2.8” TFT LCD (320×240) Module

Preliminary Datasheet

Ver. 1.0, Jan 2010

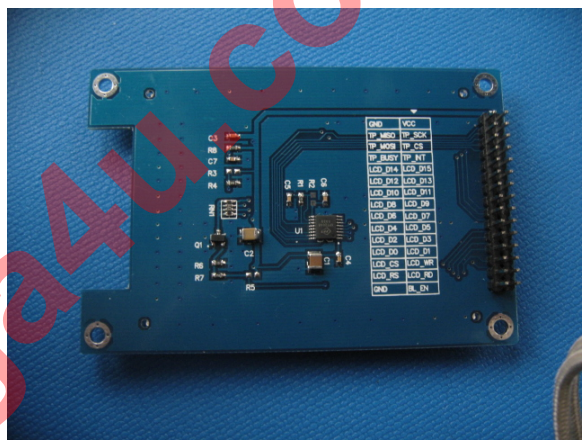
Overview

This module is a 262k Color 2.8 inch (320 × 240) TFT LCD Module based on ILI9331 LCD controller from ILITEK. The module integrated with TFT PANNEL, IC , FPC , back light and touch panel , which outputs 8/16bit bus interface and SPI interface in 2.54mm pitch PIN Header form .

ILI9331 is a 262,144-color one-chip SoC driver for a-TFT liquid crystal display with resolution of 240RGBx320 dots, comprising a 720-channel source driver, a 320-channel gate driver, 172,800 bytes RAM for graphic data of 240RGBx320 dots, and power supply circuit.

ILI9331 has many kinds of system interfaces, but this module only realize i80-system MPU interface (8-/16-bit bus width) which is widely used in embedded system.

The touch panel is 4-wire analog resistive type, the module use XPT2046 touch screen controller which is fully compatible with ADS7843 and output 8/12bit resolution digital data in SPI interface.



Features

- TFT Structure: TFT PANEL + IC + FPC
- Transmissive Type LCD , 262k Color 240 dot-source and 320 dot-gate outputs
- White LCD back light
- 8/16bit i80-system MPU interface
- Internal 172,800 byte graphic RAM
- Window address functioin to specify a rectangular area for internal GRAM access
- 8/12bit resolution touch screen controller output in SPI interface

Application

- Cellular phones
- PDAs
- Toys
- Other battery-powered products

Pin Definition

The data bus can be selected in 8bit or 16bit mode via R3/R4 resistor. When R3 is mounted 8bit bus mode is enabled , and R4 is mounted 16bit bus mode is enabled.

PIN NAME	TYPE	DESCRIPTION
1 VCC	Power	Power supply (3.3V)
2 GND	Ground	Power ground
3 SCK	Input	Touch screen controller SPI interface SCK
4MISO	Output	Touch screen controller SPI interface MISO
5 SS	Input	Touch screen controller SPI interface SS
6 MOSI	Input	Touch screen controller SPI interface MOSI
7 INT	Output	Touch screen controller SPI interface INT
8 BUSY	Input	Not used
9 D15	bidir	Data bus , D15 for 16bit mode, D7 for 8bit mode
10 D14	bidir	Data bus , D14 for 16bit mode, D6 for 8bit mode
11 D13	bidir	Data bus , D13 for 16bit mode, D5 for 8bit mode
12 D12	bidir	Data bus , D12 for 16bit mode, D4 for 8bit mode
13 D11	bidir	Data bus , D11 for 16bit mode, D3 for 8bit mode
14 D10	bidir	Data bus , D10 for 16bit mode, D2 for 8bit mode
15 D9	bidir	Data bus , D9 for 16bit mode, D1 for 8bit mode
16 D8	bidir	Data bus , D8 for 16bit mode, D0 for 8bit mode
17 D7	bidir	Data bus , D7 for 16bit mode, NA for 8bit mode
18 D6	bidir	Data bus , D6 for 16bit mode, NA for 8bit mode
19 D5	bidir	Data bus , D5 for 16bit mode, NA for 8bit mode
20 D4	bidir	Data bus , D4 for 16bit mode, NA for 8bit mode
21 D3	bidir	Data bus , D3 for 16bit mode, NA for 8bit mode
22 D2	bidir	Data bus , D2 for 16bit mode, NA for 8bit mode
23 D1	bidir	Data bus , D1 for 16bit mode, NA for 8bit mode
24 D0	bidir	Data bus , D0 for 16bit mode, NA for 8bit mode
25 WR	Input	Bus Write , active low
26 CS	Input	Bus Chipselect , active low
27 RD	Input	Bus Read , active low
28 RS	Input	Bus data or command select signal
29 BL_K	Input	Back light enable , active high
30 GND	Ground	Power ground

