ALT5; PORTC_PCR1=ALT5; PORTC_PCR2=ALT5; PORTC_PCR3=ALT5; PORTC_PCR4=ALT5;
PORTC_PCR5=ALT5; PORTC_PCR6=ALT5; PORTC_PCR7=ALT5; PORTC_PCR8=ALT5; PORTC_PCR9=ALT5;
PORTC_PCR10=ALT5; PORTC_PCR11=ALT5;\
PORTD_PCR1=ALT5; PORTD_PCR2=ALT5; PORTD_PCR3=ALT5; PORTD_PCR4=ALT5; PORTD_PCR5=ALT5;
PORTD_PCR6=ALT5; PORTB_PCR17=ALT5; PORTB_PCR18=ALT5;\
FLEX_CLK_INIT;SIM_SOPT2 |= SIM_SOPT2_FBSL(3); SIM_SCGC7 |= SIM_SCGC7_FLEXBUS_MASK;

#define D4DLCD_FLEX_BASE_ADDRESS  0x60010000
#define D4DLCD_FLEX_DC_ADDRESS    0x60000000
#define D4DLCD_FLEX_ADRESS_MASK   0x00010000

#define D4DLCD_FLEX_CS 0
//#define CSCR_RESET 0x003ffC00
#define CSCR_RESET 0x00000000

// Kinetis Flexbus Register Macro redefinitions
  #define D4DLCD_FLEX_CSAR FB_CSAR0
```

```
  #define D4DLCD_FLEX_CSMR FB_CSMR0
  #define D4DLCD_FLEX_CSCR FB_CSCR0

// MUX mode + Wait States
  #define D4DLCD_FLEX_CSCR_MUX_MASK  (FB_CSCR_BLS_MASK | CSCR_RESET)
  #define D4DLCD_FLEX_CSMR_V_MASK    FB_CSMR_V_MASK
  #define D4DLCD_FLEX_CSCR_AA_MASK   FB_CSCR_AA_MASK
  #define D4DLCD_FLEX_CSCR_PS1_MASK  (FB_CSCR_PS(2))

  /****************************************
  * Signals definition
  ****************************************/

// Define void macros, because TWR-K60 board doesn't use RESET pin
  #define D4DLCD_INIT_RESET
  #define D4DLCD_ASSERT_RESET
  #define D4DLCD_DEASSERT_RESET

// RESET pin definition -if used

  //#define D4DLCD_RESET        x           // Pin number
  //#define D4DLCD_RESET_PORT   GPIOx_PDOR  // PortX Output Data Output
  //#define D4DLCD_RESET_DDR    GPIOx_POER  // PortX Output Enable
  //#define D4DLCD_RESET_PCR    PORTx_PCRx  // PAD configuration register
```

d4dlcdhw_kinetis_spi_cfg.h:

```
/****************************************
  * Signals definition
  ****************************************/

  #define D4DLCD_SPI_ID 2       // SPI module number
  #define D4DLCD_SPI_PCS_ID 0   // Chip Select used by SPI

  // tweak off the SPI frequency to maximum 25Mb/s, standard 12Mb/s
  #define D4DLCD_SPI_DBL_BRATE



  // configure PADs for SPI functionality

  #define D4DLCD_SPI_MISO_PCR   PORTD_PCR14
  #define D4DLCD_SPI_MOSI_PCR   PORTD_PCR13
  #define D4DLCD_SPI_CLK_PCR    PORTD_PCR12
  #define D4DLCD_SPI_CS_PCR     PORTD_PCR11 // PCS0
  // #define D4DLCD_SPI_CS_PCR   PORTD_PCR15 // PCS1

  #define D4DLCD_DC             17          // PTB_17
  #define D4DLCD_DC_PORT        GPIOB_PDOR  // PortB Output Data Output
  #define D4DLCD_DC_DDR         GPIOB_PDDR  // PortB Output Enable
  #define D4DLCD_DC_PCR         PORTB_PCR17 // PAD configuration register

