Freescale Semiconductor
应用笔记

Document Number: AN4452
Rev 0, 05/2012

利用 TWR-LCD 显示(eGUI)
采用 Freescale 塔式系统

作者: Ju Yingyi
应用工程师，AISG

内容

1 简介 1
2 硬件接口 1
2.1 TWR-LCD 板 1
2.2 用于 LCD 显示的 Flexbus/mini-Flexbus 接口 3
2.3 用于 LCD 显示的 SPI 7
2.4 触摸屏界面 8
3 在 TWR-LCD 板上显示 eGUI 10
3.1 CW10.1 和 IAR 项目 11
3.2 eGUI 配置 12
3.3 裸机项目 15
3.4 MQX 项目 17
3.5 屏幕截图 17
4 利用 TWR-MCF5441X 显示 microWindows 18
5 摘要 18

1 简介

本应用笔记介绍如何利用 TWR-LCD 板显示 eGUI 或其他图形用户界面。本文档适用于集成 Flexbus 或 SPI 接口的所有 Freescale 的 ColdFire、Kinetis 32 位 MCU 和 MPU 塔式卡。

Freescale 不断推出全新 32 位器件，每款器件均集成 Flexbus 或 mini-Flexbus 和 SPI/QSPI/DSPI 接口，可用于连接 Freescale TWR-LCD 板。本文档介绍如何配合使用这些全新的塔式板和 TWR-LCD 板来显示 eGUI 或其他 GUI (microWindows)。下文内容显示 MCU 板与 TWR-LCD 板之间的硬件连接，以及驱动 LCD 模块的 eGUI 软件包，包括裸机 eGUI 版本和运行 MQX™的 eGUI 版本。内容涵盖大多数现有的 ColdFire V1、V2 和 V4 MCU/MPU 塔式卡、ColdFire+ MCU 塔式卡和 Kinetis 塔式卡。

有关 Freescale 塔式系统的更多详情，请参考 http://www.freescale.com/tower

有关 Freescale 嵌入式图形用户界面(eGUI)的更多详情，请参考 http://www.freescale.com/egui

2 硬件接口

© 2012 Freescale Semiconductor, Inc.

2.1 TWR-LCD 板

TWR-LCD 板上的 TFT-LCD 模块集成了 Solomon Systech TFT-LCD 控制器 SSD1289。该 TFT-LCD 控制器在单芯片内集成图形显示数据 RAM、电源电路、栅极驱动器和源极驱动器。本电路板上的图形显示数据 RAM 通过 16 位 6800 系列/8080 系列兼容的并行接口或 SPI 实现与通用 MCU/MPU 的连接。有关 SSD1289 的更多信息，请参考 AN4153: “在 MCF51MM 系列上使用带 TWR-LCD 的 Freescale eGUI”。

TWR-LCD 板的原理图可从 http://cache.freescale.com/files/soft_dev_tools/hardware_tools/schematics/TWRLCDSCH.pdf?fpsp=1 下载。

2.1.1 SW1 设置

适用于 16 位 FlexBus/mini-Flexbus 接口的 SW1 设置:

- SW1[1]: 开，PS2 = 0
- SW1[2]: 关，PS0 = 1
- SW1[3]: 开，禁用板上的 MCF51JM128 控制
- SW1[4]: 用于 microSD 卡接口，无关
- SW1[5]: 选择 SPI 通道，该模式下无关
- SW1[6]: TP_SEL，该模式下无关，建议开
- SW1[7]: 开/关，背灯开/关
- SW1[8]: 可选，用户可让 MCU/MPU 塔式板输出 PWM 波形来控制板上的蜂鸣器

适用于 SPI 接口的 SW1 设置:

- SW1[1]: 关，PS2 = 1
- SW1[2]: 开，PS0 = 0
- SW1[3]: 开，禁用板上的 MCF51JM128 控制
- SW1[4]: 用于 microSD 卡接口，无关
- SW1[5]: 选择 SPI 通道，若为开则选择 SPI 通道 0，若为关则选择 SPI 通道 1
- SW1[6]: TP_SEL，该模式下无关，建议开
- SW1[7]: 开/关，背灯开/关
- SW1[8]: 可选，可让 MCU/MPU 塔式板输出 PWM 波形来控制板上的蜂鸣器

由于 TWR-LCD 板上焊接了 MCF51JM128，它能采用 MCF51JM128 独立工作。

适用于（MCF51JM128，SPI）独立工作的 SW1 设置:

- SW1[1]: 关，PS2 = 1
- SW1[2]: 开，PS0 = 0
- SW1[3]: 必须关，由 MCF51JM128 控制
- SW1[4]: 用于 microSD 卡接口，无关
- SW1[5]: 选择 SPI 通道，该模式下无关
- SW1[6]: 必须关
- SW1[7]: 开/关，背灯开/关
- SW1[8]: 此模式下无关

2.1.2 SW5 设置

该 DIP 开关用于触摸屏界面。如需连接任意 MCU/MPU 板，可打开所有开关。单独使用时，建议关断所有开关。

2.1.3 SW3 设置

复位按钮。它不仅复位板上的 MCF51JM128，还复位整个塔式系统。

2.1.4 SW4 设置

该按钮用于板上的 MCF51JM128 USB 引导加载程序。要进入 USB 引导加载程序模式，可先将 SW1 设为“适用于 (MCF51JM128，SPI) 独立工作的 SW1 设置”，随后用 USB 电缆使 J7 与 PC 相连，并持续按压 SW4，最后按下 SW3 复位该板。如果顺利进入引导加载程序模式，则 LCD 将在屏幕左上方显示“Bootloader mode: waiting for S19 file…”，并且 PC 上将会出现“USB 大容量存储设备已安装”，如下图图 1 所示。