Bus Timing

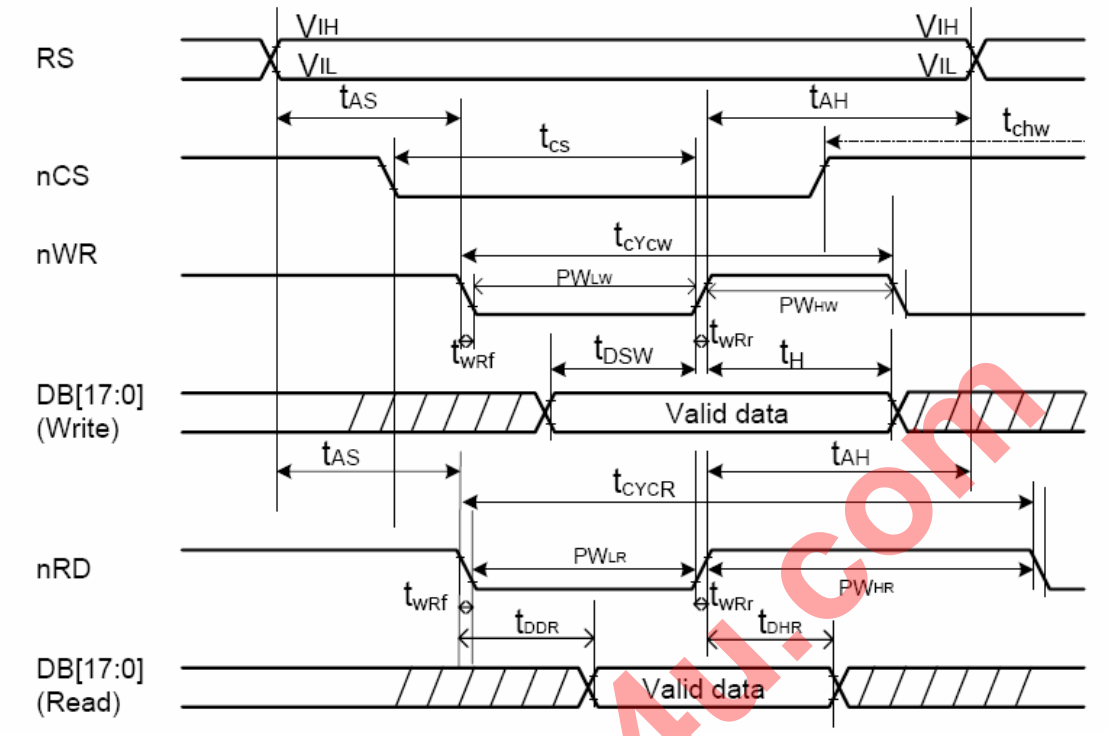


Figure 1 i80-System Bus Timing

Normal Write Mode (IOVCC = 1.65~3.3V)

Item		Symbol	Unit	Min.	Typ.	Max.	Test Condition
Bus cycle time	Write	t_{CYCW}	ns	TBD	-		-
	Read	t_{CYCR}	ns	300	-	-	-
Write low-level pulse width		PW_{LW}	ns	TBD	-	500	-
Write high-level pulse width		PW_{HW}	ns	TBD	-	-	-
Read low-level pulse width		PW_{LR}	ns	150	-	-	-
Read high-level pulse width		PW_{HR}	ns	150	-	-	
Write / Read rise / fall time		t_{WRr}/t_{WRf}	ns	-	-	25	
Setup time	Write (RS to nCS, E/nWR)	t_{AS}	ns	10	-	-	
	Read (RS to nCS, RW/nRD)			5	-	-	
Address hold time		t_{AH}	ns	5	-	-	
Write data set up time		t_{DSW}	ns	10	-	-	
Write data hold time		t_{H}	ns	15	-	-	
Read data delay time		t_{DDR}	ns	-	-	100	
Read data hold time		t_{DHR}	ns	5	-	-	

