



# LT768x

***High Performance TFT-LCD Graphics Controller***

---

## Data Sheet

V4.2

[www.levetop.tw](http://www.levetop.tw)

Levetop Semiconductor Co., Ltd.

## Contents

1. Introduction .....	7
1.1 Internal Block Diagram .....	7
1.2 System Block Diagram .....	8
1.3 Model Name .....	8
2. Features .....	9
3. Pin Assignment .....	11
4. Pin Description .....	13
4.1 LT7681/LT7683/LT7686 (128Pin-LQFP).....	13
4.1.1 Host Interface Select Signals (3 Pins).....	13
4.1.2 Host Parallel I/F Signals (22 Pins).....	13
4.1.3 MCU Serial I/F Signals (8 Pins).....	14
4.1.4 External Serial Flash / SPI Master Signals (5 Pins) .....	15
4.1.5 LCD Driver Signals (28 Pins) .....	16
4.1.6 PWM Output Signals (2 Pins).....	17
4.1.7 GPIO Signals (28 Pins) .....	18
4.1.8 Key-Matrix Signals (10 Pins) .....	18
4.1.9 Power and Clock Signals (23 Pins) .....	19
4.1.10 Reset and Test Signals (4 Pins).....	19
4.2 LT7680x-R (68Pin-QFN).....	20
4.2.1 Host Interface Select Signals (2 Pins).....	20
4.2.2 Host Serial I/F Signals (5 Pins) .....	20
4.2.3 LCD Driver Signals (22 Pins) .....	20
4.2.4 External Serial Flash / SPI Master Signals (5 Pins) .....	21
4.2.5 PWM Output Signals (2 Pins).....	21
4.2.6 GPIO Signals (7 Pins) .....	21
4.2.7 Reset (1 Pins) .....	21
4.2.8 Power and Clock Signals (26 Pins) .....	22
5. Electrical Characteristics .....	23
5.1 Absolute Maximum Ratings .....	23
5.2 Static Electrical Characteristics .....	23
5.3 Absolute Maximum Ratings(electrical sensitivity).....	25
6. Clock and Reset.....	27
6.1 Clock .....	27

6.2 Reset .....	30
6.2.1 Power-on Reset .....	30
6.2.2 External Reset.....	30
6.2.3 Software Reset.....	30
<b>7. Host Interface .....</b>	<b>31</b>
7.1 Parallel Host Interface .....	33
7.2 Serial Host Interface .....	36
7.3 Display Input Data Format .....	40
7.3.1 Input Data without Opacity (RGB).....	40
7.3.2 Input Data with Opacity ( $\alpha$ RGB).....	43
<b>8. Display Memory.....</b>	<b>45</b>
8.1 Display RAM Data Structure .....	46
8.1.1 8bpp Display Data (RGB 3:3:2).....	46
8.1.2 16bpp Display Data (RGB 5:6:5).....	46
8.1.3 24bpp Display Data (RGB 8:8:8).....	46
8.1.4 Index Display with Opacity ( $\alpha$ RGB 2:2:2:2).....	47
8.1.5 12bpp Display with Opacity ( $\alpha$ RGB 4:4:4:4).....	47
8.2 Color Palette RAM .....	47
<b>9. LCD Interface .....</b>	<b>48</b>
<b>10. Display Function .....</b>	<b>50</b>
10.1 Color Bar .....	50
10.2 Main Window .....	50
10.2.1 Configure Display Image Buffer .....	50
10.2.2 Write Image Data to Display Image Buffer .....	51
10.2.3 Display Main Window Image .....	52
10.3 Picture-In-Picture (PIP) .....	53
10.3.1 PIP Window Setting .....	53
10.3.2 PIP Display Position and PIP Image Position.....	54
10.4 Image Rotate and Mirror .....	55
<b>11. Geometric Drawing Engine .....</b>	<b>58</b>
11.1 Drawing Circle and Ellipse .....	58
11.2 Drawing Curve.....	59
11.3 Drawing Rectangle .....	60
11.4 Draw Line .....	61
11.5 Drawing Triangle .....	62
11.6 Drawing Rounded-Rectangle .....	63