  // RESET pin definition -if used

  //#define D4DLCD_RESET        x           // Pin number
  //#define D4DLCD_RESET_PORT   GPIOx_PDOR  // PortX Output Data Output
  //#define D4DLCD_RESET_DDR    GPIOx_POER  // PortX Output Enable
  //#define D4DLCD_RESET_PCR    PORTx_PCRx  // PAD configuration register
```

```
  // BACKLIGHT pin definition -if used

  //#define D4DLCD_BACKLIGHT        x           // Pin number
  //#define D4DLCD_BACKLIGHT_PORT   GPIOx_PDOR  // PortX Output Data Output
  //#define D4DLCD_BACKLIGHT_DDR    GPIOx_POER  // PortX Output Enable
  //#define D4DLCD_BACKLIGHT_PCR    PORTx_PCRx  // PAD configuration register


 // Enable clock to SPI module and Peripheral ports
  #define D4DLCD_DISPLAY_MCU_USER_INIT SIM_SCGC3 |= SIM_SCGC3_SPI2_MASK;\
                               SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK\
                               | SIM_SCGC5_PORTB_MASK | SIM_SCGC5_PORTC_MASK \
                               | SIM_SCGC5_PORTD_MASK | SIM_SCGC5_PORTE_MASK;
```

d4dtchhw_kinetis_adc_cfg.h:

This file is used to configure ADC settings for touch screen driver.

```
  /*****************************************************************************
  * Constants
  *****************************************************************************/

  #define D4DTCH_ADC_HW D4DTCH_ADC_HW_KINETIS

  #define D4DTCH_ADC_ID 1 // Use ADC module 1

  // X+ wire definition
  #define D4DTCH_X_PLUS   4
  #define D4DTCH_X_PLUS_PORT  GPIOB_PDOR  // Data output register
  #define D4DTCH_X_PLUS_DDR   GPIOB_PDDR  // Output enable register
  #define D4DTCH_X_PLUS_ADCH  10          // ADC channel number
  #define D4DTCH_X_PLUS_PCR   PORTB_PCR4

//#define D4DTCH_X_PLUS_ADCH_PIN_ENABLE  (D4DTCH_X_PLUS_PCR = PORT_PCR_MUX(0)); // Mux ADC
//#define D4DTCH_X_PLUS_ADCH_PIN_DISABLE (D4DTCH_X_PLUS_PCR = PORT_PCR_MUX(1)); // Mux GPIO

  //#define D4DTCH_INIT_X_PLUS  OUTPUT(D4DTCH_X_PLUS); RESET(D4DTCH_X_PLUS);
  //#define D4DTCH_RESET_X_PLUS RESET(D4DTCH_X_PLUS);
  //#define D4DTCH_SET_X_PLUS   SET(D4DTCH_X_PLUS);

  // X- wire definition
  #define D4DTCH_X_MINUS  6
  #define D4DTCH_X_MINUS_PORT  GPIOB_PDOR
  #define D4DTCH_X_MINUS_DDR   GPIOB_PDDR
  #define D4DTCH_X_MINUS_PCR   PORTB_PCR6

  //#define D4DTCH_INIT_X_MINUS  OUTPUT(D4DTCH_X_MINUS); RESET(D4DTCH_X_MINUS);
  //#define D4DTCH_RESET_X_MINUS RESET(D4DTCH_X_MINUS);
  //#define D4DTCH_SET_X_MINUS   SET(D4DTCH_X_MINUS);
  //#define D4DTCH_X_MINUS_HIGH_Z_ENABLE INPUT(D4DTCH_X_MINUS);
  //#define D4DTCH_X_MINUS_HIGH_Z_DISABLE OUTPUT(D4DTCH_X_MINUS);

  // Y+ wire definition
  #define D4DTCH_Y_PLUS   7
  #define D4DTCH_Y_PLUS_PORT  GPIOB_PDOR
  #define D4DTCH_Y_PLUS_DDR   GPIOB_PDDR
```

