乐升半导体
LEVETOP

LT7683/LT7381
LCD Controller Board

User Manual

V1.1

www.levetop.cn

Levetop Semiconductor Co., Ltd.

1. Introduction

1-1 LT7683

LT7683 is a powerful chip that can control up to 1024x768 TFT displays. It has 128Mb of RAM that can buffer the display. It also has a geometric drawing engine that provides command-type graphic operations such as drawing points, lines, curves, ellipses, triangles, rectangles etc. With an embedded hardware graphics acceleration engine (BTE), LT7683 enables users to implement screen rotation, flipping, mirroring, PIP, graphics blending and more. You may refer to the datasheet and application note for more details.

MCU may connect to this controller board through either 4-wire SPI or 8 bits parallel (8080/6800) interface (See Table 1-1 for the board features summary). The controller board provides RTP/CTP interfaces. Other than the CTP pins, the RTP pins are also made available so that users may connect to other touch controllers. In addition, LT7683 controller board is designed to fit in with the typical 40pins/50pins TFT displays. Please refer to the pin definition listed in Table 1-2 and Table 1-3.

In addition, there is an 128Mb SPI Flash on the board. Users may store their images, fonts, and animations in the SPI Flash, and then utilize a simple DMA command of LT7683 to access the Flash and retrieve those data.

A customized MCU board is also available as a set with the LT7683 controller board. The MCU board has a typical STM32F103 on it. Users can combine these two boards to work on their own projects in no time.

To get you started, we have prepared a library with various examples. Download them from LEVETOP website and install as described in Chapter 2. Connect a 40 or 50pins TFT to the FPC port and wire up the interface (SPI/Parallel) to your MCU. You may then try out those provided examples.

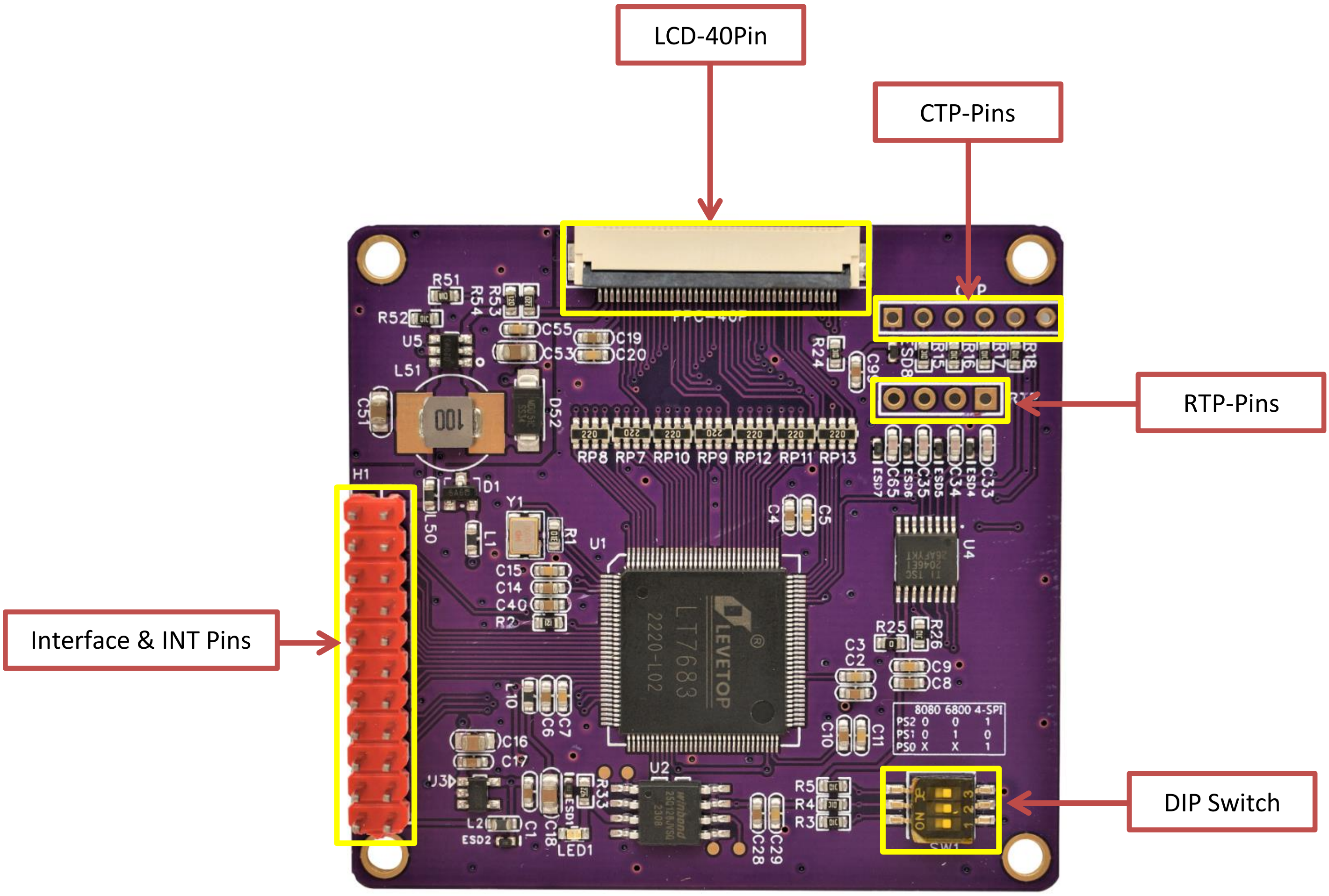
1-2 LT7381

LT7381 shares all the functionalities of LT7683, except that it does not support external Flash, and its RAM size is 32Mb.

For more detail about what LT7683/LT7381 can do, please check the datasheet and application note.
LT768x Datasheet: https://www.levetop.cn/data/LT768x_DS_V42_ENG.pdf
LT768x Application Note: https://www.levetop.cn/data/LT768x_AP-Note_V12_ENG.pdf