<b>12. Block Transfer Engine (BTE) .....</b>	<b>64</b>
12.1 BTE Basic Settings .....	66
12.2 Color Palette RAM .....	67
12.3 BTE Operation Overview .....	68
12.3.1 MCU Write with ROP.....	68
12.3.2 Memory Copy with ROP.....	68
12.3.3 Solid Fill.....	68
12.3.4 Pattern Fill.....	68
12.3.5 Pattern Fill with Chroma Key.....	68
12.3.6 MCU Write with Chroma Key .....	68
12.3.7 Memory Copy with Chroma Key.....	68
12.3.8 MCU Write with Color Expansion .....	68
12.3.9 Memory Copy with Color Expansion .....	69
12.3.10 Memory Copy with Opacity.....	69
12.3.11 MCU Write with Opacity .....	69
12.4 BTE Memory Access Method .....	70
12.5 BTE Chroma Key (Transparency Color) Function .....	70
12.6 BTE Operation Detail.....	71
12.6.1 MCU Write with ROP.....	71
12.6.2 Memory Copy (move) with ROP.....	72
12.6.3 MCU Write with Chroma Key (w/o ROP).....	74
12.6.4 Memory Copy with Chroma Key (w/o ROP) .....	75
12.6.5 Pattern Fill with ROP .....	76
12.6.6 Pattern Fill with Chroma Key .....	77
12.6.7 MCU Write with Color Expansion .....	78
12.6.8 MCU Write with Color Expansion and Chroma key .....	81
12.6.9 Memory Copy with Opacity .....	82
12.6.10 MCU Write with Opacity .....	86
12.6.11 Memory Copy with Color Expansion .....	87
12.6.12 Memory Copy with Color Expansion and Chroma Key.....	89
12.6.13 Solid Fill.....	90
<b>13. Display Text .....</b>	<b>91</b>
13.1 Internal CGROM .....	91
13.2 User-defined Character Graphic (UCG) .....	94
13.2.1 8*16 UCG Data Format.....	94
13.2.2 16*16 UCG Data Format.....	95
13.2.3 12*24 UCG Data Format.....	96

13.2.4 24*24 UCG Data Format .....	97
13.2.5 16*32 UCG Data Format .....	98
13.2.6 32*32 UCG Data Format .....	99
13.2.7 Initialize CGRAM from MCU .....	100
13.2.8 Initialize CGRAM from Serial Flash .....	101
13.3 Character Rotation by 90 Degree .....	102
13.4 Size Enlargement .....	102
13.5 Background Transparency .....	103
13.6 Character Full-Alignment .....	103
13.7 Automatic Line Feed .....	104
13.8 Cursor .....	104
13.8.1 Text Cursor .....	104
13.8.2 Graphic Cursor .....	106
<b>14.Pulse Width Modulation (PWM) .....</b>	<b>108</b>
14.1 PWM Clock Source .....	108
14.2 PWM Output .....	109
<b>15.Serial Bus Master .....</b>	<b>111</b>
15.1 Power-on Display .....	111
15.2 SPI Master .....	115
15.3 Serial Flash Controller .....	117
15.3.1 External Serial Flash .....	120
15.3.2 DMA in Linear Mode for External Serial Flash .....	121
15.3.3 DMA in Block Mode for External Serial Flash .....	121
15.4 I2C Master .....	124
<b>16.Keypad-Scan .....</b>	<b>127</b>
16.1 Keypad-scan Operation .....	127
<b>17.GPIO Interface .....</b>	<b>131</b>
<b>18.Power Management .....</b>	<b>132</b>
18.1 Normal Mode .....	132
18.2 Standby Mode .....	132
18.3 Suspend Mode .....	133
18.4 Sleep Mode .....	133
<b>19.Register Description .....</b>	<b>134</b>
19.1 Status Register .....	134
19.2 Configuration Registers .....	136