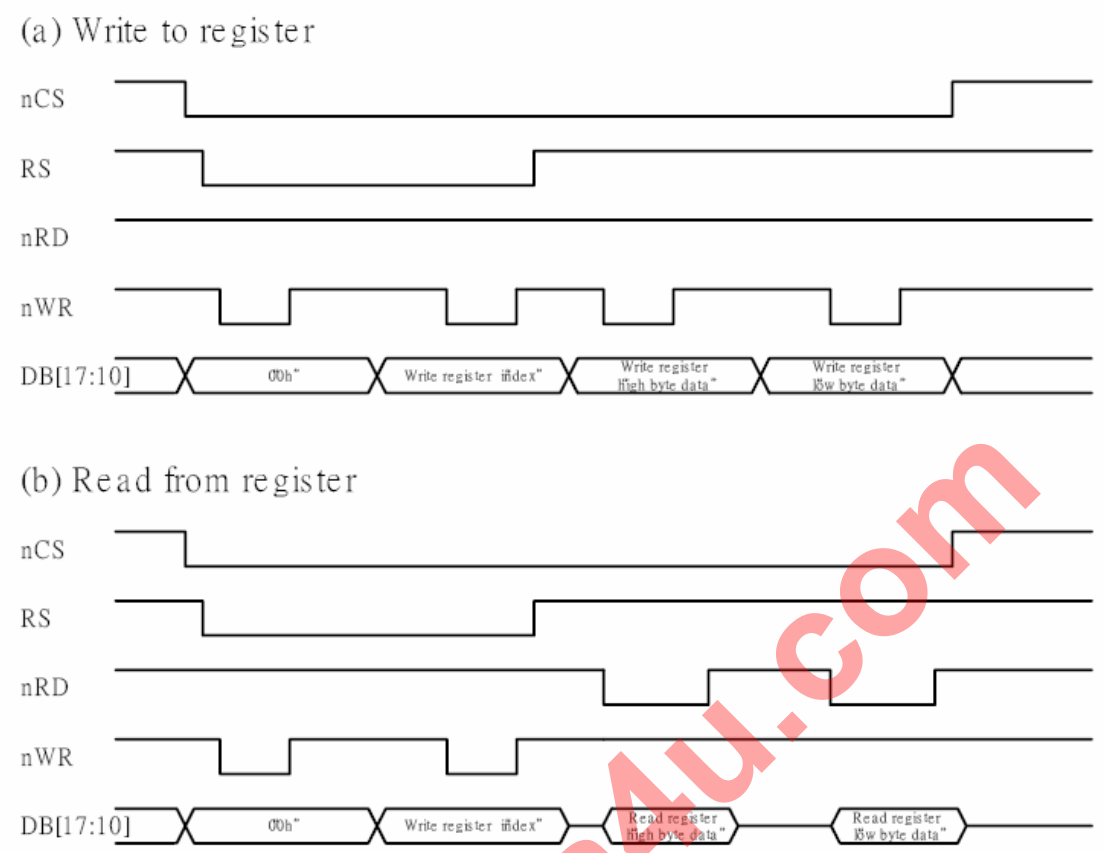


Figure 2 Register Read/Write Timing of 8bit i80 System Interface