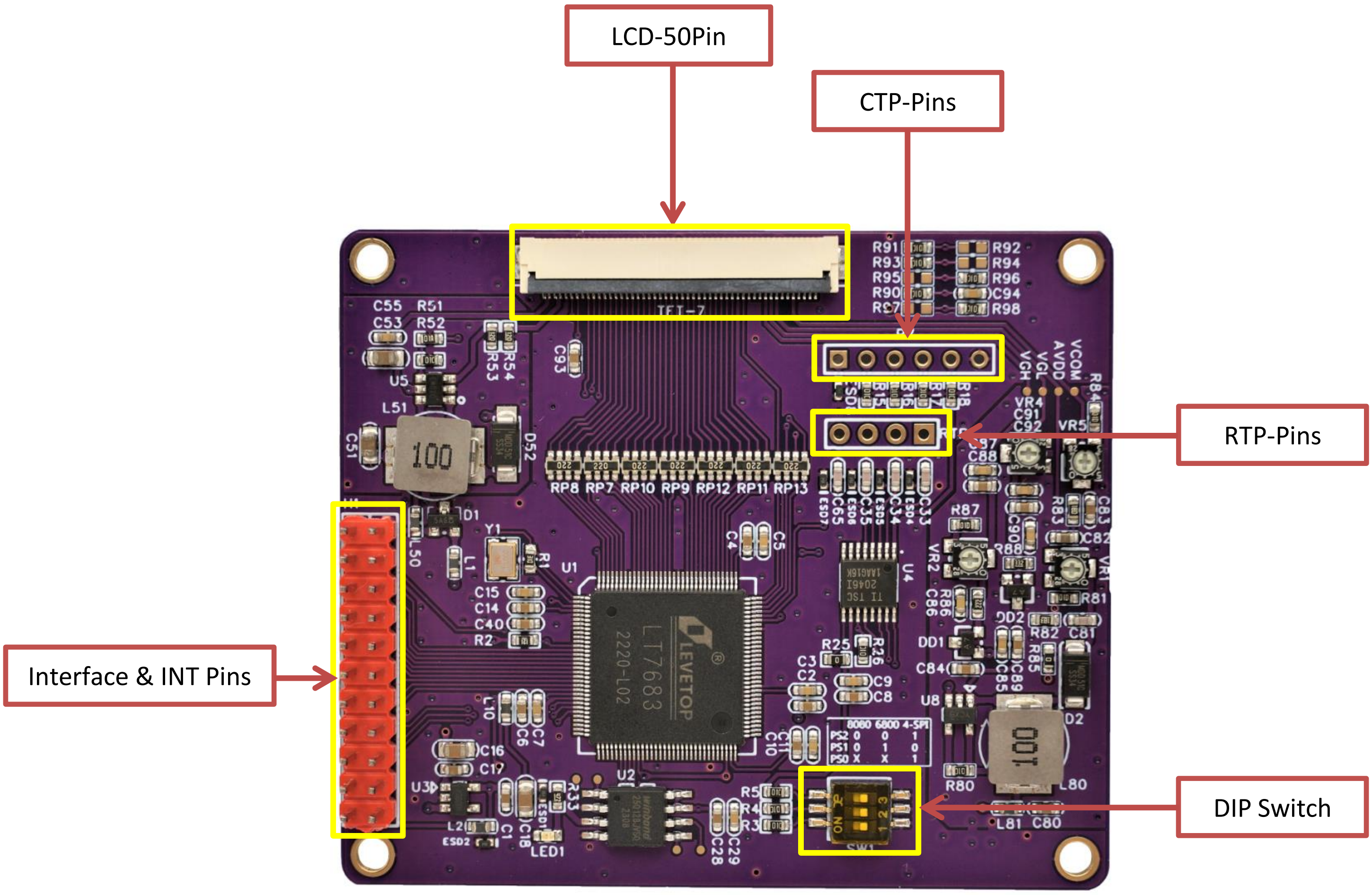
1-3 Board Features

LCD controller board for 40Pin LCD panel:



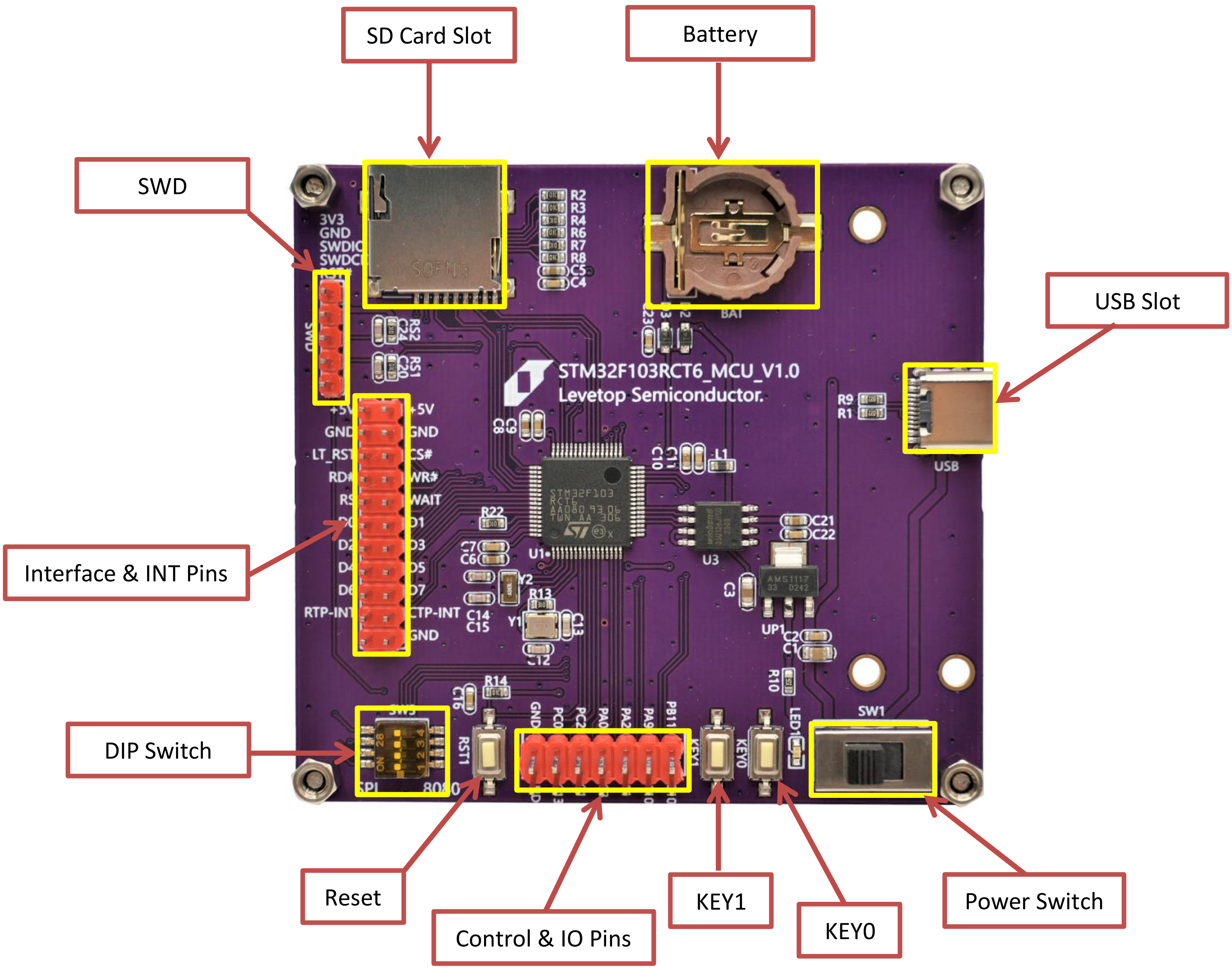
LCD-40Pin	:	Connector for 40Pin RGB panel
CTP-Pins	:	CTP connector (I2C)
RTP-Pins	:	RTP connector
DIP Switch	:	For setting the interface between LT7683/LT7381 and the host MCU
Interface & INT Pins	:	4-wire SPI & 8bit-parallel (6800/8080 mode), INT pins of RTP/CTP

LCD controller board for 50Pin LCD panel:



LCD-50Pin	:	Connector for 50Pin RGB panel
CTP-Pins	:	CTP connector (I2C)
RTP-Pins	:	RTP connector
DIP Switch	:	For setting the interface between LT7683/LT7381 and the host MCU
Interface & INT Pins	:	4-wire SPI & 8bit-parallel (6800/8080 mode), INT pins of RTP/CTP

STM32F103x MCU board:



SD Card Slot	:	For programming MCU code and SPI Flash data
Battery	:	For RTC functions
USB Slot	:	Used as power entry
Power Switch	:	Power Switch
KEY0	:	Programmable button
KEY1	:	Programmable button
Control & IO Pins	:	Available IO pins (PA9 & PA10 are used as Uart interface in the demo code)
Reset	:	Reset for the MCU board
DIP Switch	:	For differentiating SPI and Parallel interfaces
Interface & INT pins	:	4-wire SPI & 8bit-Parallel (6800/8080 mode), INT pins of RTP/CTP
SWD	:	Debugging interface for STM32 MCU