19.3	PLL Setting Register .....	141
19.4	Interrupt Control Register.....	143
19.5	LCD Display Control Registers .....	148
19.6	Geometric Engine Control Registers.....	159
19.7	PWM Control Registers .....	168
19.8	Bit Block Transfer Engine (BTE) Control Registers.....	171
19.9	Serial Flash & SPI Master Control Registers.....	179
19.10	Text Engine Registers.....	185
19.11	Power Management Control Register .....	190
19.12	Display RAM Control Register .....	191
19.13	I2C Master Register .....	195
19.14	GPIO Register .....	197
19.15	Keypad-scan Control Register .....	199
20.	Package Information .....	201
20.1	LT7681/LT7683/LT7686 (LQFP-128pin) .....	201
20.2	LT7680 (QFN-68pin).....	202
21.	Revision.....	203
22.	Copyright .....	203

## High Performance TFT-LCD Graphics Controller

### 1. Introduction

LT768x is a series of high-performance TFT-LCD graphic accelerated display controllers. Its main function is to assist MCU to display the contents on the TFT screen. LT768x provides graphic acceleration, PIP (picture-in-picture), geometry graphics and other functions. In addition to enhancing the display efficiency, LT768x can also ease the MCU loading on processing graphic data. LT768x supports 16/18/24-bits RGB interface TFT LCD with various display resolutions ranging from 320\*240 (QVGA) to 1280\*1024 (SXGA).

LT768x supports a variety of MCU interface, including SPI, I2C serial port, and 8-bit/16-bit parallel interface. In order to achieve multi-layers high-resolution display effect, it has a built-in 128Mb Display RAM, which can support assorted color depths from 1bit per pixel (2 gray shades) to



24bits per pixel (16M color), and alleviate the processing burden on MCU while displaying animation. With built-in geometric drawing engine, LT768X supports drawing points, lines, curves, ellipse, triangle, rectangle, rounded rectangle, and other functions. In addition, LT768x has an embedded hardware graphics acceleration engine (BTE), which provides command-type graphic operations such as screen rotation, flipping, mirroring, PIP and graphics blending, and transparent display. These functions can greatly enhance the display performance, and ease the processing burden on the MCU. Furthermore, users can use SPI interface to connect with MCU, and save the I/O ports of the MCU for other purposes. The powerful display performance of LT768x is ideal for embedded systems with TFT-LCD displays such as home appliances, industrial controls, electronic instruments, medical devices, human-machine interfaces, industrial equipment, inspection equipment, charging stations, multi-function machines, elevator, check-in gate, etc.