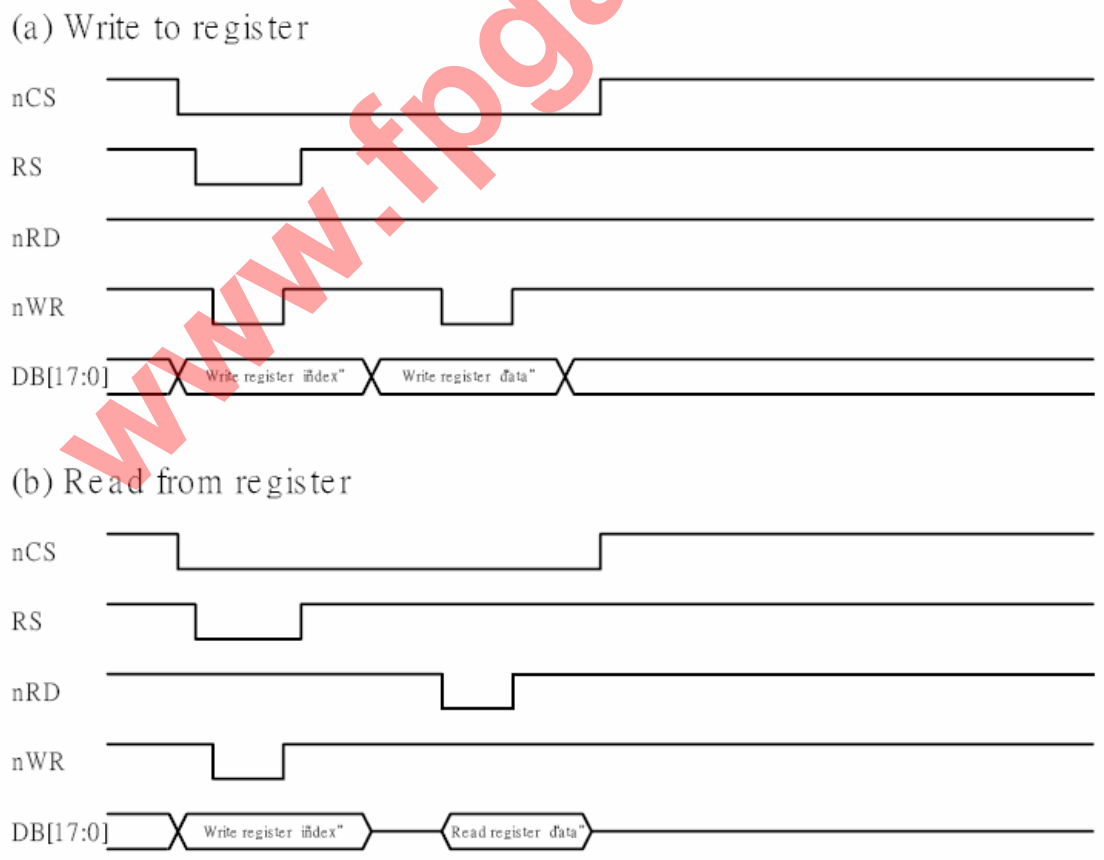
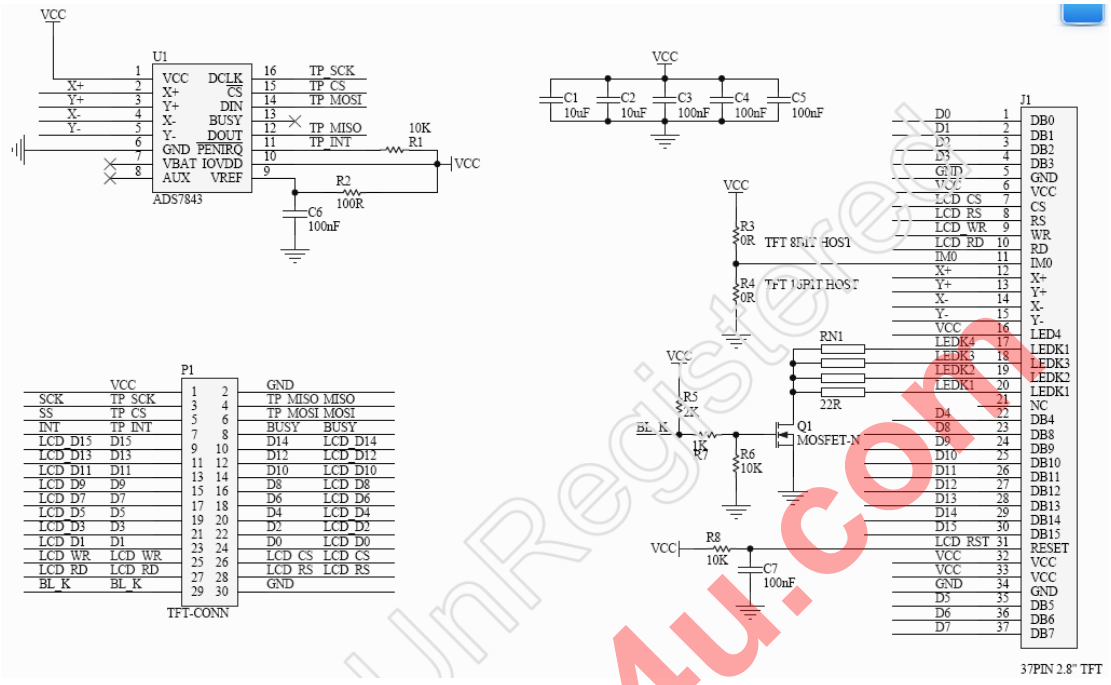