Table 1-1: Board Features

No.	Feature	Description
1	MCU Interface	4-wire SPI / 8bits Parallel (8080/6800)
2	LCD Interface	Typical 40pins / 50pins
3	Resolution	Max. 1024 x 768
4	Color depth	Max. 24bits /16.7M colors
5	IC embedded Display RAM	128Mb / 32Mb
6	Touch Panel Interface	RTP / CTP (I2C)
7	External SPI Flash	128Mb Nor Flash on board

Table 1-2: LCD Pin Definition Table (40pins)

Pin no.	Pin name	Description
1	LEDK	LED backlight
2	LEDA	LED backlight
3	GND	GND
4	VDD	Power supply
5~12	R0~R7	Red data bus
13~20	G0~G7	Green data bus
21~28	B0~B7	Blue data bus
29	GND	GND
30	PCLK	Data clock
31	ON/OFF	Standby mode select pin
32	HSYNC	Horizontal SYNC signal
33	VSYNC	Vertical SYNC signal
34	DE	Data enable pin
35	NC	NC
36	GND	GND
37	X+	For Resistive Touch Panel
38	Y+	For Resistive Touch Panel
39	X-	For Resistive Touch Panel
40	Y-	For Resistive Touch Panel

Table 1-3: LCD Pin Definition Table (50pins)

Pin no.	Pin name	Description
1	LEDA	LED backlight
2	LEDA	LED backlight
3	LEDK	LED backlight
4	LEDK	LED backlight
5	GND	GND
6	VCOM	VCOM
7	VDD_3.3V	Power for digital circuit
8	MODE	DE/SYNC mode select
9	DE	Data enable
10	VSYNC	Vertical SYNC signal
11	HSYNC	Horizontal SYNC signal
12~19	R7~R0	Red data bus
20~27	G7~G0	Green data bus
28~35	B7~B0	Blue data bus
36	GND	GND
37	PCLK	Data clock
38	GND	GND
39	L/R	Left/Right selection
40	U/D	Up/Down selection
41	VGH	Gate ON voltage
42	VGL	Gate OFF voltage
43	LCD_AVDD	Power for analog circuit
44	LCD_RST	Reset pin
45	NC	NC
46	VCOM	VCOM
47	DITH	Dithering function
48	GND	GND
49	NC	NC
50	NC	NC

1-4 PCB outline

LT7683/LT7381 Controller Board

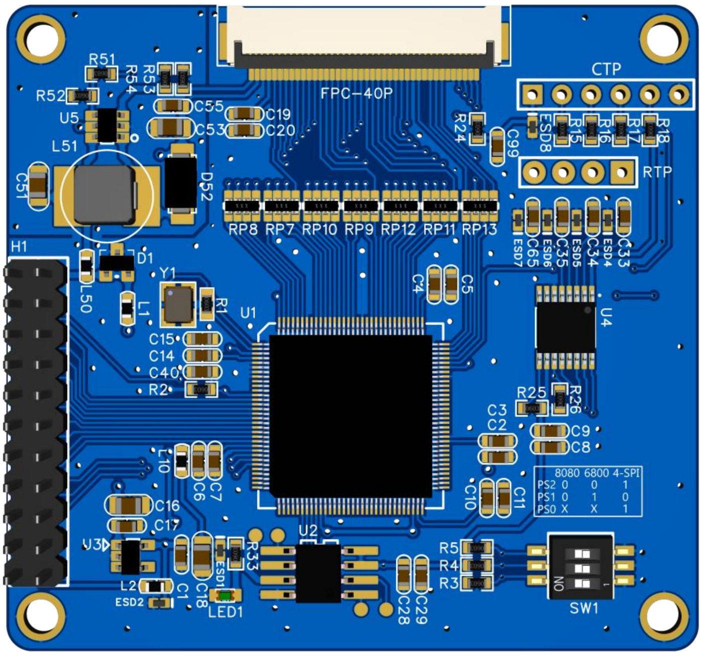


Figure 1-1: LT7683/LT7381 Controller Board for 40Pin LCD

(Dimension: 60mm x 56mm x 4.6mm (2.36" x 2.2" x 0.18"))

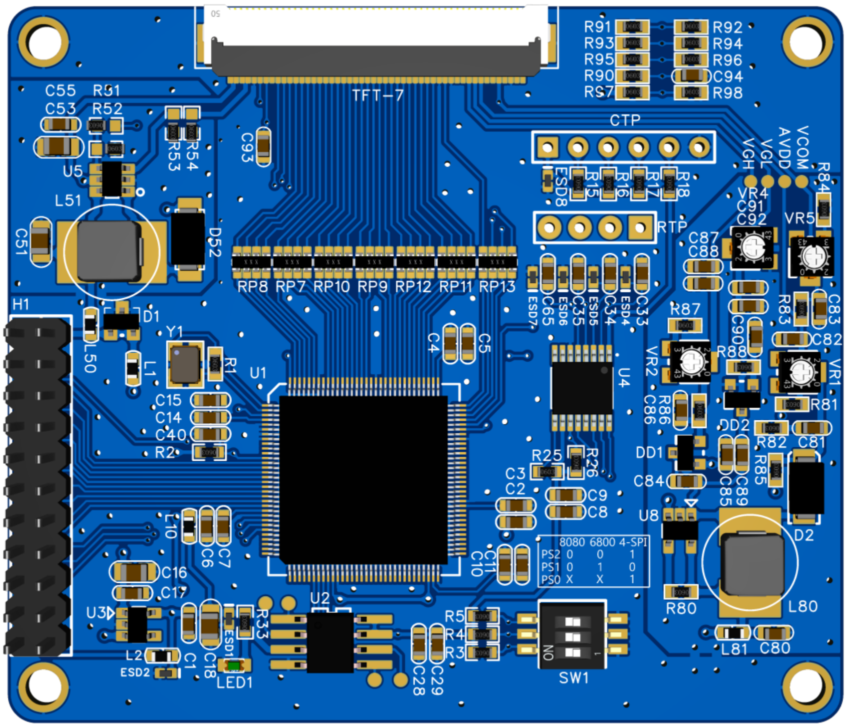


Figure 1-2: LT7683/LT7381 Controller Board for 50Pin LCD

(Dimension: 70mm x 60mm x 4.6mm (2.76" x 2.36" x 0.18"))