#### 1.1 Internal Block Diagram

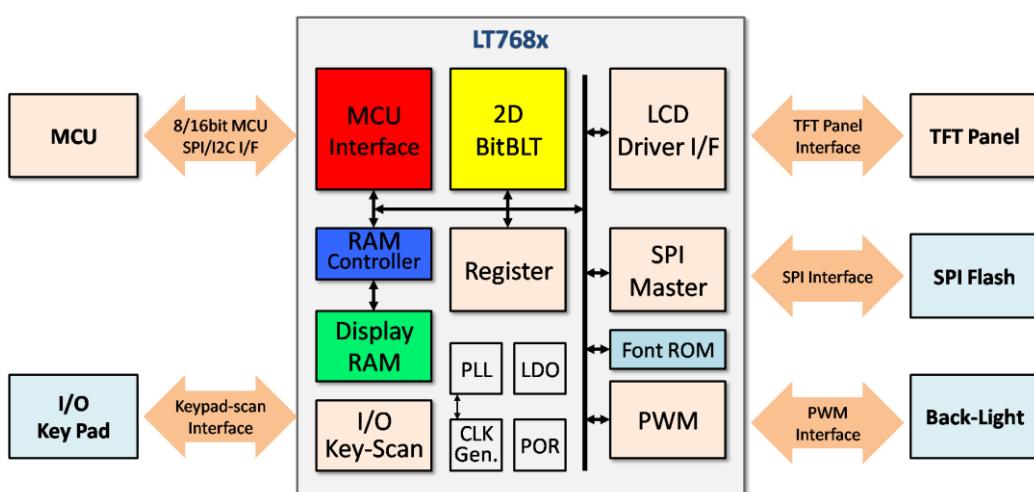


Figure 1-1: Internal Block Diagram

## 1.2 System Block Diagram

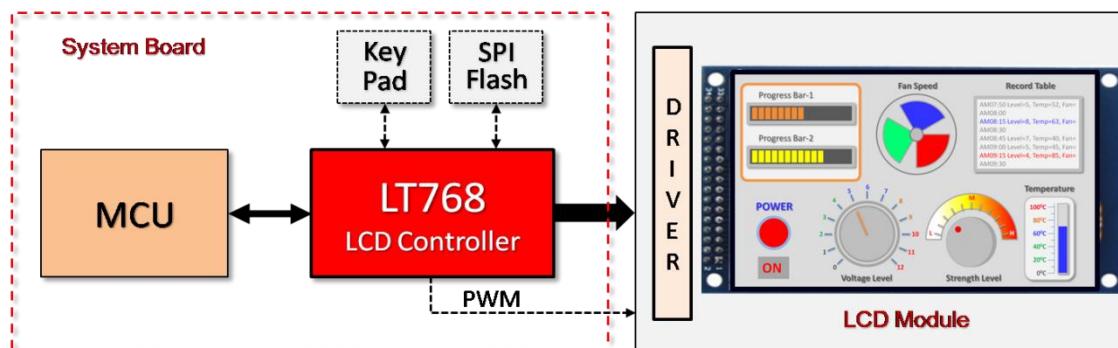


Figure 1-2: LT768x Designed on System Board

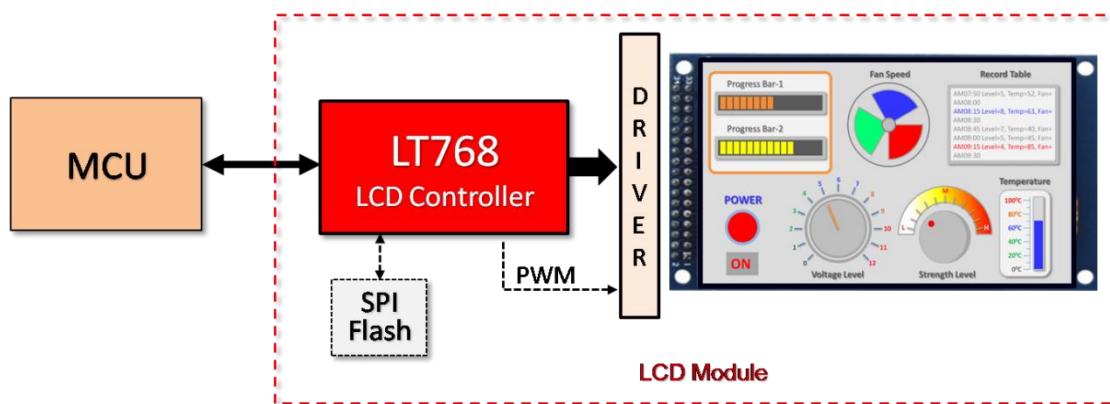


Figure 1-3: LT768x Designed on TFT-LCD Module

## 1.3 Model Name

Table 1-1: Model Selection

Model Name	Package	Embedded Display RAM	Resolution	Colors
LT7681	LQFP-128	128Mb	640*480	16.7M
LT7683	LQFP-128	128Mb	1024*768	16.7M
LT7686	LQFP-128	128Mb	1280*1024	16.7M
LT7680A-R	QFN-68 (8*8)	128Mb	1280*1024	65K / 262K
LT7680B-R	QFN-68 (8*8)	128Mb	480*320	65K / 262K

## 2. Features

### Host Interface

- Support 8/16bits Parallel Interface
  - Intel-8080 Bus Interface
  - Motorola-6800 Bus Interface
- Provide Insert Wait State Mechanism on Parallel Host Cycle
- Support I<sup>2</sup>C Bus Interface
- Support Various SPI Protocol. Ex. 3 or 4-wire SPI

### Display Data Formats

- 1bpp: Monochrome Data (1-bit/Pixel)
- 8bpp: RGB 3:3:2 (1-byte/Pixel)
- 16bpp: RGB 5:6:5 (2-byte/Pixel)
- 24bpp: RGB 8:8:8 (3-byte/Pixel or 4-byte/Pixel)
- Index 2:6 (64 Index Colors/Pixel with Opacity Attribute)
- aRGB 4:4:4:4 (4096 Colors/Pixel with Opacity Attribute)