Figure 3 Register Read/Write Timing of 16bit i80 System Interface

Appendix A

Module Schematic



Appendix B

File “lcd.h” has all the functions prototypes for the lcd firmware driver.

```
/********************** (C) COPYRIGHT 2008 FPGA4u ***********************
* File Name          : lcd.h
* Author             : FPGA4u
* Version            : V1.0.0
* Date               : 07/22/2008
* Description        : This file contains all the functions prototypes for the
*                      lcd firmware driver.
*/

#ifndef __LCD_H
#define __LCD_H

#include "common.h"


/* Private typedef -----------------------------------------------------------*/
typedef struct
{
#ifdef LCD_16BIT_MODE
   vu16 LCD_REG;
```

```
   vu16 LCD_RAM;
#else
   vu8 LCD_REG;
   vu8 LCD_RAM;
#endif
} LCD_TypeDef;


#define LCD             ((LCD_TypeDef *) LCD_BASE)

/* LCD color */
#define White            0xFFFF
#define Black            0x0000
#define Grey             0xF7DE
#define Blue             0x001F
#define Blue2            0x051F
#define Red              0xF800
#define Magenta          0xF81F
#define Green            0x07E0
#define Cyan             0x7FFF
#define Yellow           0xFFE0

#define Line0            0
#define Line1            24
#define Line2            48
#define Line3            72
#define Line4            96
#define Line5            120
#define Line6            144
#define Line7            168
#define Line8            192
#define Line9            216

#define Horizontal       0x00
#define Vertical         0x01

/* LCD Registers */
#define R0               0x00
#define R1               0x01
#define R2               0x02
#define R3               0x03
#define R4               0x04
#define R5               0x05
#define R6               0x06
```

```
#define R7          0x07
#define R8          0x08
#define R9          0x09
#define R10         0x0A
#define R12         0x0C
#define R13         0x0D
#define R14         0x0E
#define R15         0x0F
#define R16         0x10
#define R17         0x11
#define R18         0x12
#define R19         0x13
#define R20         0x14
#define R21         0x15
#define R22         0x16
#define R23         0x17
#define R24         0x18
#define R25         0x19
#define R26         0x1A
#define R27         0x1B
#define R28         0x1C
#define R29         0x1D
#define R30         0x1E
#define R31         0x1F
#define R32         0x20
#define R33         0x21
#define R34         0x22
#define R36         0x24
#define R37         0x25
#define R40         0x28
#define R41         0x29
#define R43         0x2B
#define R45         0x2D
#define R48         0x30
#define R49         0x31
#define R50         0x32
#define R51         0x33
#define R52         0x34
#define R53         0x35
#define R54         0x36
#define R55         0x37
#define R56         0x38
#define R57         0x39
#define R59         0x3B
```

```
#define R60             0x3C
#define R61             0x3D
#define R62             0x3E
#define R63             0x3F
#define R64             0x40
#define R65             0x41
#define R66             0x42
#define R67             0x43
#define R68             0x44
#define R69             0x45
#define R70             0x46
#define R71             0x47
#define R72             0x48
#define R73             0x49
#define R74             0x4A
#define R75             0x4B
#define R76             0x4C
#define R77             0x4D
#define R78             0x4E
#define R79             0x4F
#define R80             0x50
#define R81             0x51
#define R82             0x52
#define R83             0x53
#define R96             0x60
#define R97             0x61
#define R106            0x6A
#define R118            0x76
#define R128            0x80
#define R129            0x81
#define R130            0x82
#define R131            0x83
#define R132            0x84
#define R133            0x85
#define R134            0x86
#define R135            0x87
#define R136            0x88
#define R137            0x89
#define R139            0x8B
#define R140            0x8C
#define R141            0x8D
#define R143            0x8F
#define R144            0x90
#define R145            0x91
```

```
#define R146              0x92
#define R147              0x93
#define R148              0x94
#define R149              0x95
#define R150              0x96
#define R151              0x97
#define R152              0x98
#define R153              0x99
#define R154              0x9A
#define R157              0x9D
#define R192              0xC0
#define R193              0xC1
#define R229              0xE5

/* Exported functions ------------------------------------------------------- */
/*----- High layer function -----*/
void LCD_Init(void);
void LCD_SetTextColor(vu16 Color);
void LCD_SetBackColor(vu16 Color);
void LCD_ClearLine(u8 Line);
void LCD_Clear(u16 Color);
void LCD_SetCursor(u8 Xpos, u16 Ypos);
void LCD_DrawChar(u8 Xpos, u16 Ypos, uc16 *c);
void LCD_DisplayChar(u8 Line, u16 Column, u8 Ascii);
void LCD_DisplayStringLine(u8 Line, u8 *ptr);
void LCD_SetDisplayWindow(u8 Xpos, u16 Ypos, u8 Height, u16 Width);
void LCD_WindowModeDisable(void);
void LCD_DrawLine(u8 Xpos, u16 Ypos, u16 Length, u8 Direction);
void LCD_DrawRect(u8 Xpos, u16 Ypos, u8 Height, u16 Width);
void LCD_DrawCircle(u8 Xpos, u16 Ypos, u16 Radius);
void LCD_DrawMonoPict(uc32 *Pict);
void LCD_WriteBMP(u32 BmpAddress);

/*----- Medium layer function -----*/
void LCD_WriteReg(u8 LCD_Reg, u16 LCD_RegValue);
u16 LCD_ReadReg(u8 LCD_Reg);
void LCD_WriteRAM_Prepare(void);
void LCD_WriteRAM(u16 RGB_Code);
u16 LCD_ReadRAM(void);
void LCD_PowerOn(void);
void LCD_DisplayOn(void);
void LCD_DisplayOff(void);
void LCD_Disp_Image(unsigned char *);
```

```
/*----- Low layer function -----*/