STM32F103RCT6 MCU Board

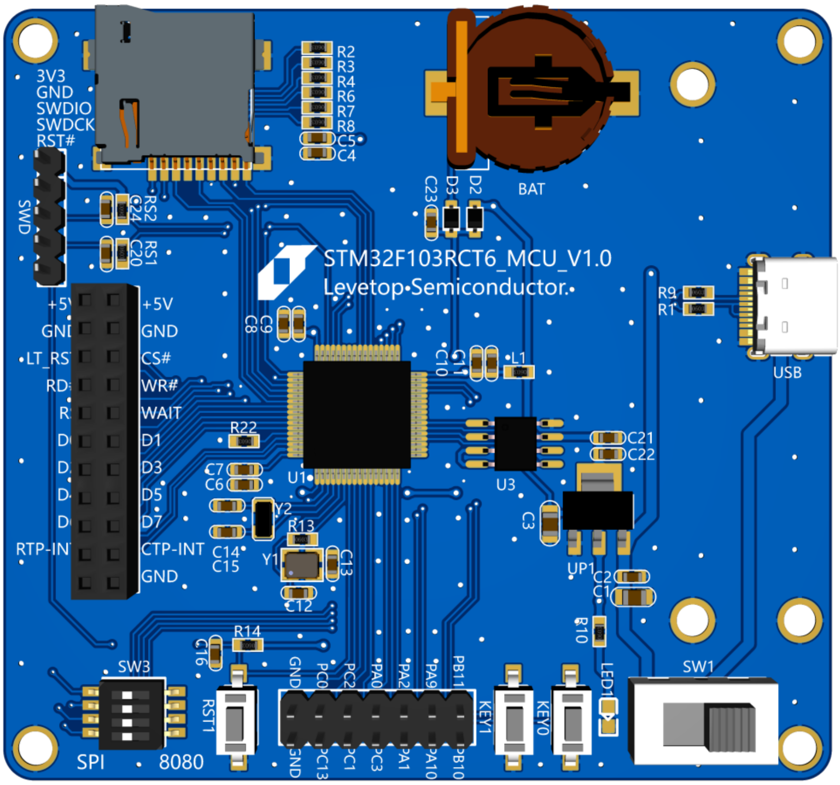
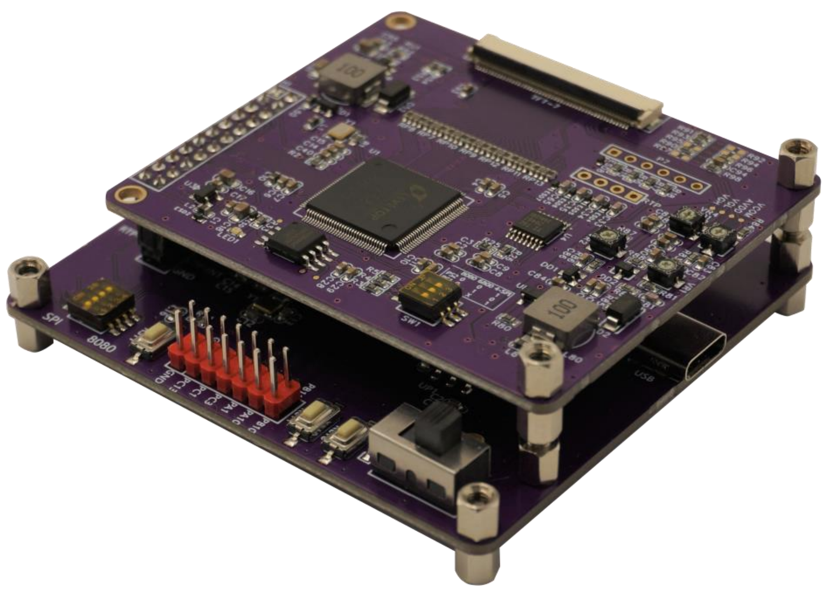


Figure 1-3: STM32F103RCT6 MCU Board

(Dimension: 77.3mm x 72mm x 4.6mm (3.04” x 2.83” x 0.18”))

MCU Board + Controller Board



2. Quick Start

2-1 Connect MCU Board to Controller Board

When connecting the MCU board to Levetop controller board, note the followings:

(1) MCU board must provide a set of 3.3V/GND and a set of 5V/GND to the controller board.

(2) MCU board should connect to the controller board by either 4-wire SPI or 8bits Parallel

Note: The DIP switches on both MCU board and Controller board must be set according to the communication interface (SPI, 8080, or 6800)

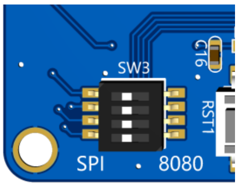


Figure 2-1 MCU Board: DIP switch settings

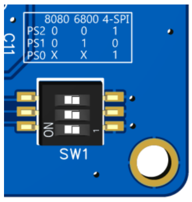


Figure 2-2 Controller Board: DIP switch settings

2-2 Test the demo program

Well documented demo code is available for design references. Our dedicated engineering team is also ready to assist you to best utilize Levetop's TFT controller.

Please download the demo code at https://www.levetop.tw/en/index.html, or contact Levetop for it.

```c
int main(void)
{
    NVIC_SetVectorTable(0x8008000,0);   // Set the program to start running at 0x8008000
    SystemInit();                       // Initialize STM32 system clock
    delay_init();                       // Initialize delay function
    uart_init(115200);                  // Initialize Uart port - for printf function
    printf("App Mode\r\n");

    mcu_IF_LT768x();                    // Initialize the interface between STM32 and LT768 - SPI/Parallel

    // (1) Initialize LT768x
    LT768_Init();

    // (2) Setup display window parameters --------------------
    Main_Image_Start_Address(0);
    Main_Image_Width(LCD_XSIZE_TFT);
    Main_Window_Start_XY(0,0);
    Canvas_Image_Start_address(0);
    Canvas_image_width(LCD_XSIZE_TFT);
    Active_Window_XY(0,0);
    Active_Window_WH(LCD_XSIZE_TFT,LCD_YSIZE_TFT);
    //-------------------------------------------------------

    // Draw full screen in black color
    LT768_DrawSquare_Fill(0,0,LCD_XSIZE_TFT-1,LCD_YSIZE_TFT-1,Black);

    // Intialize the PWM for LCD backlight control
    LT768_PWM1_Init(1,0,200,100,100);

    // Turn on the TFT display
    Display_ON();

    // (3) Demo show
    Show();                             // Basic desmonstration

    while(1)
    {
        delay_ms(10);
    }
}
```

Note: The interface setting (SPI & 8bits Parallel) is defined in "if_port.h", and should be set as same as the actual hardware configuration.

2-3 SPI Flash

On LT7683 controller board, there is an 128Mb SPI Flash chip. Developers may store materials such as picture, animation, font, and audio to the flash chip in advance, and access to the flash to retrieve those data by simple DMA functions for further process when needed.

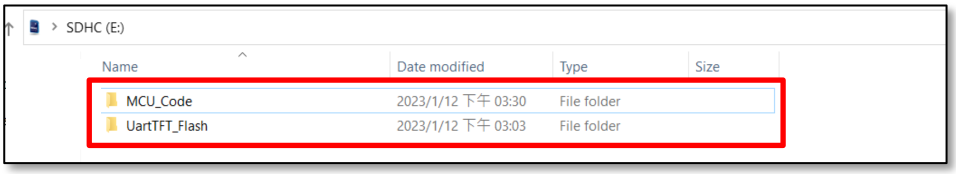
These materials must be converted to binary format before stored to the flash. Levetop provides a convenient tool, “UartTFT.exe”, to help developers generate the corresponding bin file for various materials. Please refer to the user manual, “UartTFT_V3.33_ENG.pdf”, for detail information.
Download the tool at: https://www.levetop.tw/data/UartTFT_V3.33.rar
Download the user manual at https://www.levetop.tw/data/UartTFT_V3.33_ENG.pdf

There are two ways to program the generated bin files to the flash:
(1) MCU writes data to flash → MCU reads the source bin file through a Uart tool, and then program it to the flash chip through LT7683. Please contact Levetop for the demo program.

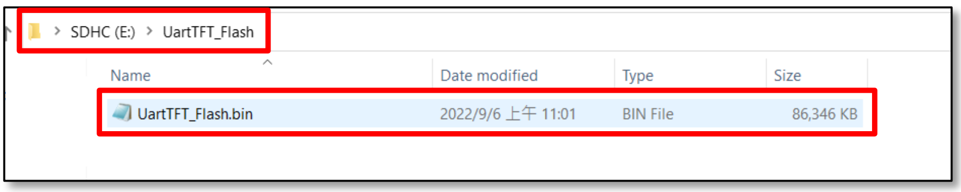
(2) Using Levetop MCU board → Follow the below procedure to program the flash:

A. Format an SD card by FAT32

B. Make a file directory, and name it as “UartTFT_Flash” in the SD card, as shown below. (Developers may also make another directory, and name it as “MCU_Code”. This one can be used for updating the MCU code.)