```
  #define D4DTCH_Y_PLUS_ADCH   13
  #define D4DTCH_Y_PLUS_PCR   PORTB_PCR7

//#define D4DTCH_Y_PLUS_ADCH_PIN_ENABLE  (D4DTCH_Y_PLUS_PCR = PORT_PCR_MUX(0)); // Mux ADC
//#define D4DTCH_Y_PLUS_ADCH_PIN_DISABLE (D4DTCH_Y_PLUS_PCR = PORT_PCR_MUX(1)); // Mux GPIO

  //#define D4DTCH_INIT_Y_PLUS  OUTPUT(D4DTCH_Y_PLUS); RESET(D4DTCH_Y_PLUS);
  //#define D4DTCH_RESET_Y_PLUS RESET(D4DTCH_Y_PLUS);
  //#define D4DTCH_SET_Y_PLUS   SET(D4DTCH_Y_PLUS);

  // Y- wire definition
  #define D4DTCH_Y_MINUS  5
  #define D4DTCH_Y_MINUS_PORT  GPIOB_PDOR
  #define D4DTCH_Y_MINUS_DDR   GPIOB_PDDR
  #define D4DTCH_Y_MINUS_PCR   PORTB_PCR5

  //#define D4DTCH_INIT_Y_MINUS  OUTPUT(D4DTCH_Y_MINUS); RESET(D4DTCH_Y_MINUS);
  //#define D4DTCH_RESET_Y_MINUS RESET(D4DTCH_Y_MINUS);
  //#define D4DTCH_SET_Y_MINUS   SET(D4DTCH_Y_MINUS);
  //#define D4DTCH_Y_MINUS_HIGH_Z_ENABLE INPUT(D4DTCH_Y_MINUS);
  //#define D4DTCH_Y_MINUS_HIGH_Z_DISABLE OUTPUT(D4DTCH_Y_MINUS);

  // definition of calibration cross offset on screen in pixels
  //#define D4DTCH_CALIB_CROSS_OFFSET 30

  // Constant specifying maximum ADC value for a screen touch (=12bits)
  #define D4DTCH_FULL_SCALE          0x0FFF

  // Constants specifying minimum ADC value for a screen touch
  //#define D4DTCH_X_TOUCH_MIN             (D4DTCH_FULL_SCALE / 10)
  //#define D4DTCH_Y_TOUCH_MIN             (D4DTCH_FULL_SCALE / 10)

  //#define D4DTCH_X_TOUCH_OFFMAX             (D4DTCH_X_TOUCH_MIN * 4 / 2)
  //#define D4DTCH_Y_TOUCH_OFFMAX             (D4DTCH_Y_TOUCH_MIN * 4 / 2)