#endif /*__LCD_H */

/******************* (C) COPYRIGHT 2008 FPGA4u *****END OF FILE****/
File “lcd.c” has all the functions definition for the lcd firmware driver.
/******************** (C) COPYRIGHT 2008 FPGA4u ********************
* File Name              : lcd.c
* Author                 : FPGA4u
* Version                : V1.0.0
* Date                   : 07/22/2008
* Description            : This file contains all the functions definition for the
*                          lcd firmware driver.
*/

#include "LCD.h"
#include "common.h"
#include <stdio.h>

/*******************************************************************************
*
* Function Name  : LCD_Init
* Description    : Initializes the LCD.
* Input          : None
* Output         : None
* Return         : None
*******************************************************************************
/
void LCD_Init(void)
{
/* Configure the LCD Control pins ---------------------------------------*/


   LCD_BL_L();
   Delay(5); /* delay 50 ms */
/* Start Initial Sequence -----------------------------------------------*/
//    LCD_WriteReg(R229,0x8000); /* Set the internal vcore voltage */
  // LCD_WriteReg(R0,    0x0001); /* Start internal OSC. */
   LCD_WriteReg(R1,    0x0000); /* set SS and SM bit */
   LCD_WriteReg(R2,    0x0700); /* set 1 line inversion */
   LCD_WriteReg(R3,    0x1030); /* set GRAM write direction and BGR=1. */
   LCD_WriteReg(R4,    0x0000); /* Resize register */
   LCD_WriteReg(R8,    0x0207); /* set the back porch and front porch */
```

```
    LCD_WriteReg(R9,    0x0000); /* set non-display area refresh cycle ISC[3:0] */
    LCD_WriteReg(R10, 0x0000); /* FMARK function */
    LCD_WriteReg(R12, 0x0000); /* RGB interface setting */
    LCD_WriteReg(R13, 0x0000); /* Frame marker Position */
    LCD_WriteReg(R15, 0x0000); /* RGB interface polarity */

/* Power On sequence -----------------------------------------------------------*/
    LCD_WriteReg(R16, 0x0080); /* SAP, BT[3:0], AP, DSTB, SLP, STB */
    LCD_WriteReg(R17, 0x0007); /* DC1[2:0], DC0[2:0], VC[2:0] */
    LCD_WriteReg(R18, 0x0000); /* VREG1OUT voltage */
    LCD_WriteReg(R19, 0x0000); /* VDV[4:0] for VCOM amplitude */
    Delay(20);                 /* Dis-charge capacitor power voltage (200ms) */
    LCD_WriteReg(R16, 0x1190); /* SAP, BT[3:0], AP, DSTB, SLP, STB */
    LCD_WriteReg(R17, 0x0227); /* DC1[2:0], DC0[2:0], VC[2:0] */
    Delay(5);                  /* Delay 50 ms */
    LCD_WriteReg(R18, 0x001a); /* VREG1OUT voltage */
    Delay(5);                  /* Delay 50 ms */
    LCD_WriteReg(R19, 0x1000); /* VDV[4:0] for VCOM amplitude */
    LCD_WriteReg(R41, 0x0013); /* VCM[4:0] for VCOMH */
    LCD_WriteReg(R43, 0x000c); // Set Frame Rate
    Delay(5);                  /* Delay 50 ms */
    LCD_WriteReg(R32, 0x0000); /* GRAM horizontal Address */
    LCD_WriteReg(R33, 0x0000); /* GRAM Vertical Address */

/* Adjust the Gamma Curve ------------------------------------------------------*/
    LCD_WriteReg(R48, 0x0000);
    LCD_WriteReg(R49, 0x0606);
    LCD_WriteReg(R50, 0x0505);
    LCD_WriteReg(R53, 0x0101);
    LCD_WriteReg(R54, 0x0202);
    LCD_WriteReg(R55, 0x0000);
    LCD_WriteReg(R56, 0x0202);
    LCD_WriteReg(R57, 0x0505);
    LCD_WriteReg(R60, 0x0202);
    LCD_WriteReg(R61, 0x1f01);

/* Set GRAM area ---------------------------------------------------------------*/
    LCD_WriteReg(R80, 0x0000); /* Horizontal GRAM Start Address */
    LCD_WriteReg(R81, 0x00EF); /* Horizontal GRAM End Address */
    LCD_WriteReg(R82, 0x0000); /* Vertical GRAM Start Address */
    LCD_WriteReg(R83, 0x013F); /* Vertical GRAM End Address */

    LCD_WriteReg(R96,    0xa700); /* Gate Scan Line */
    LCD_WriteReg(R97,    0x0001); /* NDL,VLE, REV */
```

```
    LCD_WriteReg(R106, 0x0000); /* set scrolling line */

/* Partial Display Control ---------------------------------------------------*/
    LCD_WriteReg(R128, 0x0000);
    LCD_WriteReg(R129, 0x0000);
    LCD_WriteReg(R130, 0x0000);
    LCD_WriteReg(R131, 0x0000);
    LCD_WriteReg(R132, 0x0000);
    LCD_WriteReg(R133, 0x0000);

/* Panel Control -------------------------------------------------------------*/
    LCD_WriteReg(R144, 0x0010);
    LCD_WriteReg(R146, 0x0000);
    LCD_WriteReg(R147, 0x0003);
    LCD_WriteReg(R149, 0x0110);
    LCD_WriteReg(R151, 0x0000);
    LCD_WriteReg(R152, 0x0000);

    /* Set GRAM write direction and BGR = 1 */
    /* I/D=01 (Horizontal : increment, Vertical : decrement) */
    /* AM=1 (address is updated in vertical writing direction) */
    LCD_WriteReg(R3, 0x1018);
    LCD_WriteReg(R7, 0x0173); /* 262K color and display ON */
    /* Clear the LCD */
    LCD_Clear(0);
    Delay(50);
    LCD_BL_H();
}

/*******************************************************************************
*
* Function Name  : LCD_Clear
* Description    : Clears the hole LCD.
* Input          : Color: the color of the background.
* Output         : None
* Return         : None
*******************************************************************************
/
void LCD_Clear(u16 Color)
{
    u32 index = 0;

    LCD_SetCursor(0x00, 0x013F);
```

```
    LCD_WriteRAM_Prepare(); /* Prepare to write GRAM */

    for(index = 0; index < 76800*2; index++)
    {
      LCD->LCD_RAM = Color;
    }
}

void LCD_Disp_Image(unsigned char *gImage_ptr)
{
    u32 i = 0;
    LCD_SetCursor(0x00, 0x013F);

    LCD_WriteRAM_Prepare(); /* Prepare to write GRAM */

    for(i = 0; i < 153600; i++)
    {
      LCD->LCD_RAM = gImage_ptr[i];
    }
}

/*******************************************************************************
*
* Function Name    : LCD_SetCursor
* Description      : Sets the cursor position.
* Input            : - Xpos: specifies the X position.