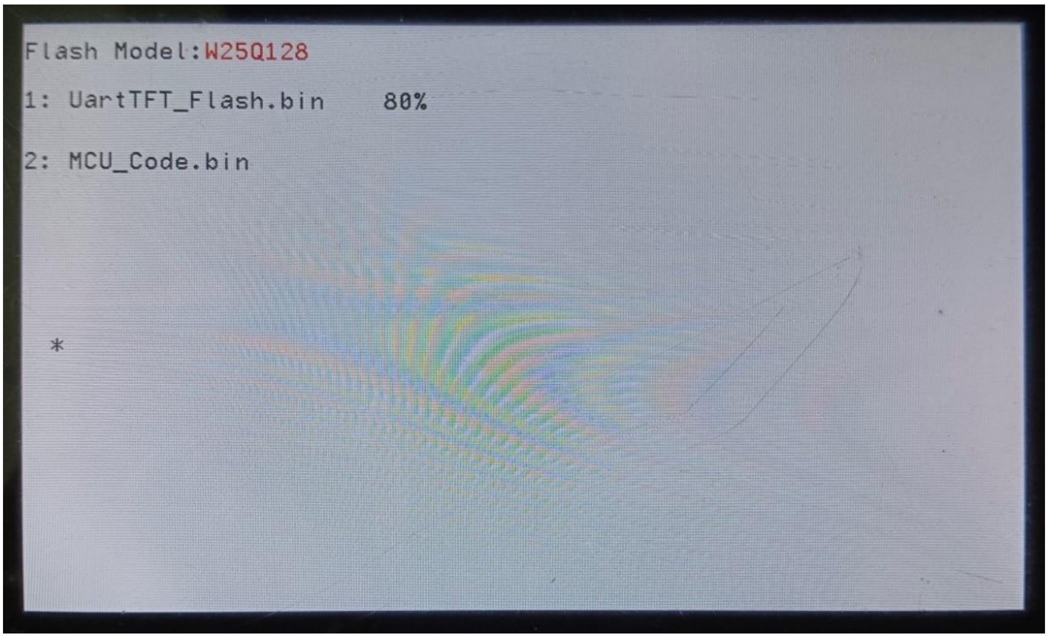
C. Store the bin file in the directory, and name the bin file as “UartTFT_Flash.bin”



D. Insert the SD card to the MCU board

E. Connect the MCU board to LT7683 controller board
– MCU provides a set of 5V/GND, and a set of 3.3V/GND to the controller board
– MCU connects to the controller board by either 4-wire SPI or 8bits Parallel

F. Power-on the MCU board, then the upload process will start automatically. The LCD will show the updating progress as shown below:



For further technical support, please feel free to contact us at info@levetop.cn

3. Libraries & Examples

The demo code and the examples in this section are based on the STM32 MCU board and LT7683 controller board. Please note that LT7683 needs to be well set and initialized before put to work:

(1) Go to if_port.h to set the interface between MCU and LT7683:

```
/*******************************************************************************
 * Copyright(c) 2023 Levetop Semiconductor Co., Ltd. All rights reserved
 * @file      if_port.h
 * @author    Levetop TFT Controller Application Team
 * @version   V1.0.0
 * @date      2023-02-24
 * @brief     Configure various communication interfaces
********************************************************************************/

#ifndef _if_port_h
#define _if_port_h
#include "sys.h"                // head file for STM32

//-------- Select only one host interface ----------
#define STM32_S_8080  0     // Use STM32 8080 host interface to drive LT768x
#define STM32_SPI_8   1     // Use STM32 SPI host interface to drive LT768x

// Select 8bit 8080 host interface
#if STM32_S_8080

#define LCD_CS_SET  GPIOC->BSRR |=1<<4    // chip select
#define LCD_RS_SET  GPIOC->BSRR |=1<<5    // Data/Command
#define LCD_WR_SET  GPIOA->BSRR |=1<<15   // Write data
#define LCD_RD_SET  GPIOC->BSRR |=1<<6    // Read data

#define LCD_CS_CLR  GPIOC->BRR |=1<<4     // chip select
#define LCD_RS_CLR  GPIOC->BRR |=1<<5     // Data/Command
#define LCD_WR_CLR  GPIOA->BRR |=1<<15    // Write data
#define LCD_RD_CLR  GPIOC->BRR |=1<<6     // Read data
```

(2) Go to LT768_Lib.h to set LCD resolution, color depth, and SPI Flash type:

```
/*******************************************************************************
 * Copyright(c) 2023 Levetop Semiconductor Co., Ltd. All rights reserved
 * @file      LT768_Lib.h
 * @author    Levetop TFT Controller Application Team
 * @version   V1.3
 * @date      2023-02-24
 * @brief     LT768x Function Libraries
********************************************************************************/

#ifndef _LT768_Lib_H
#define _LT768_Lib_H
#include "LT768.h"
#include "if_port.h"

#define LCD_16bit   1
#define LCD_24bit   0

#define NandFlash   0
#define NorFlash    1

#if NandFlash
#define DMA_BufferLayer  800*480*3*13    // SRAM buffer layer for LT768_DMA_24bit_Block
#endif

// External Xtal
#define XI_4M       0
#define XI_8M       0
#define XI_10M      1
#define XI_12M      0

// LCD Resolution
#define LCD_XSIZE_TFT   800
#define LCD_YSIZE_TFT   480
```

Set color depth

Set SPI flash type

Set LCD resolution

(3) Go to LT768_Lib.c, Set_LCD_Panel(), to set RGB color arrangement according to the LCD :

```
/*---------------------------------------------------------------------------------------*/
/* Function:     Set_LCD_Panel                                                            */
/*                                                                                        */
/* Parameters:   None                                                                     */
/* Returns:      None                                                                     */
/* Description:  Set LCD panel related parameters                                         */
/*---------------------------------------------------------------------------------------*/
void Set_LCD_Panel(void)
{
    #if STM32_S_8080
        Host_Bus_8bit();     // Host Bus 8bit
    #else
        Host_Bus_16bit();    // Host Bus 16bit
    #endif

    PCLK_Falling();          // REG[12h]: Falling edge
    //PCLK_Rising();

    HSYNC_Low_Active();      // REG[13h]:
    //HSYNC_High_Active();

    VSYNC_Low_Active();      // REG[13h]:
    //VSYNC_High_Active();

    DE_High_Active();        // REG[13h]:
    //DE_Low_Active();

    VSCAN_T_to_B();          // REG[12h]: Scan from top to bottom
    //VSCAN_B_to_T();        // From bottom to top

    LCD_HorizontalWidth_VerticalHeight(LCD_XSIZE_TFT ,LCD_YSIZE_TFT);
    LCD_Horizontal_Non_Display(LCD_HBPD);
    LCD_HSYNC_Start_Position(LCD_HFPD);
    LCD_HSYNC_Pulse_Width(LCD_HSPW);
    LCD_Vertical_Non_Display(LCD_VBPD);
    LCD_VSYNC_Start_Position(LCD_VFPD);
    LCD_VSYNC_Pulse_Width(LCD_VSPW);

    PDATA_Set_RGB();         // REG[12h]:Select RGB output
    //PDATA_Set_RBG();
    //PDATA_Set_GRB();
    //PDATA_Set_GBR();
    //PDATA_Set_BRG();
    //PDATA_Set_BGR();

#if  LCD_16bit
        TFT_16bit();
        RGB_16b_16bpp();
        Memory_16bpp_Mode();
        Select_Main_Window_16bpp();
```