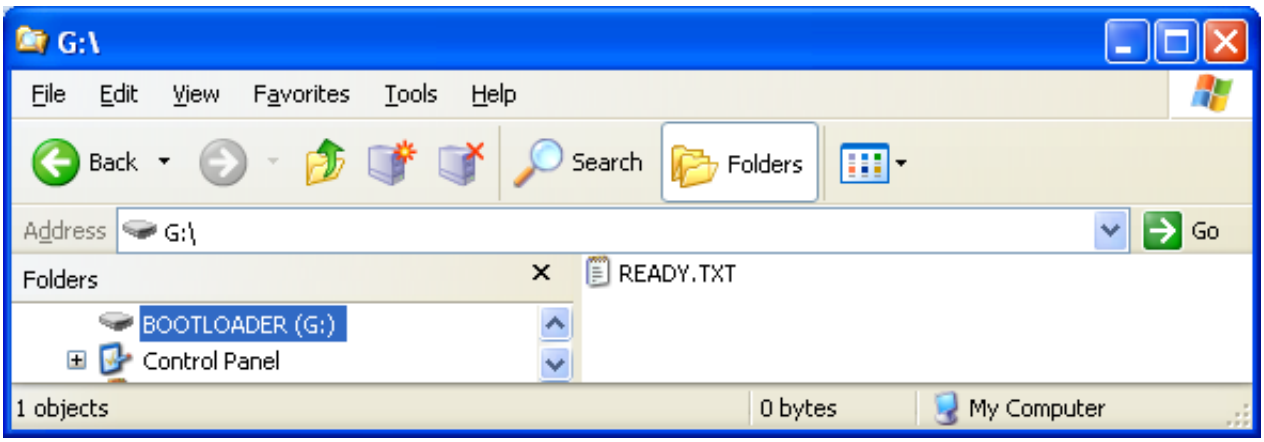


图 1. 引导加载程序盘

如需更新板上的 MCF51JM128 固件，则可将.S19 文件拖放至该盘，就像复制文件那样操作。在 TWR-LCD 板入门 CD 中可找到有关 USB 引导加载程序及其源代码的更多信息。

2.1.5 J3 设置

该跳线用于对板上的 MCF51JM128 进行编程和调试。然而，如果擦除了原始固件或无法从 USB 引导加载程序，则可使用 P&E USB BDM Multilink 或 P&E USB Multilink Universal 对固件重新进行编程。

2.2 用于 LCD 显示的 Flexbus/mini-Flexbus 接口

2.2.1 TWR-LCD 端

TWR-LCD 板上的 Flexbus/mini-Flexbus 接口配置为 16 位数据总线模式。

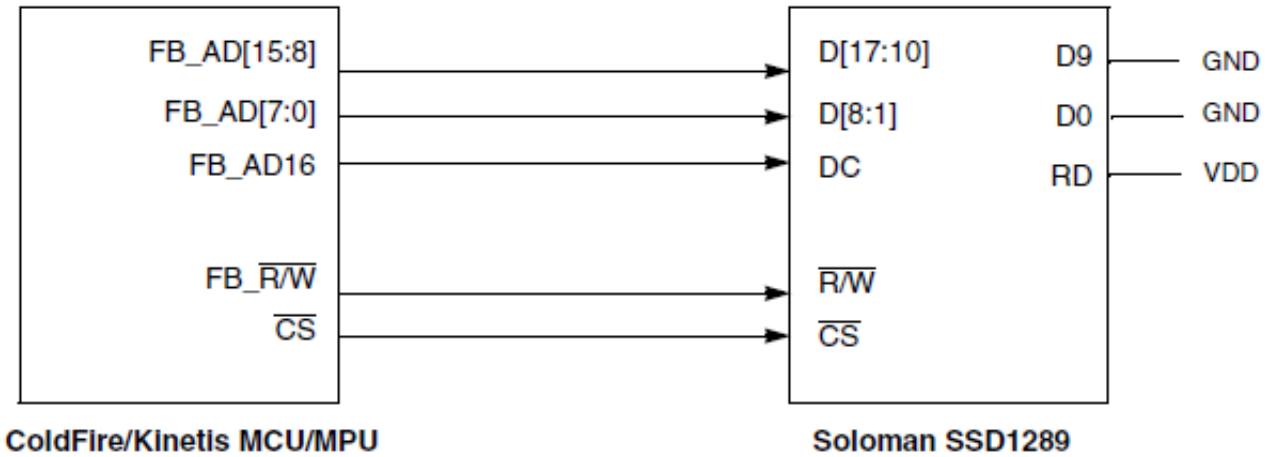


图 2. TWR-LCD 板上的 Flexbus/mini-Flexbus 接口

由于在本设计中 SSD1289 DC 信号连接至 FB_AD16，因此当 FB_AD16 为低电平时，可访问 SSD1289 的变址寄存器。FB_AD16 为高电平时，可访问 SSD1289 的控制寄存器或显示数据。

例如，假定 MCU 的 $\overline{CS0}$ 连接 SSD1289 的 $\overline{CS}$，且 MCU 侧的 CSAR0 已设为 0x400000。这种情况下，地址 0x400000 用来访问 SSD1289 的变址寄存器，而地址 0x410000 可用来访问 SSD1289 的控制寄存器或显示数据。