  // Constants specifying ADC difference for touch screen sample
  //#define D4DTCH_SAMPLE_MARGIN           (D4DTCH_FULL_SCALE / 256)
```

You may need to modify these files if you want to port eGUI to your own board.

3.3 Baremetal project

For TWR-K60N512 board, you may find IAR or CW 10.1 baremetal project from \Freescale_embedded_GUI_SW_Official_Demos\EGUI_D4D_Demo\TWR_K60N512\baremetal\. Open and compile the project (default LCD configuration is for SPI connection, you can change the configuration in d4d_user_cfg.h to support Flexbus connection for LCD).

NOTE

Please remove C5 (for Rev. B or previous version) or C2 (for Rev. C) on the left top corner of the board. If you are using Rev.D or later version, please remove the jumper on J16.

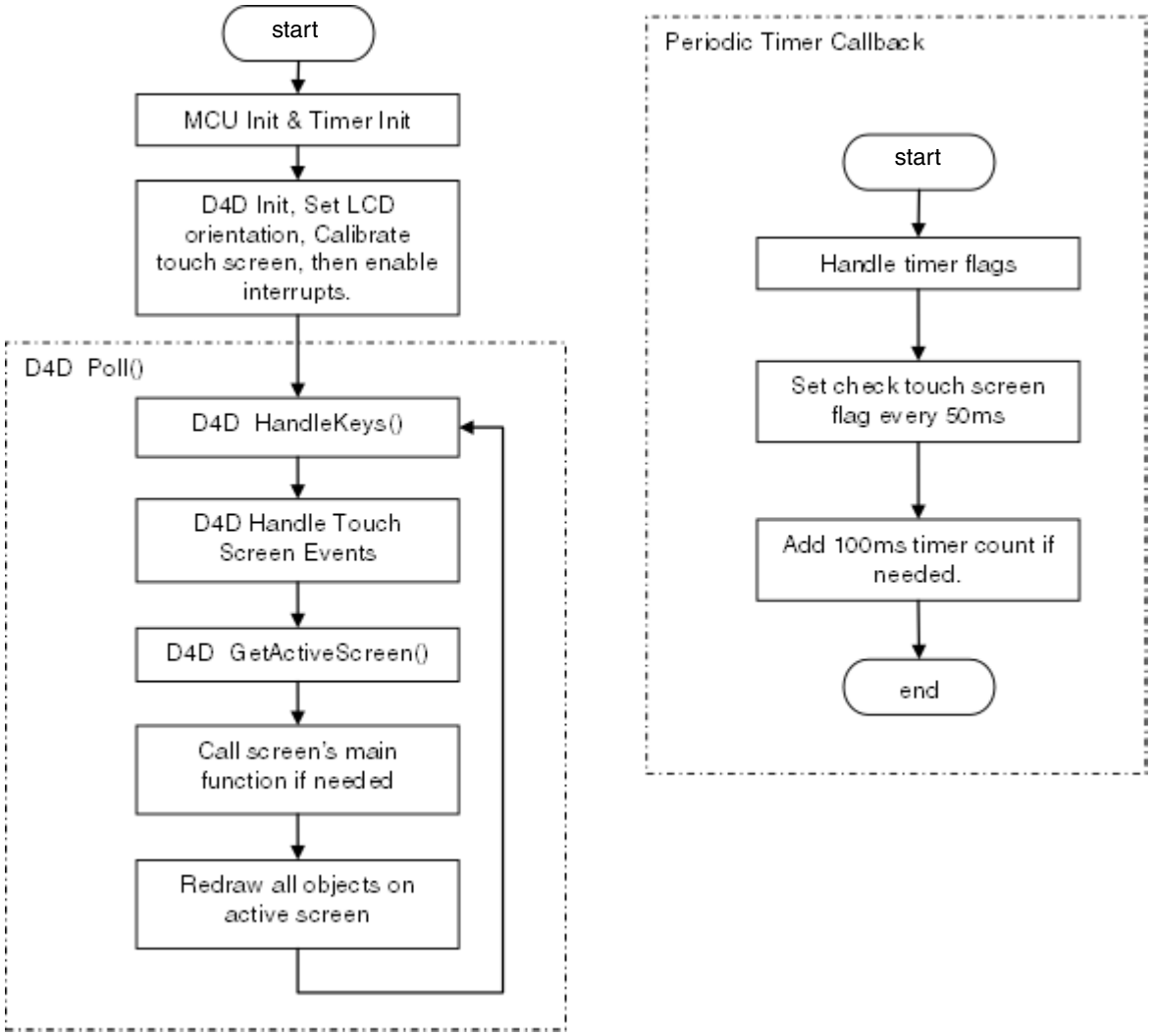

Figure 11. eGUI main flowchart

As Figure 11 shows, the main loop of eGUI is *D4D_Poll()* and periodic timer callback is necessary for main loop. *D4D_Poll()* handles keys, touch screen, or timer events and redraws objects on an active screen.

So the applications of eGUI will have the same main loop like below:

```
void main (void)
{

  MCU_Init();   // MCU Initialization Clock, WatchDog etc
  Timer_Init(); // Periodic Timer interrupt initialization - 25ms

  D4D_Init(&screen_entry);
  D4D_SetOrientation(D4D_ORIENT_LANDSCAPE);
  D4D_CalibrateTouchScreen();
  EnableInterrupts; /* enable interrupts */

  for(;;) {
```

```
    D4D_Poll(); // D4D poll loop

  } /* loop forever */
  /* please make sure that you never leave main */
}
```

Please refer to the eGUI reference manual available at www.freescale.com/egui for more details.

3.4 MQX project

For the TWR-K60N512 board, to compile an eGUI MQX project you must install MQX 3.6 or 3.7 first. Then, please make sure that you have to define *BSPCFG_ENABLE_IO_SUBSYSTEM*, *BSP_DEFAULT_IO_CHANNEL*, *BSPCFG_ENABLE_ADC1* and *BSPCFG_ENABLE_SPI2* (if you want to use SPI for LCD connection) to 1 in user_config.h of MQX BSP and PSP projects. And you must recompile MQX BSP and PSP projects to generate the new MQX library for eGUI, or your eGUI project won't work properly.

For MQX project, we create two tasks for eGUI demo:

```
const TASK_TEMPLATE_STRUCT  MQX_template_list[] =
{
   /* Task Index,Function,Stack,Priority, Name, Attributes,Param,Time Slice */
    { LCD_TASK,  lcd_task,  3000, 9,  "LCD",  MQX_AUTO_START_TASK, 0,     0 },
    { TIME_TASK, Time_task, 1500, 10, "time", 0,                   0,     0 },
    { 0,         0,         0,    0,  0,      0,                   0,     0 }
};
```

For the LCD task, the main loop is totally the same as the one in the baremetal project. The difference is that it needs to open the ADC as an I/O device before it can be used for touch screen simulation and we need to create a timer task before *D4D_Init()*. However, *_time_delay()* is necessary to switch the active task to timer task for timer counting. So the mail loop of the MQX project looks like:

```
   for(;;)
   {
          D4D_Poll();
          _time_delay(10);
   }
```

3.5 Screen shots

After you successfully program the baremetal or MQX version of the official eGUI demo to your controller module board, you will see a display on TWR-LCD as Figure 12 shows.

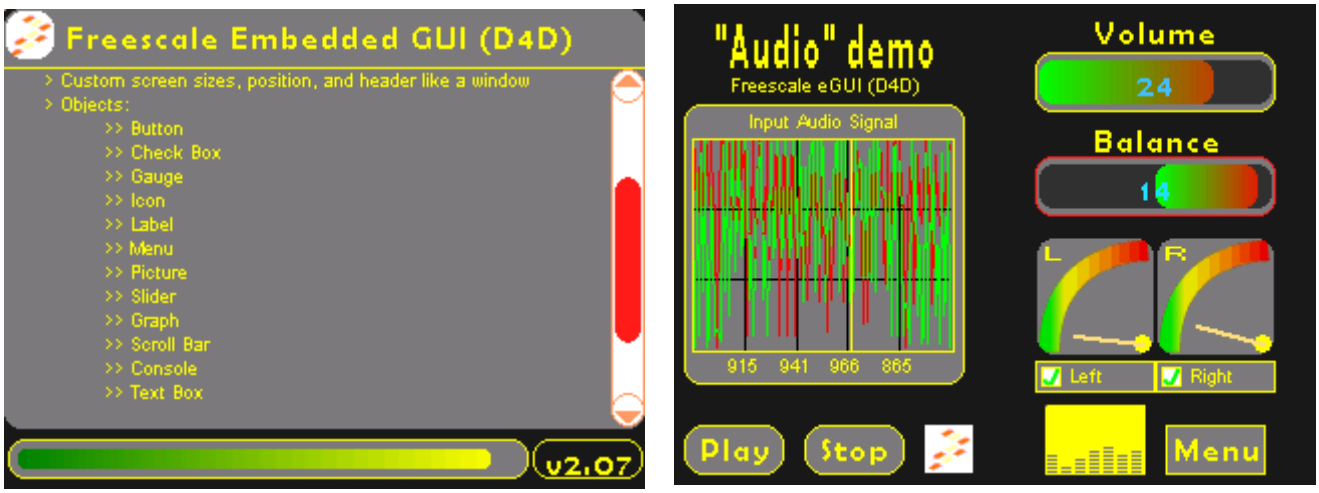

Figure 12. Official eGUI demo screen shots

4 Display microWindows with TWR-MCF5441X

Freescale has a TWR-MCF5441X board that can be used to run Linux. Freescale Linux BSP engineers have made it possible to display microWindows on TWR-LCD board.

The SSD1289, the LCD controller on the TWR-LCD board, has a display data buffer. This makes it convenient for those MCUs that don't have large RAM areas to display graphics for a user interface. But the disadvantage is that the user has to create a thread to update the onboard display data buffer periodically to simulate a frame buffer—this lowers the performance of whole system. However, by doing this, it really works.

You may find such code (thread to update display data buffer) below in /drivers/video/fsl-ssd1289-fb.c (for the latest LTIB ISO image, go to www.freescale.com and search "MCF5441X"):