(4) To initialize the MCU and LT7683, users may refer to the demo code provided by Levetop, as shown below:

```
int main(void)
{
    SystemInit();                       // Initialize STM32 system clock
    delay_init();                       // Initialize delay function
    uart_init(115200);                  // Initialize Uart port - for printf function
    printf("App Mode\r\n");

    mcu_IF_LT768x();                    // Initialize the interface between STM32 and LT768 - SPI/Parallel

    // (1) Initialize LT768x
    LT768_Init();

    // (2) Setup display window parameters --------------------
    Main_Image_Start_Address(0);
    Main_Image_Width(LCD_XSIZE_TFT);
    Main_Window_Start_XY(0,0);
    Canvas_Image_Start_address(0);
    Canvas_image_width(LCD_XSIZE_TFT);
    Active_Window_XY(0,0);
    Active_Window_WH(LCD_XSIZE_TFT,LCD_YSIZE_TFT);
    //------------------------------------------------------

    // Draw full screen in black color
    LT768_DrawSquare_Fill(0,0,LCD_XSIZE_TFT-1,LCD_YSIZE_TFT-1,Black);

    // Intialize the PWM for LCD backlight control
    LT768_PWM1_Init(1,0,200,100,100);

    // Turn on the TFT display
    Display_ON();

    // (3) Demo show
    Show();                             // Basic desmonstration

    while(1)
    {
        delay_ms(10);
    }
}
```

3-1 Display a full screen of Red color

Sample code:

```
/**************** Display a full screen of Red *****************/
// Setup display window parameters
Select_Main_Window_16bpp();
Main_Image_Start_Address(0);
Main_Image_Width(LCD_XSIZE_TFT);
Main_Window_Start_XY(0,0);
Canvas_Image_Start_address(0);
Canvas_image_width(LCD_XSIZE_TFT);
Active_Window_XY(0,0);
Active_Window_WH(LCD_XSIZE_TFT,LCD_YSIZE_TFT);

// Draw full screen of Red color
LT768_DrawSquare_Fill(0,0,LCD_XSIZE_TFT-1,LCD_YSIZE_TFT-1,Red);
/*******************************************************/
```

Before displaying a new image to the LCD, users must first setup the color depth:

Select_Main_Window_16bpp() → To set the main window as 16bit color depth

LT7683 has 128Mb display memory (SDRAM). The display memory can be set by three categories:

(1) Main Window: Used to define an area of SDRAM as a display area for the LCD panel. The data written to this area will be shown onto the LCD right away.

Main_Image_Start_Address(0); → The address of the main window in the SDRAM

Main_Image_Width(LCD_XSIZE_TFT); → Usually the width of the LCD resolution

Main_Window_Start_XY(0,0); → The start coordinate of the Main window

(2) Canvas Window: Used to define an area of SDRAM as a region to be written to when transferring data, such as DMA.

Canvas_Image_Start_address(0); → The address of the canvas window in the SDRAM

Canvas_image_width(LCD_XSIZE_TFT); → Usually the width of the LCD resolution

(3) Active Window: In the main window area, the LCD working area specified for drawing geometry or text display action

Active_Window_XY(0,0); → The start coordinate of the active window

Active_Window_WH(LCD_XSIZE_TFT,LCD_YSIZE_TFT); → Active window size

In the example here, since Main window and Canvas window share the same start address(0), when a new set of data is updated to Canvas window, the LCD panel will show the change instantly.

The below function is to draw a solid square from left-top (X1, Y1) to right-bottom (X2, Y2) in a designated filled color (ForegroundColor)

```
void LT768_DrawSquare_Fill
(
 unsigned short X1
,unsigned short Y1
,unsigned short X2
,unsigned short Y2
,unsigned long ForegroundColor
)
```

3-2 Display an image stored in the external SPI Flash

Sample code:

```
/************* Display an image stored in the external SPI Flash *****************/
// Setup display window parameters
Select_Main_Window_16bpp();
Main_Image_Start_Address(0);
Main_Image_Width(LCD_XSIZE_TFT);
Main_Window_Start_XY(0,0);
Canvas_Image_Start_address(0);
Canvas_image_width(LCD_XSIZE_TFT);
Active_Window_XY(0,0);
Active_Window_WH(LCD_XSIZE_TFT,LCD_YSIZE_TFT);

// Retrieve image data from SPI Flash, and store them to designated SDRAM area
LT768_DMA_24bit_Block(1, 0, 0, 0, LCD_XSIZE_TFT, LCD_YSIZE_TFT, LCD_XSIZE_TFT, 0, 0,
                      LCD_XSIZE_TFT);
/************************************************************************/
```

The DMA function is to retrieve data from the SPI Flash and transfer them to the designated Canvas area in SDRAM

```
void LT768_DMA_24bit_Block
(
 unsigned char SCS            → Select SPI Flash:  SCS: 0    SCS: 1
,unsigned char Clk            → SPI Clock = System Clock /{(Clk+1)*2}, Clk: 0 ~ 3
,unsigned short X1            → Starting X coordinate in Canvas window
,unsigned short Y1            → Starting Y coordinate in Canvas window
,unsigned short X_W           → Width of the data to be transmitted, e.g. picture width
,unsigned short Y_H           → Height of the data to be transmitted, e.g. picture height
,unsigned short P_W           → Picture width
,unsigned long Addr           → Starting address in Flash (to retrieve the data)
,unsigned long Layer          → Canvas address
,unsigned short Canvas_W      → Canvas width
)
```

In the example here, since Canvas address is the same as Main window address, the designated picture in SPI Flash will be shown onto the LCD after the DMA function is executed.

3-3 Memory copy in SDRAM

Sample code:

```
/******************* Memory copy in SDRAM ****************************/
// Define SDRAM layers → The LCD resolution is 800*480, and the color depth is 16bits
// Each layer = 800*480*2 bytes
#define  LAYER_0   0
#define  LAYER_1   800*480*2

// Setup display window parameters
Select_Main_Window_16bpp();
Main_Image_Start_Address(LAYER_0);
Main_Image_Width(800);
Main_Window_Start_XY(0,0);
Canvas_Image_Start_address(LAYER_1);
Canvas_image_width(800);
Active_Window_XY(0,0);
Active_Window_WH(800, 480);

// Retrieve image data from SPI Flash, and store them to LAYER_1
LT768_DMA_24bit_Block(1, 0, 0, 0, 800, 480, 800, 0, LAYER_1, 800);

// Copy LAYER_1 data (800*480*2bytes) to LAYER_0
LT768_BTE_Memory_Copy(
                        LAYER_1, 800, 0, 0,   // S0
                        LAYER_1, 800, 0, 0,   // S1
                        LAYER_0, 800, 0, 0,   // DT
                        0x0c,                 // ROP code, 0x0c -> DT = S0
                        800, 480              // Data area (width * height)
                        );
/*********************************************************************/
```

In the example here, two layers are specified in LT7683 SDRAM. Each layer represents 800*480*2 bytes of data block.

```
#define  LAYER_0   0          → Set the starting address of LAYER_0 to 0
#define  LAYER_1   800*480*2  → Set the starting address of LAYER_1 to 800*480*2
```

The DMA function is used to retrieve the data from the SPI Flash, and then transfer these data to LAYER_1 of SDRAM

```
LT768_DMA_24bit_Block(1, 0, 0, 0, 800, 480, 800, 0, LAYER_1, 800);
```