2.2.2 控制器端

使用 Flexbus/mini-Flexbus 接口在 LCD 上显示内容时，需留意一些重要事项。深入研究 LCD 底层驱动之前，请阅读下列内容。

2.2.2.1 地址/数据总线多路复用或非多路复用模式

首先，让我们来比较 TWR-MCF51CN 和 TWR-K60N512 的原理图。请参考下图。

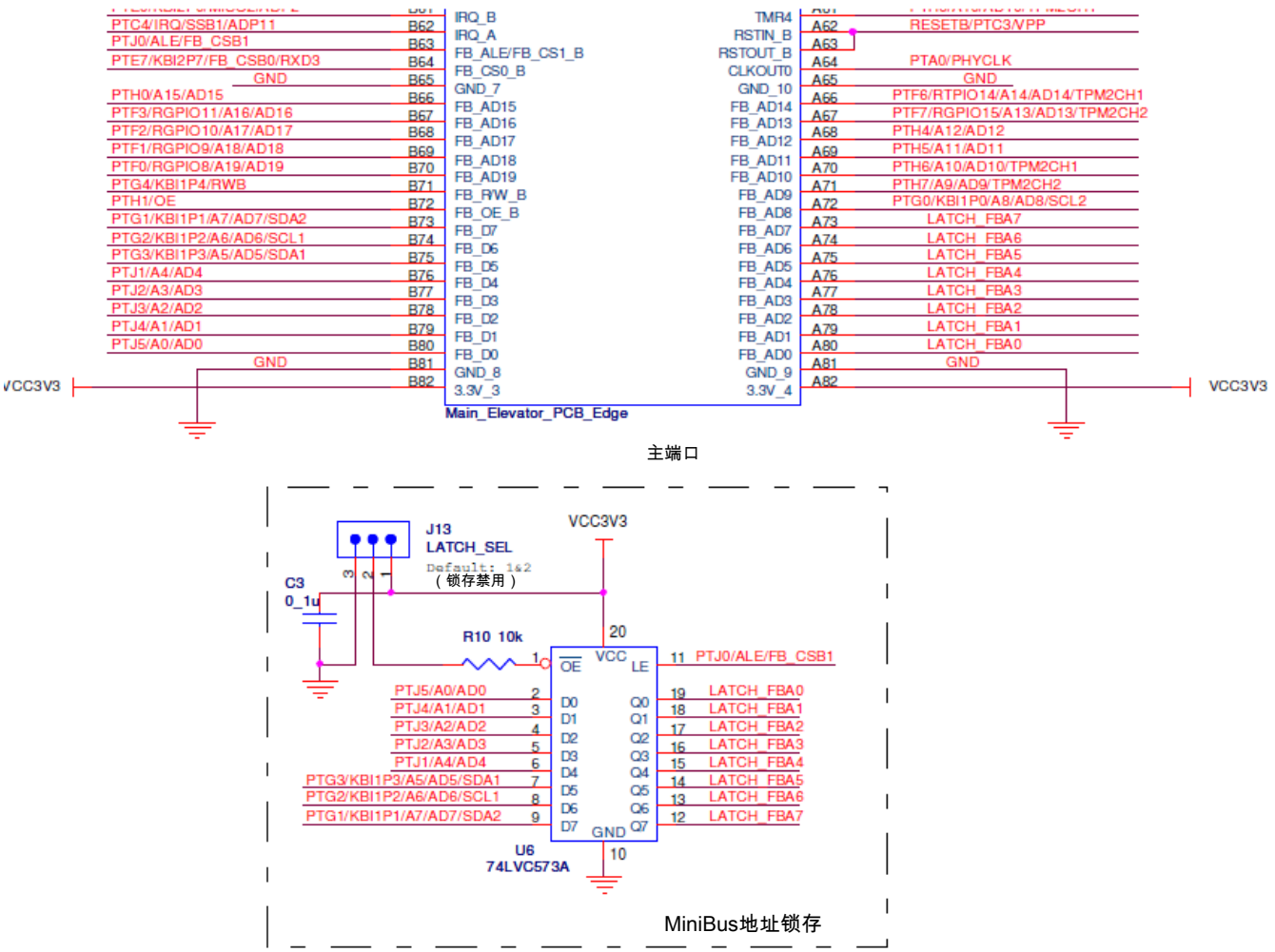


图 3. TWR-MCF51CN 的 Mini-Flexbus 接口

TWR-MCF51CN 板上使用了锁存芯片。使用该芯片是因为 TWR-MCF51CN 的 mini-Flexbus 接口设计成地址/数据总线多路复用模式。

图 4. TWR-K60N512 的 Flexbus 接口

TWR-K60N512 板上无锁存芯片，因为 TWR-K60N512 的 Flexbus 接口设计为地址/数据总线非多路复用模式（连接 TWR-MEM 卡时）。

由于 TWR-LCD 板上的数据总线连接 FB_AD[15:0]（或 EBI_AD[15:0]），因此无需 B73–B80 上的信号（这些信号仅用于地址/数据总线非多路复用模式下）。但在使用 TWR-MCF51CN 时，用户必须注意 FB_ALE 信号。在整个数据访问周期中，FB_ALE 必须始终保持高电平。因此，ALE 信号可配置为 GPIO 输出引脚，并且始终输出高电平。如果 FB_ALE 在数据访问周期期间工作正常，则 FB_AD[7:0]上的数据无法通过锁存芯片(74LVC573A)，因为当 Flexbus/mini-Flexbus 需锁存来自 LCD 控制器的数据时，FB_ALE 处于低电平状态。

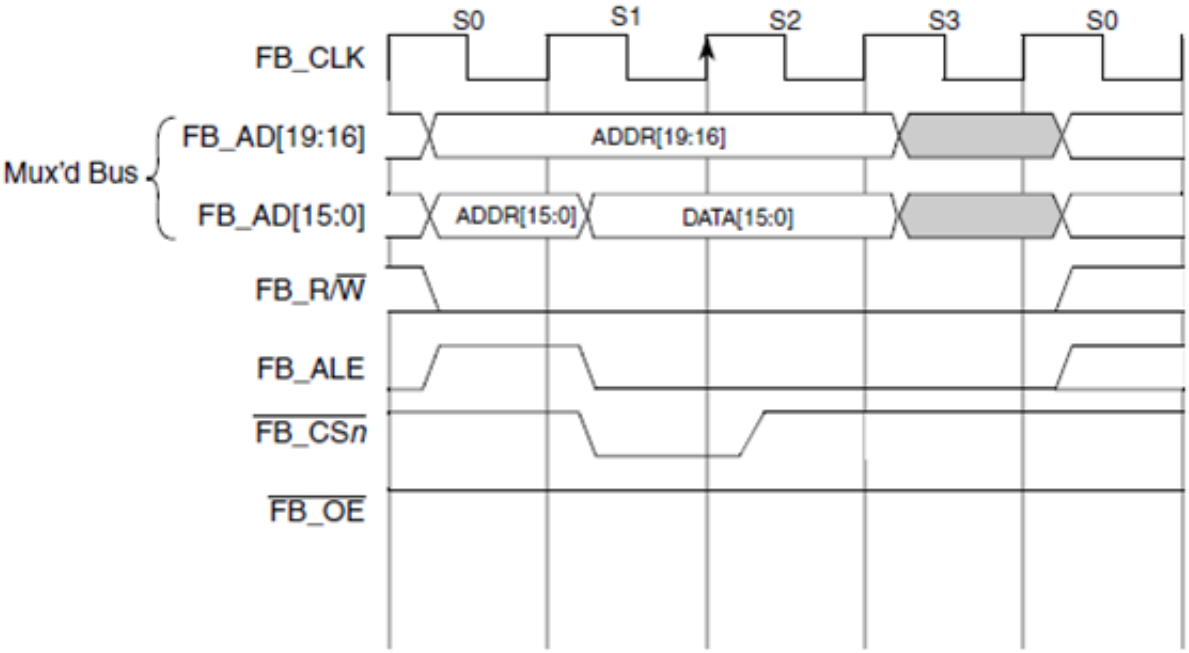


图 5. Flexbus/mini-Flexbus 单次字写入传输

目前为止，由于引脚多路复用的使用量巨大，Freescale 多款塔式卡的 Flexbus/mini-Flexbus 接口均设计用于地址/数据总线多路复用模式，且板上带有地址锁存芯片：

- TWR-MCF51CN
- TWR-MCF51JF
- TWR-MCF51QM
- TWR-K40X256
- TWR-K53N512

2.2.2.2 数据字节对齐

想要通过控制器模块在 TWR-LCD 板上显示某些内容时，请记得根据物理连接设置正确的数据字节对齐（如果 MCU/MPU 的 Flexbus/mini-Flexbus 控制器具有 CSCRn[BLS]位）。