*                      - Ypos: specifies the Y position.
* Output           : None
* Return           : None
*******************************************************************************
/
void LCD_SetCursor(u8 Xpos, u16 Ypos)
{
    LCD_WriteReg(R32, Xpos);
    LCD_WriteReg(R33, Ypos);
}

/*******************************************************************************
*
* Function Name    : LCD_DisplayOn
* Description      : Enables the Display.
* Input            : None
* Output           : None
* Return           : None
```

```
*******************************************************************************
/
void LCD_DisplayOn(void)
{
  /* Display On */
  LCD_WriteReg(R7, 0x0173); /* 262K color and display ON */
}

/*******************************************************************************
*
* Function Name  : LCD_DisplayOff
* Description    : Disables the Display.
* Input          : None
* Output         : None
* Return         : None
*******************************************************************************
/
void LCD_DisplayOff(void)
{
  /* Display Off */
  LCD_WriteReg(R7, 0x0);
}

/*******************************************************************************
*
*                                    Low level routine for LCD Moudle
*
*******************************************************************************
/

/*******************************************************************************
*
* Function Name  : LCD_WriteReg
* Description    : Writes to the selected LCD register.
* Input          : - LCD_Reg: address of the selected register.
*                  - LCD_RegValue: value to write to the selected register.
* Output         : None
* Return         : None
*******************************************************************************
/
void LCD_WriteReg(u8 LCD_Reg, u16 LCD_RegValue)
{
#ifdef LCD_16BIT_MODE
  /* Write 16-bit Index, then Write Reg */
```

```
   LCD->LCD_REG = LCD_Reg;
   /* Write 16-bit Reg */
   LCD->LCD_RAM = LCD_RegValue;
#else
       /* Write 16-bit Index, then Write Reg */
       LCD->LCD_REG = 0x00;
   LCD->LCD_REG = LCD_Reg;
   /* Write 16-bit Reg */
   LCD->LCD_RAM = LCD_RegValue>>8 & 0x00ff;
   LCD->LCD_RAM = LCD_RegValue & 0x00ff;
#endif
}

/*******************************************************************************
*
* Function Name    : LCD_ReadReg
* Description      : Reads the selected LCD Register.
* Input            : None
* Output           : None
* Return           : LCD Register Value.
*******************************************************************************
/
u16 LCD_ReadReg(u8 LCD_Reg)
{
       u16 val = 0;
#ifdef LCD_16BIT_MODE
   /* Write 16-bit Index (then Read Reg) */
       LCD->LCD_REG = LCD_Reg;
   /* Read 16-bit Reg */
   val = LCD->LCD_RAM;
#else
       /* Write 16-bit Index (then Read Reg) */
   LCD->LCD_REG = 0x00;
   LCD->LCD_REG = LCD_Reg;
   /* Read 16-bit Reg */
   val = ((LCD->LCD_RAM)<<8) & 0xff00;
   val |= LCD->LCD_RAM;
#endif
       return val;
}

/*******************************************************************************
*
* Function Name    : LCD_WriteRAM_Prepare
```

```
* Description    : Prepare to write to the LCD RAM.
* Input          : None
* Output         : None
* Return         : None
*******************************************************************************
/
void LCD_WriteRAM_Prepare(void)
{
#ifdef LCD_16BIT_MODE
  LCD->LCD_REG = R34; /* Select GRAM Reg */
#else
  LCD->LCD_REG = 0x00;
  LCD->LCD_REG = R34; /* Select GRAM Reg */
#endif
}

/*******************************************************************************
*
* Function Name  : LCD_WriteRAM
* Description    : Writes to the LCD RAM.
* Input          : - RGB_Code: the pixel color in RGB mode (5-6-5).
* Output         : None
* Return         : None
*******************************************************************************
/
void LCD_WriteRAM(u16 RGB_Code)
{
    /* Write 16-bit GRAM Reg */
#ifdef LCD_16BIT_MODE
  LCD->LCD_RAM = RGB_Code;
#else
  LCD->LCD_REG = (RGB_Code>>8) & 0x00ff;
  LCD->LCD_RAM = RGB_Code & 0x00ff ;
#endif
}

/*******************************************************************************
*
* Function Name  : LCD_ReadRAM
* Description    : Reads the LCD RAM.
* Input          : None
* Output         : None
* Return         : LCD RAM Value.
*******************************************************************************
```

```
/
u16 LCD_ReadRAM(void)
{
      u16 val = 0;
#ifdef LCD_16BIT_MODE
   /* Write 16-bit Index (then Read Reg) */
      LCD->LCD_REG = R34; /* Select GRAM Reg */
   /* Read 16-bit Reg */
   val = LCD->LCD_RAM;
#else
      /* Write 16-bit Index (then Read Reg) */
   LCD->LCD_REG = 0x00;
   LCD->LCD_REG = R34; /* Select GRAM Reg */
   /* Read 16-bit Reg */
   val = ((LCD->LCD_RAM)<<8) & 0xff00;
   val |= LCD->LCD_RAM;
#endif
   return val;
}


/******************* (C) COPYRIGHT 2008 FPGA4u *****END OF FILE****/

File “main.c” demonstrate for the usage of lcd firmware driver.
#include <stdio.h>
#include "common.h"
#include "lcd.h"
//#include "rain.h"
#include "winxp.h"
#include "iGolf.h"
#include "girl.h"

int main()
{

      LCD_RST_L();
      usleep(10000);
      LCD_RST_H();
      LCD_Init();
      /* Infinite loop */
      while(1)
      {
            LCD_Disp_Image(gImage_winxp);
            usleep(3000000);
```

```
            LCD_Clear(0);
            LCD_Disp_Image(gImage_iGolf);
            usleep(3000000);
            LCD_Clear(0);
            /*
            LCD_Disp_Image(gImage_girl);
            usleep(3000000);
            LCD_Clear(0);

            LCD_Disp_Image(gImage_rain);
            usleep(3000000);
            LCD_Clear(0);
            */
      }
      return 0;
}
```