The BTE_Memory_Copy function is used to copy designated data to the DT layer in SDRAM.

```
void LT768_BTE_Memory_Copy
(
unsigned long S0_Addr      → Starting address (SDRAM) of S0 picture
,unsigned short S0_W       → Width of the S0 picture
,unsigned short XS0        → Left-top X coordinate of the S0 picture
,unsigned short YS0        → Left-top Y coordinate of the S0 picture
,unsigned long S1_Addr     → Starting address (SDRAM) of S1 picture
,unsigned short S1_W       → Width of the S1 picture
,unsigned short XS1        → Left-top X coordinate of the S1 picture
,unsigned short YS1        → Left-top Y coordinate of the S1 picture
,unsigned long Des_Addr    → Starting address (SDRAM) of DT picture
,unsigned short Des_W      → Width of the DT picture
,unsigned short XDes       → Left-top X coordinate of the DT picture
,unsigned short YDes       → Left-top Y coordinate of the DT picture
,unsigned int ROP_Code     → Memory operation mode
,unsigned short X_W        → Width of data to be transferred
,unsigned short Y_H        → Height of data to be transferred
)
```

The BTE function is combined with a ROP code for performing designated operations. In the example here, ROP_Code = 0x0c, which means transferring S0 (LAYER_1) data to DT (LAYER_0). Since LAYER_0 is set to be the Main Window layer, the transferred data will be shown onto the LCD.

The complete ROP code and definition is listed below for reference:

ROP Function Code REG[91h] bit[7:4]	Function Description (Boolean)
0000b	0 (Blackness)
0001b	~S0・~S1 or ~ (S0+S1)
0010b	~S0・S1
0011b	~S0
0100b	S0・~S1
0101b	~S1
0110b	S0^S1
0111b	~S0+~S1 or ~ (S0・S1)
1000b	S0・S1
1001b	~ (S0^S1)
1010b	S1
1011b	~S0+S1
1100b	S0
1101b	S0+~S1
1110b	S0+S1
1111b	1 (Whiteness)

3-4 Scrolling Picture

Sample Code:

```
/************************ Scrolling Picture ****************************/
// The LCD resolution is 800*480, and the color depth is 16bits
void Scrolling_Picture(void)
{
    Select_Main_Window_16bpp();
    Main_Image_Start_Address(LAYER_0);
    Main_Image_Width(LCD_XSIZE_TFT);
    Main_Window_Start_XY(0,0);
    Canvas_Image_Start_address(LAYER_0);
    Canvas_image_width(LCD_XSIZE_TFT);
    Active_Window_XY(0,0);
    Active_Window_WH(LCD_XSIZE_TFT,LCD_YSIZE_TFT);

    // Load a 800x480 picture to Main window layer to show it as the background
    // The picture starting address in SPI Flash is 0, and the data size is 800*480*2
    LT768_DMA_24bit_Block(1, 0, 0, 0, LCD_XSIZE_TFT, LCD_YSIZE_TFT, LCD_XSIZE_TFT, 0,
                          LAYER_0, LCD_XSIZE_TFT);

    // Load the 420x322 picture to LAYER_3
    Canvas_Image_Start_address(LAYER_3);
    Canvas_image_width(420);

    // Load a small picture from the SPI Flash. Starting address: 0x000bb800
    // The background picture occupies 800*480*2 = 0x000bb800 bytes
    LT768_DMA_24bit_Block(1, 0 ,0 ,0, 420, 322, 420, 0x000bb800, LAYER_3, 420);

    while(1)
    {
        // Use LAYER_2 as the buffer layer of LAYER_3, and move the data location every
        // 3 rows at a time
        //
        // 1. Copy the top 3 rows data of the scrolling picture on LAYER_3 to the bottom
        // of LAYER_2
        LT768_BTE_Memory_Copy( LAYER_3, 420, 0, 0,
                               LAYER_3, 420, 0, 0,
                               LAYER_2, 420, 0, 322-3,
                               0x0c, 420, 3);

        // 2. Copy the rest of the picture data on LAYER_3 to the top of LAYER_2
        LT768_BTE_Memory_Copy( LAYER_3, 420, 0, 3,
                               LAYER_3, 420, 0, 3,
                               LAYER_2, 420, 0, 0,
                               0x0c, 420, 322-3);
```

```
    // 3. Copy the processed data back to LAYER_3, for next loop processing
    LT768_BTE_Memory_Copy( LAYER_2, 420, 0, 0,
                           LAYER_2, 420, 0, 0,
                           LAYER_3, 420, 0, 0,
                           0x0c, 420, 322);

    // 4. Copy the processed data to LAYER_0 for displaying
    LT768_BTE_Memory_Copy( LAYER_2, 420, 0, 0,
                           LAYER_2, 420, 0, 0,
                           LAYER_0, 800, 190, 90,
                           0x0c, 420, 322);
  }
}

/******************************************************************************/
```

In this example, three SDRAM layers are used.

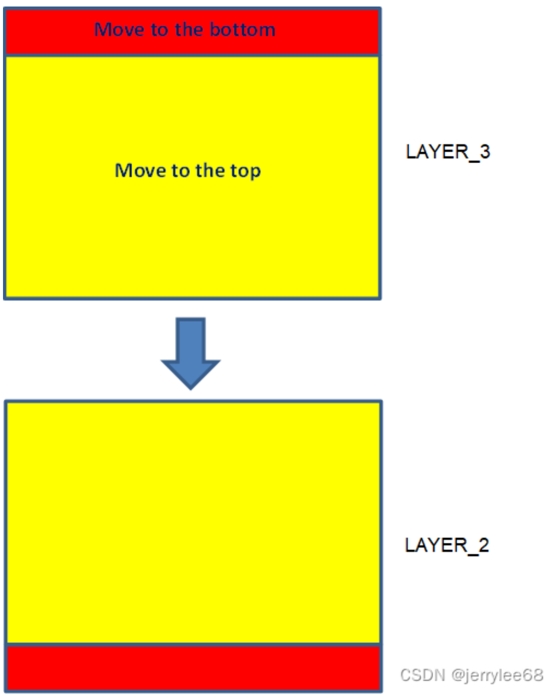
LAYER_0: Main window layer

LAYER_3: buffer layer for storing the data of the scrolling picture.

LAYER_2: buffer layer for storing the scrolled result

An 800x480 picture is loaded to LAYER_0 and used as the background, and a 420x322 picture is loaded to LAYER_3.

After the pictures are loaded to LT7683 SDRAM layers, we can then use BTE_MEMORY_COPY function to move the picture data, and create the scrolling effect.
The concept is as shown below:



The scrolling steps are implemented as described below:

Step 1: Copy the top 3 rows of the picture stored in LAYER_3 to the bottom of LAYER_2

Step 2: Copy the rest of the rows of the picture stored in LAYER_3 to the top of LAYER_2

Step 3: Copy the new formed picture data in LAYER_2 to LAYER_3

Step 4: Copy the new formed picture data in LAYER_2 to LAYER_0

Step 5: Run Step 1 ~ 4 in loop to form the scrolling effect.

Users may adjust the number of rows in Step 1 and Step 2 to control the scrolling speed.