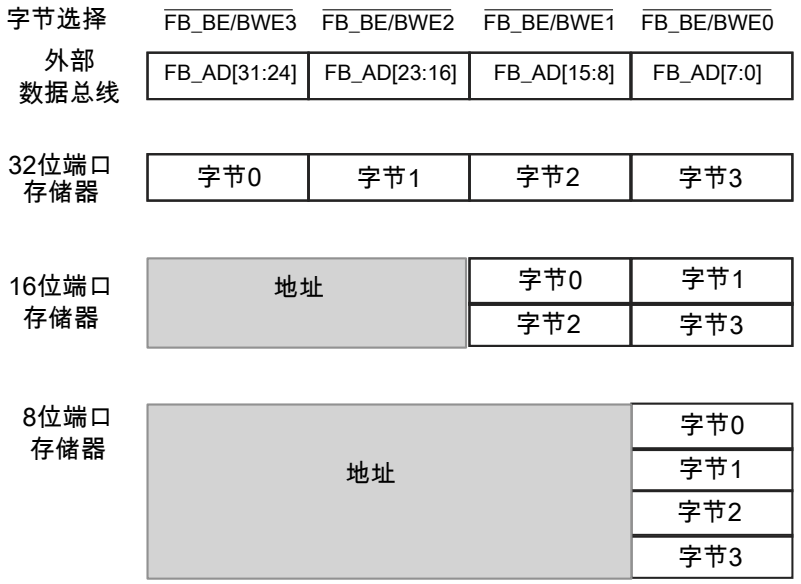


图 6. 连接外部存储器端口尺寸(CSCRn[BLS] = 1)

一般而言，ColdFire V1 和 ColdFire+ MCU 仅提供 mini-Flexbus 接口，不提供该控制位。而 ColdFire V4 和 Kinetis MCU 具有 Flexbus 接口，提供该控制位。必须将其置 1 才能用于 TWR-LCD 板。

2.2.2.3 等待状态

根据 SSD1289 数据手册，最小写时钟周期为 100 ns。单个读/写 Flex 总线周期中至少有 4 个 FB_CLK 周期。因此，FB_CLK 不可超过 40 MHz，否则必须插入适当的等待周期。

例如:

对于 TWR-MCF51CN/TWR-MCF51JE/TWR-MCF51MM/TWR-MCF51JF/TWR-MCF51QM，若系统时钟 = 50 MHz、FB_CLK = 25 MHz，则无需等待状态。

对于 TWR-K40X256/TWR-K53N512/TWR-K60N512，若系统时钟 = 100 MHz、FB_CLK = 50 MHz，则需 1 个等待状态。

对于 TWR-MCF5441X，若系统时钟 = 250 MHz、FB_CLK = 1/4 系统时钟 = 62.5 MHz，则需 3 个等待状态。将 MISCCR2[FBHALF]置 0 可更改 FB_CLK = 1/2 系统时钟；此时需 9 个等待状态。

2.2.2.4 基地址设置

TWR-LCD 的基地址可通过设置 CSAR*n* 和 CSMR*n* 来初始化。请先阅读相关 MCU/MPU 参考手册中的“存储器映射”部分，确定基地址。不同器件的片外扩展存储器区域各不相同。

- ColdFire V1 和 ColdFire+ MCU 能使用 0x0040_0000–0x007FF_FFFF 和 0x00A0_0000–0x00BFF_FFFF
- ColdFire V2 MCU (MCF5225X)能使用 0x8000_0000–0xFFFF_FFFF
- ColdFire V4 MPU (MCF5441X)能使用 0x0000_0000–0x3FFF_FFFF()和 0xC000_0000–0xDFFF_FFFF
- Kinetis 系列 MCU 能使用 0x6000_0000–0xDFFF_FFFF

在控制器模块上，切勿设置超出 MCU/MPU 范围的地址值。另外，即使将其他 $\overline{CS}$ 连接到了 TWR-LCD 板，也要记得将 CSMR0[V]置 1 以使能全局片选。

2.3 用于 LCD 显示的 SPI

由于 PS3 和 PS1 在 TWR-LCD 板上固定为高电平，用户可以使用四线式 SPI 模式（PS3 = PS2 = PS1 = 高电平，PS0 = 低电平）或三线式 SPI 模式（PS3 = PS2 = PS1 = PS0 = 高电平）。

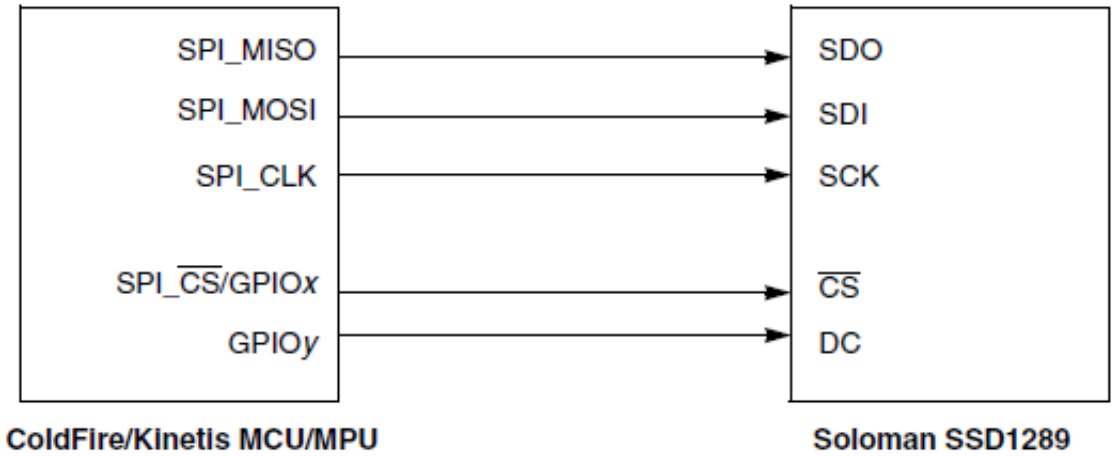


图 7. TWR-LCD 上的 SPI 接口

2.3.1 四线式 SPI 模式

在此模式下，TWR-LCD 侧需要 SDI、SCK、$\overline{CS}$ 和 DC。SDI 在每个 SCK 上升沿移位至 8 位移位寄存器中，顺序是 MSB 优先。DC 每八个时钟采样一次，以确定移位寄存器中的数据字节在相同时钟时是写入命令寄存器还是显示数据 RAM。然而，在 MCU/MPU 侧，$\overline{CS}$ 可通过 SPI_$\overline{CS}$ 或 GPIO 引脚控制，而 DC 必须通过 GPIO 引脚控制。

2.3.2 三线式 SPI 模式

相比四线式 SPI 模式，TWR-LCD 侧不使用 DC 信号。每过 9 个时钟，就会有总共 9 位依次移入移位寄存器，顺序为：DC 位、D7 至 D0 位。DC 位将决定的是移位寄存器中的后续数据字节是写入命令寄存器还是显示数据 RAM 中。在 MCU/MPU 侧，$\overline{CS}$ 可通过 SPI_$\overline{CS}$ 或 GPIO 引脚控制。