File “common.h” contains platform based definition for the main application.

```
/********************* (C) COPYRIGHT 2008 FPGA4u **********************
* File Name              : common.h
* Author                 : FPGA4u
* Version                : V1.0.0
* Date                   : 07/22/2008
* Description            : This file contains platform based definition for the
*                          main application.
*/

#ifndef __COMMON_H
#define __COMMON_H

#include "my_types.h"
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include "stdio.h"

//#define LCD_16BIT_MODE 0

#define LCD_BASE        ((u32)AVALON_TFT_0_BASE)
#define LCD_BL_H()    IOWR_ALTERA_AVALON_PIO_DATA(TFT_LCD_BL_BASE,1)
#define LCD_BL_L()    IOWR_ALTERA_AVALON_PIO_DATA(TFT_LCD_BL_BASE,0)
#define LCD_RST_H()    IOWR_ALTERA_AVALON_PIO_DATA(TFT_LCD_RSTN_BASE,1)
#define LCD_RST_L()    IOWR_ALTERA_AVALON_PIO_DATA(TFT_LCD_RSTN_BASE,0)
#define Delay(val)    usleep(val*1000)
```

```
#endif /*__COMMON_H */

/******************* (C) COPYRIGHT 2008 FPGA4u *****END OF FILE****/
File “my_types.h” has contains all the common data types used for the lcd firmware driver.
/******************* (C) COPYRIGHT 2008 FPGA4u *********************
* File Name              : my_types.h
* Author                 : MCD Application Team
* Version                : V2.0.2
* Date                   : 07/11/2008
* Description            : This file contains all the common data types used for the
*                          firmware library.
********************************************************************************
*
* THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS
* WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME.
* AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT,
* INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE
* CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE CODING
* INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.
********************************************************************************
/

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MY_TYPES_H
#define __MY_TYPES_H

/* Includes ------------------------------------------------------------------*/
/* Exported types ------------------------------------------------------------*/
typedef signed long    s32;
typedef signed short s16;
typedef signed char    s8;

typedef signed long    const sc32;    /* Read Only */
typedef signed short const sc16;    /* Read Only */
typedef signed char    const sc8;     /* Read Only */

typedef volatile signed long    vs32;
typedef volatile signed short vs16;
typedef volatile signed char    vs8;
```

```
typedef volatile signed long    const vsc32;    /* Read Only */
typedef volatile signed short const vsc16;    /* Read Only */
typedef volatile signed char    const vsc8;     /* Read Only */

typedef unsigned long    u32;
typedef unsigned short u16;
typedef unsigned char    u8;

typedef unsigned long    const uc32;    /* Read Only */
typedef unsigned short const uc16;    /* Read Only */
typedef unsigned char    const uc8;     /* Read Only */

typedef volatile unsigned long    vu32;
typedef volatile unsigned short vu16;
typedef volatile unsigned char    vu8;

typedef volatile unsigned long    const vuc32;    /* Read Only */
typedef volatile unsigned short const vuc16;    /* Read Only */
typedef volatile unsigned char    const vuc8;     /* Read Only */

typedef enum {FALSE = 0, TRUE = !FALSE} bool;

typedef enum {RESET = 0, SET = !RESET} FlagStatus, ITStatus;

typedef enum {DISABLE = 0, ENABLE = !DISABLE} FunctionalState;
#define IS_FUNCTIONAL_STATE(STATE) (((STATE) == DISABLE) || ((STATE) ==
ENABLE))

typedef enum {ERROR = 0, SUCCESS = !ERROR} ErrorStatus;

#define U8_MAX       ((u8)255)
#define S8_MAX       ((s8)127)
#define S8_MIN       ((s8)-128)
#define U16_MAX      ((u16)65535u)
#define S16_MAX      ((s16)32767)
#define S16_MIN      ((s16)-32768)
#define U32_MAX      ((u32)4294967295uL)
#define S32_MAX      ((s32)2147483647)
#define S32_MIN      ((s32)-2147483648)

/* Exported constants --------------------------------------------------*/
/* Exported macro ------------------------------------------------------*/
/* Exported functions ------------------------------------------------- */
```

```
#endif /* __MY_TYPES_H */

/****************** (C) COPYRIGHT 2008 FPGA4u *****END OF FILE****/
```