3-5 Picture in Picture

Sample Code:

```
/*************************Picture in Picture******************************/
void PIP_Demo(void)
{
        Select_Main_Window_16bpp();
        Main_Image_Start_Address(LAYER_0);
        Main_Image_Width(LCD_XSIZE_TFT);
        Main_Window_Start_XY(0,0);
        Canvas_Image_Start_address(LAYER_0);
        Canvas_image_width(LCD_XSIZE_TFT);
        Active_Window_XY(0,0);
        Active_Window_WH(LCD_XSIZE_TFT,LCD_YSIZE_TFT);

        // Load the background picture to Main window layer (LAYER_0)
        LT768_DMA_24bit_Block(1, 0, 0, 0, LCD_XSIZE_TFT,LCD_YSIZE_TFT,LCD_XSIZE_TFT,0,
                                LAYER_0, LCD_XSIZE_TFT);

        // Load 1st small picture, starting address (in SPI Flash): 0x000bb800
        Canvas_Image_Start_address(LAYER_1);
        Canvas_image_width(136);
        LT768_DMA_24bit_Block(1, 0, 0, 0, 136, 136, 136, 0x000bb800, LAYER_1, 136);

        // Load 2nd small picture, starting address (in SPI Flash): 0x000c4448
        Canvas_Image_Start_address(LAYER_2);
        Canvas_image_width(276);
        LT768_DMA_24bit_Block(1, 0, 0, 0, 276, 206, 276, 0x000c4448, LAYER_2, 276);

        // Initialize PIP
        LT768_PIP_Init(1, 1, LAYER_1, 0, 0, 136, 0, 173, 136, 136);       // PIP-1
        LT768_PIP_Init(1, 2, LAYER_2, 0, 0, 276, 524, 137, 276, 206);   //  PIP-2

        // Display PIP-1 and PIP-2
        LT768_Set_DisWindowPos (1, 1, 100, 173);     // Show PIP-1
        LT768_Set_DisWindowPos (1, 2, 524, 137);     // Show PIP-2
}
/**********************************************************************/
```

LT7683 supports 2 sets of Picture-in-Picture feature (PIP-1 & PIP-2). Users can display sub-image on the main screen without overwriting the image data of the main display window. If PIP-1 and PIP-2 are overlapping, then the PIP-1 image is always on the top of PIP-2.

The PIP function must be initialized before use.

```
void LT768_PIP_Init             // Initialize Picture-In-Picture function
(
 unsigned char On_Off           // 0: Disable   1: Enable
,unsigned char Select_PIP       // 1: Use PIP1  2: Use PIP2
,unsigned long PAddr            // Starting address of the PIP
,unsigned short XP              // X coordinate of the PIP, must be dividable by 4
,unsigned short YP              // Y coordinate of the PIP, must be dividable by 4
,unsigned long ImageWidth       // Width of the image
,unsigned short X_Dis           // X coordinate of the display window
,unsigned short Y_Dis           // Y coordinate of the display window
,unsigned short X_W             // Width of the display window, must be dividable by 4
,unsigned short Y_H             // Height of the display window, must be dividable by 4
)
```

After initialized, the PIP pictures can be displayed by below function:

```
void LT768_Set_DisWindowPos     // Set the PIP display position
(
 unsigned char On_Off           // 0: Disable   1: Enable
,unsigned char Select_PIP       // 1: Use PIP1  2: Use PIP2
,unsigned short X_Dis           // X coordinate of the display window
,unsigned short Y_Dis           // Y coordinate of the display window
)
```

3-6 Pop up window with dimming background

Sample Code:

```
/****************Pop up window with dimming background*******************/
void BTE_Demo_Alpha_Blending_1(void)
{
        Select_Main_Window_16bpp();
        Main_Image_Start_Address(LAYER_0);
        Main_Image_Width(LCD_XSIZE_TFT);
        Main_Window_Start_XY(0,0);
        Canvas_Image_Start_address(LAYER_0);
        Canvas_image_width(LCD_XSIZE_TFT);
        Active_Window_XY(0,0);
        Active_Window_WH(LCD_XSIZE_TFT,LCD_YSIZE_TFT);

        // Clear display layer (SDRAM)
        LT768_DrawSquare_Fill(0, 0, LCD_XSIZE_TFT, LCD_YSIZE_TFT, Black);

        // Retrieve the background picture from flash and display it to the LCD
        Canvas_Image_Start_address(LAYER_3);
        LT768_DMA_24bit_Block(1, 0, 0, 0, LCD_XSIZE_TFT, LCD_YSIZE_TFT, LCD_XSIZE_TFT,
                              0x00000000, LAYER_3, LCD_XSIZE_TFT);

        LT768_BTE_Memory_Copy(
            LAYER_3, LCD_XSIZE_TFT, 0, 0,       // S0
            LAYER_3, LCD_XSIZE_TFT, 0, 0,       // S1
            LAYER_0, LCD_XSIZE_TFT, 0, 0,       // DT
            0x0c,                               // ROP code, 0x0c -> DT = S0
            LCD_XSIZE_TFT, LCD_YSIZE_TFT        // Data area (width * height)
                            );
        delay_ms(5000);  // Show the background picture for bout 5 seconds

        // Write "black" data to LAYER_4
        Canvas_Image_Start_address(LAYER_4);
        LT768_DrawSquare_Fill(0, 0, LCD_XSIZE_TFT, LCD_YSIZE_TFT, Black);

        // Background dimming effect by BTE_Alpha_Blending
        // Blend LAYER_3 and LAYER_4 data and then transfer the result to LAYER_0
        BTE_Alpha_Blending( LAYER_3, LCD_XSIZE_TFT, 0, 0,      // S0
                            LAYER_4, LCD_XSIZE_TFT, 0, 0,      // S1
                            LAYER_0, LCD_XSIZE_TFT, 0, 0,      // DT
                            LCD_XSIZE_TFT, LCD_YSIZE_TFT,      // Data area
                            20                                 // Alpha value (0~31)
                          );
```

```
            // Use DMA function to retrieve picture data (pop-up window) from Flash, and
            //   then transmit the data to LAYER_0 (Main window) for display
            Canvas_Image_Start_address(LAYER_0);
            LT768_DMA_24bit_Block(1, 0, 291, 134, 440, 332, 440, 0x0012c000, LAYER_0,
                                LCD_XSIZE_TFT);
}
/*************************************************************************/
```

The dimming effect in this example is basically performed by the function, BTE_Alpha_Blending:

```
void BTE_Alpha_Blending
(
unsigned long S0_Addr        → Starting address (SDRAM) of S0 picture
,unsigned short S0_W         → Width of the S0 picture
,unsigned short XS0          → Left-top X coordinate of the S0 picture
,unsigned short YS0          → Left-top Y coordinate of the S0 picture
,unsigned long S1_Addr       → Starting address (SDRAM) of S1 picture
,unsigned short S1_W         → Width of the S1 picture
,unsigned short XS1          → Left-top X coordinate of the S1 picture
,unsigned short YS1          → Left-top Y coordinate of the S1 picture
,unsigned long Des_Addr      → Starting address (SDRAM) of DT picture
,unsigned short Des_W        → Width of the DT picture
,unsigned short XDes         → Left-top X coordinate of the DT picture
,unsigned short YDes         → Left-top Y coordinate of the DT picture
,unsigned short X_W          → Width of the active window
,unsigned short Y_H          → Height of the active window
,unsigned char alpha         → Alpha blending level (0~31 levels)
)
```

The above function will blend S0 and S1 image based on the designated alpha value, and then copy the result data to DT.
The alpha value can be set from 0 to 31. In the example, the higher the alpha value is set, the darker the combined image will be

Revision

Version History

Version	Date	Description
V1.0	2023/03/08	Preliminary Release
V1.1	2023/08/09	Adding “Libraries & Examples” section

Copyright Notice