注

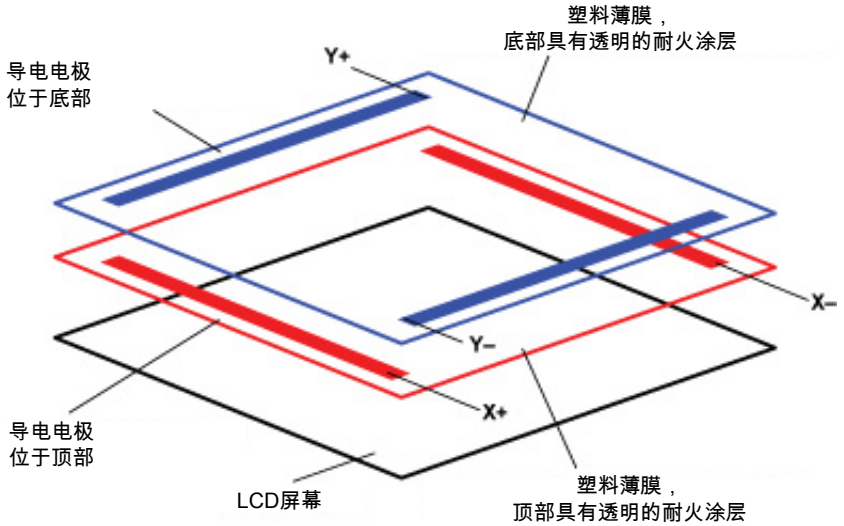
目前发布的 eGUI 包不支持三线式 SPI 模式。

如果在 MCU/MPU 侧使用 16 位 SPI 模式，那么建议传输数据两次。例如，如果要将数据 0x9290 发送至寄存器 0x03，则数据格式应当为 0303 (index of command register)、9292（高位字节）、9090（低位字节）。更多详情，请参考 http://www.solomon-systech.com 处提供的 SSD1289 应用笔记

SSD1289 的 SPI 时钟速度有所限制。其最大频率为 13 MHz。因此，用户必须为塔式控制器模块的 SPI 模块设置正确的波特率(≤13 MHz)。

2.4 触摸屏界面

TWR-LCD 板上集成了覆盖触摸屏的 LCD 屏幕。该屏幕是一个四线电阻式触摸屏。图 8 是触摸屏结构和工作原理的基本示意图。



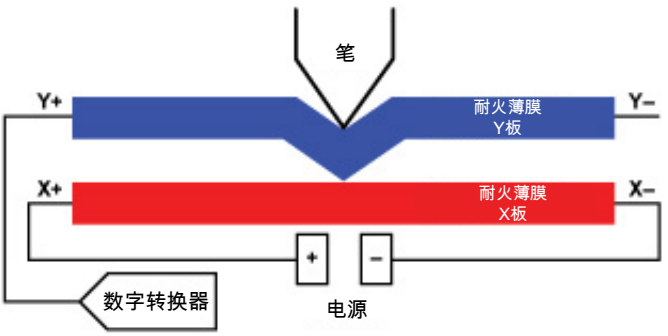


图 8. 触摸屏结构和工作原理的基本示意图

触摸屏由两块塑料薄膜组成，每一块薄膜都覆盖一层金属导电层（通常是铟锡氧化物（ITO）），中间被气隙隔开。上图中的 X 板由电源电压提供激励。触摸屏幕时，两块导电板相互靠近，沿 X 板建立电阻分压器。接触点处的电压表示 X 板位置，通过 Y+电极感测，如图 9 所示。随后激励 Y 板并通过 X+电极感测 Y 位置可重复该过程。

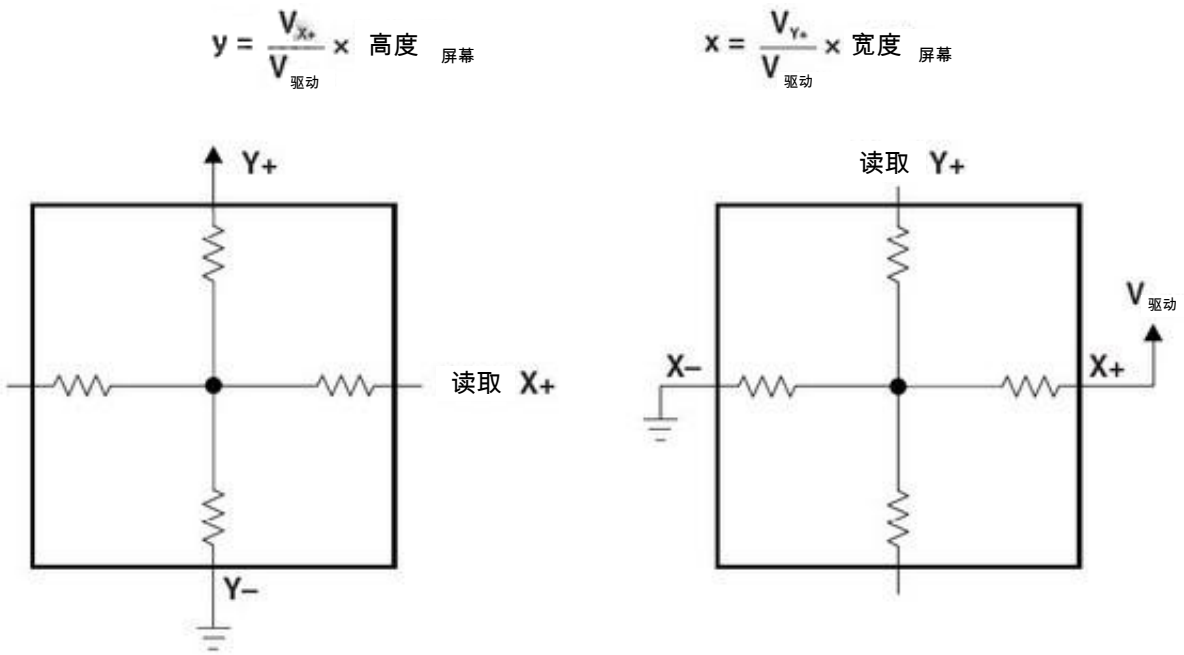


图 9. 位置测量

因此，我们能通过 GPIO 引脚多路复用的两个 ADC 通道以及两个 GPIO 引脚来模拟触摸屏控制器。

根据图 9，如要读取 X 位置:

1. 驱动 X+至高电平并驱动 X-至低电平，将 Y-设为高阻态（可设为输入端口）。
2. 将 Y+设为 ADC 通道，然后便可从 Y+读取原始 X 电压值。

如要读取 Y 位置:

1. 驱动 Y+至高电平并驱动 Y-至低电平，将 X-设为高阻态（可设为输入端口）。
2. 将 X+设为 ADC 通道，然后便可从 X+读取原始 Y 电压值。

此处仅举例说明。用户还可使用 Y-/X-来读取 X/Y 位置。但请记住，需将另一侧(Y+/X+)设为高阻态，以免读取无效值。

获得原始电压值后，X/Y 位置便可通过图 9 中的等式计算得到。

3 在 TWR-LCD 板上显示 eGUI

本节介绍如何通过配置 IAR 或 CW 10.1 在 TWR-LCD 上显示 eGUI

最新发布的 eGUI 可通过访问 http://www.freescale.com/egui 获取。下表可在 eGUI 发布包中找到 (\Freescale_embedded_GUI_SW_Official_Demos\EGUI_D4D_Demo\):

电路板名称	MCU类型	LCD控制器	接口				裸机			MQX 3.6			MQX 3.7		
			串行	并行											
			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	帧缓冲区													
TWR-K40X256	Kinetis	SSD1289													
TWR-K60N512	Kinetis	SSD1289													
TWR-MPC5125	MPC	帧缓冲区													
DEMOQE_HCS08QE128	HCS08	SSD1289													
DEMOQE_MCF51QE128	ColdFire V1	SSD1289													

图例	
选项已实现 (in rel. 2.1)	
选项可实现	
MQX不支持	
不适用	

图 10. eGUI 选项

下列内容重点关注如何通过正确配置 IAR 或 CW 10.1 项目在 TWR-LCD 上显示 eGUI。以 TWR-K60N512 项目为例。

开始前:

- 请阅读“2.1 TWR-LCD 板”或\Freescale_embedded_GUI_SW_Official_Demos\EGUI_D4D_Demo\readme.txt，以便了解跳线/开关设置。
- 如果您使用的是 TWR-K60N512 修订版 C 或更早版本的板卡，请移除 C5（修订版 A 或修订版 B），或移除 C2（修订版 C）。该电容会影响 Flexbus 信号(FB_AD9)。如果您使用的是 TWR-K60N512 修订版 D，则请移除 J16 上的跳线。

3.1 CW10.1 和 IAR 项目

要打开 CW 10.1 项目，请先运行 CW 10.1，然后单击 File --> Import…

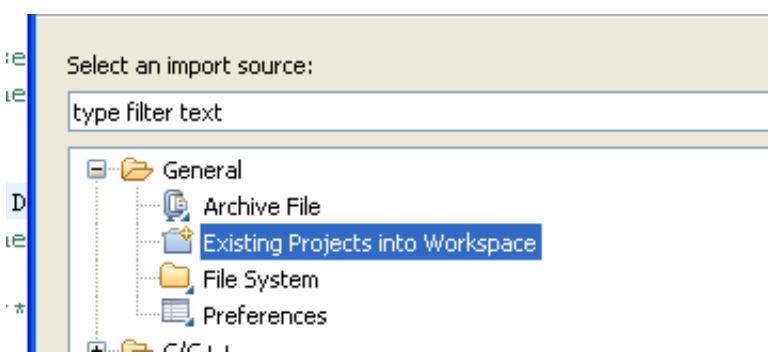


图 11. 如何打开 CW10.1 Project 1

选择“Existing Projects into Workspace”，然后单击“Next”:

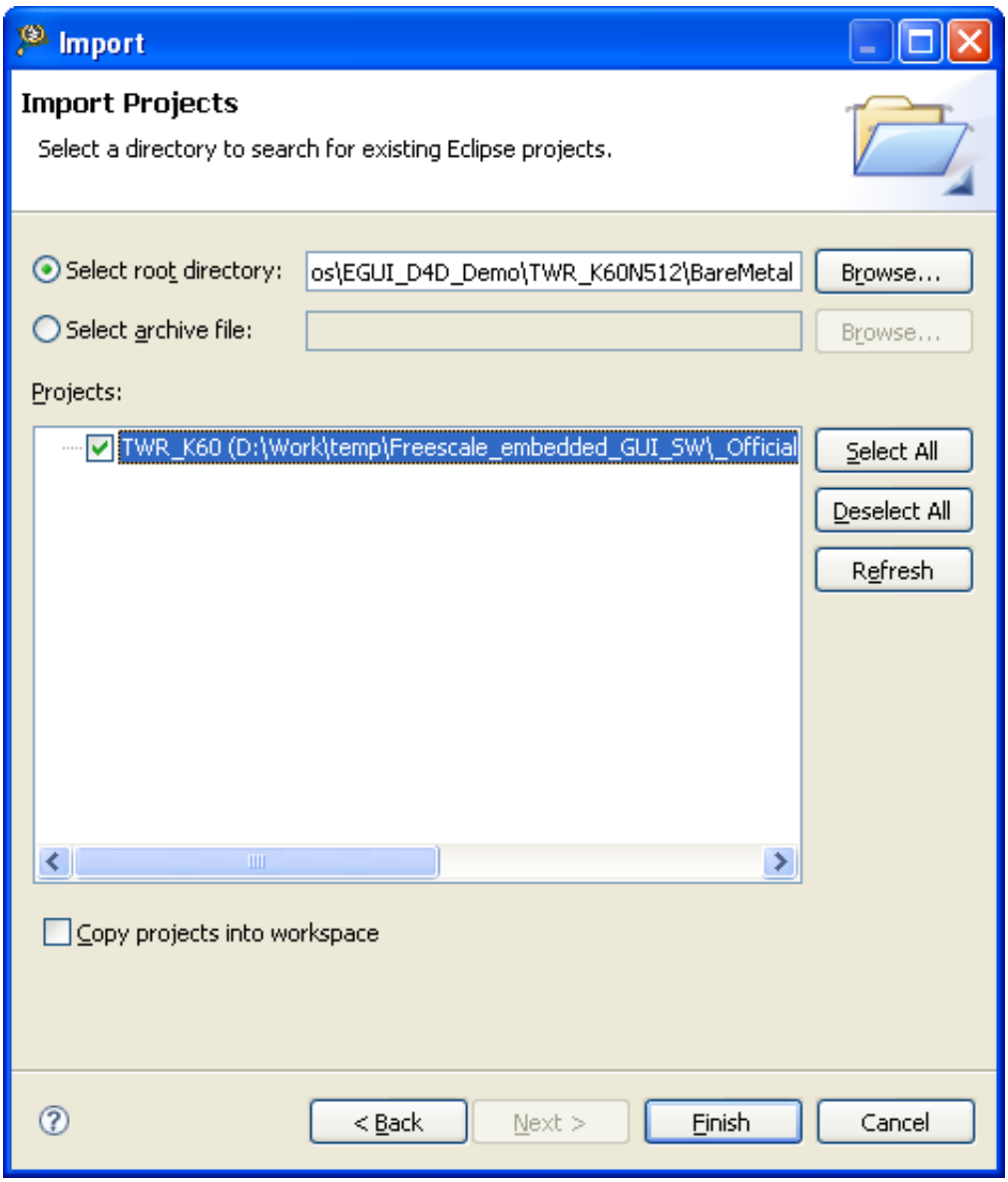


图 12. 如何打开 CW10.1 Project 2

单击“Browse…”选择根目录，随后就能找到要打开的项目。单击“Finish”。

要打开 IAR 项目，只需双击 IAR_6_1 文件夹中的“project.eww”即可。

3.2 eGUI 配置

编译前，应根据 TWR-LCD 板和控制器模块的设置配置 LCD 和触摸屏驱动程序。

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
```

然而，除了底层驱动程序定义外，可能还需修改其他软件配置，以满足你的个人需要。

d4dlcdhw_flexbus_16b_cfg.h 和 **d4dlcdhw_kinetis_spi_cfg.h**:

这两个头文件用于配置 LCD 驱动程序的 Flexbus 和 SPI 硬件设置。

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; ORTC_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 does not 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:

该文件用于配置触摸屏驱动程序的 ADC 设置。

```
  /*****************************************************************************
  * 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)
```

若要将 eGUI 移植到你自己的板，则可能需修改这些文件。

3.3 裸机项目

对于 TWR-K60N512 板，可在\Freescale_embedded_GUI_SW_Official_Demos\EGUI_D4D_Demo\TWR_K60N512\baremetal\目录下找到 IAR 或 CW 10.1 裸机项目。打开并编译项目 (默认 LCD 配置供 SPI 连接使用，您可更改 d4d_user_cfg.h 中的配置，以支持 Flexbus 连接)。

注

请移除 C5（适用于修订版 B 或更早版本），或移除 C2（适用于修订版 C），位置是板卡左上角。如您使用的是修订版 D 或更新版本，请移除 J16 上的跳线。

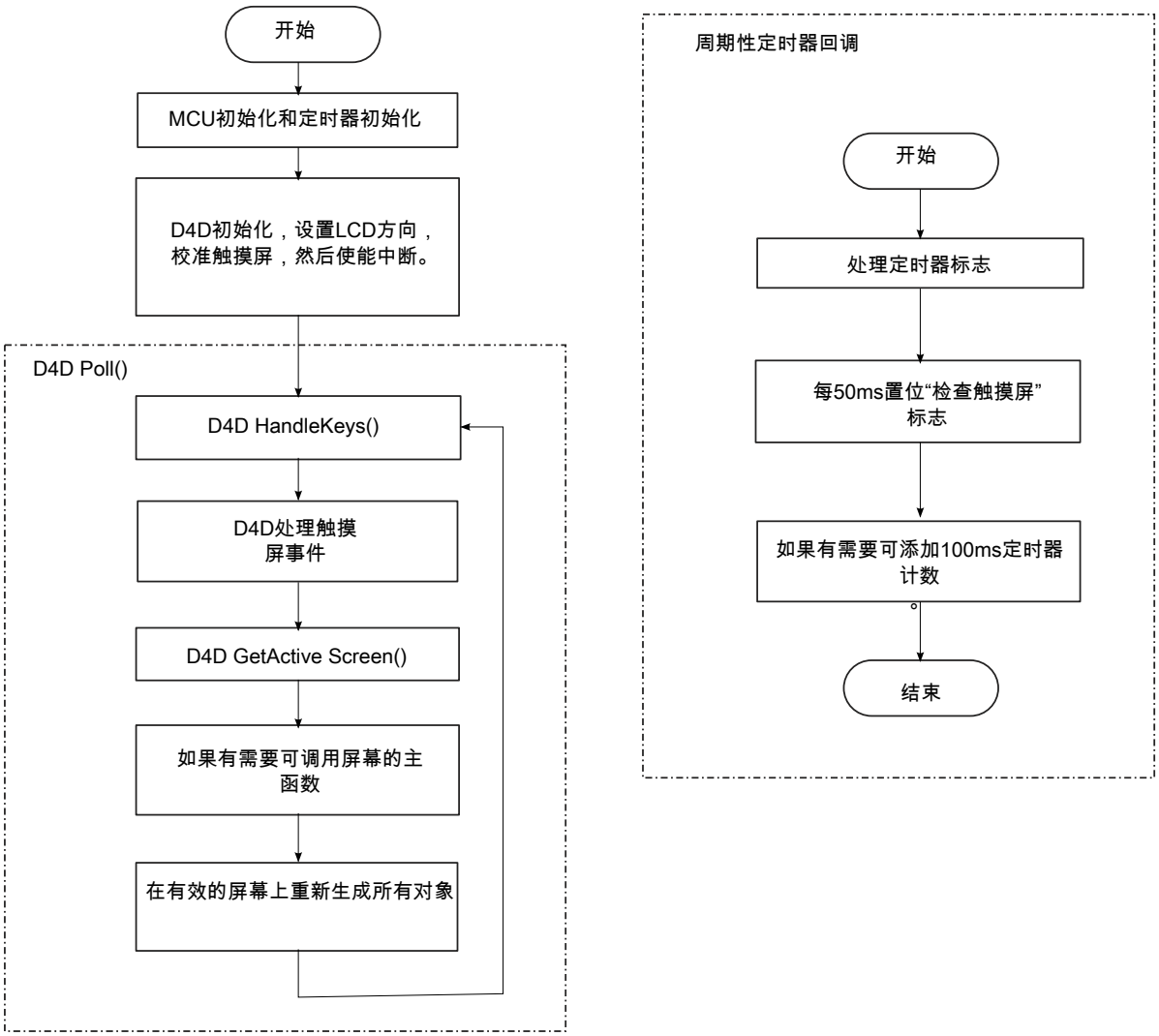


图 13. eGUI 主流程图

如图 13 所示，eGUI 主循环是 *D4D_Poll()*，而对于主循环来说，周期性定时器回调是必不可少的。*D4D_Poll()*处理按键、触摸屏或定时器事件，并在有效的屏幕上重新生成对象。

因此，eGUI 应用将具有与下文相同的主循环:

```
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 */
}
```

更多详情，请参考 http://www.freescale.com/egui 处提供的 eGUI 参考手册。

3.4 MQX 项目

对于 TWR-K60N512 板，要编译 eGUI MQX 项目，则必须先安装 MQX 3.6 或 3.7。然后，请确保在 MQX BSP 和 PSP 项目的 user_config.h 中，将 *BSPCFG_ENABLE_IO_SUBSYSTEM*、*BSP_DEFAULT_IO_CHANNEL*、*BSPCFG_ENABLE_ADC1* 和 *BSPCFG_ENABLE_SPI2*（如需将 SPI 用作 LCD 连接）定义为 1。并且，您必须重新编译 MQX BSP 和 PSP 项目，以便为 eGUI 生成新的 MQX 库，否则您的 eGUI 项目将无法正常工作。

对于 MQX 项目，我们为 eGUI 演示创建了两项任务：

```
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 }
};
```

对于 LCD 任务，主循环与裸机项目完全一致。不同之处在于，它需要开启 ADC 并将其用作 I/O 器件，才能用于触摸屏仿真；同时，我们还需在 *D4D_Init()*之前创建一个定时器任务。然而，要将有效任务切换至定时器任务以便进行定时器计时，那么*_time_delay()*是必不可少的。因此，MQX 项目的主循环看上去是这样子的：

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

3.5 屏幕截图

成功将正式 eGUI 例程的裸机或 MQX 版本下载至控制器模块板后，可看到 TWR-LCD 的显示内容如图 14 所示。

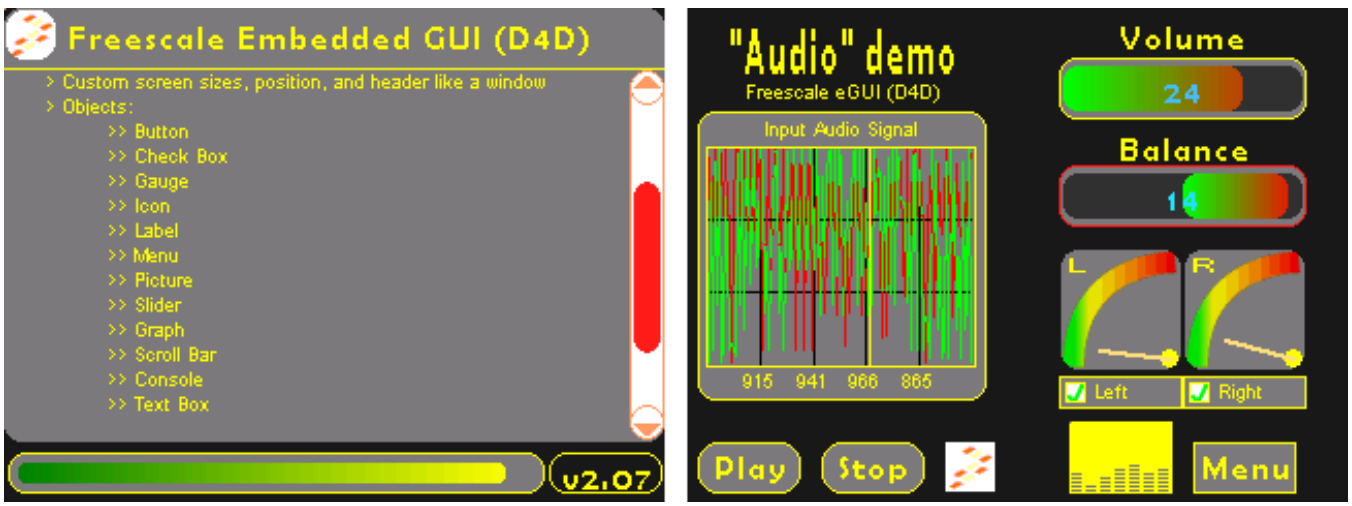


图 14. 正式 eGUI 例程的屏幕截图

4 利用 TWR-MCF5441X 显示 microWindows

Freescale 提供 TWR-MCF5441X 板，可用于运行 Linux。Freescale Linux BSP 工程师已实现在 TWR-LCD 板上显示 microWindows。

SSD1289 是 TWR-LCD 板上的 LCD 控制器，具有显示数据缓冲区。这就便于那些不具备大容量 RAM 的 MCU 显示用户界面的图形。但其劣势在于，用户必须创建一个线程才能模拟帧缓冲器，从而定期更新板载的显示数据缓冲区；这样就会降低整个系统的性能。但这样做确实有效。

这类代码（更新显示数据缓冲区的线程）在/drivers/video/fsl-ssd1289-fb.c 中可找到（如需最新的 LTIB ISO 镜像，请访问 http://www.freescale.com 并搜索“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);
}
```

要使能用于 TWR-MCF5441X 的 TWR-LCD 帧缓冲器驱动程序，可能需要阅读 BSP ISO 包中的帮助文件：\help \documents\html\M54418TWR_LCD.htm。更多详情，请参考 M54418 塔式 Linux BSP ISO 的 LTIB 帮助文件、用户手册和快速入门指南。

注

由于 MCF5441X 的 ADC_IN 引脚无法配置为 GPIO 端口，连接 TWR-MCF5441X 时，TWR-LCD 板上的触摸屏无法使用。

5 摘要

TWR-LCD 板可通过 Flexbus/mini-Flexbus 接口或 SPI/DSPI/QSPI 接口实现连接，从而在屏幕上显示一定内容。模拟具有两个 ADC 通道（可多路复用为 GPIO 引脚）和两个 GPIO 引脚的触摸屏接口并不复杂。

Freescale eGUI 可单独使用，也可集成到 MQX 操作系统中。Freescale 计划发布新版本 eGUI，以支持包括单通道 LCD 控制器在内的更多 LCD 控制器。它还将支持包括中文、韩文、日文等在内的多种语言。另外，eGUI 是完全免费的！

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

本文档中的信息仅供系统和软件实施方使用 Freescale 产品。 本文并未明示或者暗示授予利用本文档信息进行设计或者加工集成电路的版权许可。 Freescale 保留对此处任何产品进行更改的权利，恕不另行通知。

Freescale 对其产品在任何特定用途方面的适用性不做任何担保、表示或保证，也不承担因为应用程序或者使用产品或电路所产生的任何责任，明确拒绝承担包括但不局限于后果性的或附带性的损害在内的所有责任。 Freescale 的数据表和/或规格中所提供的“典型”参数在不同应用中可能并且确实不同，实际性能会随时间而有所变化。 所有运行参数，包括“经典值”在内，必须经由客户的技术专家对每个客户的应用程序进行验证。 Freescale 未转让与其专利权及其他权利相关的许可。 Freescale 销售产品时遵循以下网址中包含的标准销售条款和条件：freescale.com/SalesTermsandConditions.

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

© 2012 Freescale Semiconductor, Inc.

© 2012 飞思卡尔半导体有限公司

Document Number AN4452
Revision 0, 05/2012