```
static int ssd1289fbd(void *arg)
{
        struct fb_info *info = arg;
        int i;
        unsigned short *buf_p;

        while (!kthread_should_stop()) {
                set_current_state(TASK_INTERRUPTIBLE);
                ssd1289_write(info, SSD1289_REG_H_RAM_ADR_POS, 0);
                ssd1289_write(info, 0xef00, 1);

                ssd1289_write(info, SSD1289_REG_V_RAM_ADR_START, 0);
                ssd1289_write(info, 0x0000, 1);

                ssd1289_write(info, SSD1289_REG_V_RAM_ADR_END, 0);
                ssd1289_write(info, 0x013f, 1);

                ssd1289_write(info, SSD1289_REG_GDDRAM_X_ADDR, 0);
                ssd1289_write(info, 0x00ef, 1);

                ssd1289_write(info, SSD1289_REG_GDDRAM_Y_ADDR, 0);
                ssd1289_write(info, 0x0000, 1);
```

```
                ssd1289_write(info, SSD1289_REG_GDDRAM_DATA, 0);

                buf_p = (unsigned short *)(info->screen_base + 1);

                for (i = 0; i < info->screen_size; i += 2)
                        ssd1289_write(info, *(buf_p++), 1);

                schedule_timeout(HZ/25);
}
```

To enable TWR-LCD frame buffer driver for TWR-MCF5441X, you may need to read the help file from the BSP ISO package: \help\documents\html\M54418TWR_LCD.htm. For more detailed information, please read the LTIB help file, the user manual, and the quick start guide from the M54418 Tower Linux BSP ISO.

NOTE

Because the ADC_IN pins of MCF5441X cannot be configured as a GPIO port, the touch screen on the TWR-LCD board cannot be used when connected to the TWR-MCF5441X.

5 Summary

The TWR-LCD board can be connected through the Flexbus/mini-Flexbus interface or SPI/DSPI/QSPI interface to display something onscreen. And it is not complicated to simulate a touch screen interface with two ADC channels (which could be multiplexed as GPIO pins) and two GPIO pins.

Freescale eGUI supports most Freescale Coldfire, Coldfire+, and Kinetis Tower controller boards. Freescale eGUI can be used standalone or integrated into the MQX operating system. Freescale plans to release new versions of eGUI to support more LCD controllers, including mono LCD controllers. It will also support multiple languages, including Chinese, Korean, Japanese, and so on. And, eGUI is totally FREE!

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN4452
Rev. 0
1/2012

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.
All other product or service names are the property of their respective owners.
© Freescale Semiconductor, Inc. 2012. All rights reserved.