### Support Panel and Resolution

- Support 16/18/24-bits RGB Interface Type Panel
- Support Resolution:
  - QVGA : 320\*240, 16/18/24-bit LCD Panel
  - WQVGA: 480\*272, 16/18/24-bit LCD Panel
  - VGA : 640\*480, 16/18/24-bit LCD Panel
  - WVGA : 800\*480, 16/18/24-bit LCD Panel
  - SVGA : 800\*600, 16/18/24-bit LCD Panel
  - QHD : 960\*540, 16/18/24-bit LCD Panel
  - WSVGA : 1024\*600, 16/18/24-bit LCD Panel
  - XGA : 1024\*768, 16/18/24-bit LCD Panel
  - SXGA : 1280\*1024, 16/18/24-bit LCD Panel

### Display Functions

- Multiple Display Buffer: Multi buffering allows the main display window to be switched among buffers. Multi buffering allows a simple animation display to be performed by switching the buffers
- Horizontal/Vertical Flip Display: Vertical Flip display functions are available for image data reads. PIP window will be disabled if flip display function is enabled

- Mirror and Rotation Functions are Available for Image Data Writes
- Provide four User-defined 32\*32 Pixels Graphic Cursor
- Virtual Display: Virtual display is available to show an image which is larger than LCD panel size. The image may scroll easily in any direction
- Picture-in-Picture (PIP) Display: Support two PIP windows area: Enabled PIP windows are always displayed on top of Main window. The PIP1 window is always on top of PIP2 window
- Wake-up Display: Wake-up Display is available to quickly show the display data stored in Display RAM. This feature is used when returning from the Standby mode or Suspend mode
- Initial Display: Embedded a tiny processor with 12 instructions to show display data stored in the serial flash without involving external MCU. It will auto execute after power-on, until the program is executed completely, then external MCU will retrieve the control.
- Color Bar: It could display color bar on panel directly. Default resolution is 640 dots by 480 dots

### Bit Block Transfer Engine (BTE)

- 2D BTE Engine
- Copy Image with Raster Operators
- Color Depth Conversion
- Solid Fill & Pattern Fill
- Provide User-defined Patterns with 8\*8 Pixels or 16\*16 Pixels
- Opacity (Alpha-Blend) Control: It blends two images and then generates a new image
  - Chroma-Keying Function: Mixes images with applying the specified RGB color according to the transparency rate
  - Window Alpha-Blending Function: Mixes two images according to the transparency rate in the specified region (fade-in and fade-out functions are available)
  - Dot Alpha-Blending Function: Mixes images according to the transparency rate when the target is a graphic image in the RGB format

## Display RAM (Frame Buffer)

- LT7680A-R / 7680B-R: Embedded 128Mb Display RAM
- LT7681 / 7683 / 7686: Embedded 128Mb Display RAM

## Shape Drawing Engine

- Provide Smart Drawing Features: Line, Rectangle, Triangle, Polygon, Poly-Line, Circle, Ellipse, Arc, and Rounded-Rectangle.

## Text Features

- Embedded 8\*16, 12\*24, 16\*32 Character Sets of ISO/IEC 8859-1/2/4/5
- User-defined Characters Support Half Size & Full Size for 8\*16, 12\*24 and 16\*32
- Programmable Text Cursor
- Character Enlargement Function \*1, \*2, \*3, \*4 for Horizontal/Vertical Direction
- Support Character to Rotate 90 Degrees

## PWM Interface

- Embedded Two 16bits Timers
- One 8-bit Pre-Scalars & One 4bits Divider
- Programmable Duty Control of Output Waveform (PWM)
- Auto Reload Mode or One-Shot Pulse Mode
- Dead-Zone Generator

## SPI Master Interface

- Provide DMA Function: Support Direct Data Transfer from External Serial Flash to Frame Buffer
- Compatible with Standard SPI Specifications
- Provide 16bytes Read FIFO and 16bytes Write FIFO
- Provide Interrupt when Tx FIFO is Empty and SPI Tx/Rx Engine is Idle

## I2C Interface

- Support Standard Mode (100kbps) and Fast Mode (400kbps)

## Key-Matrix Interface

- Support up to 5\*5 key matrix
- Programmable Scan Period
- Support Long Key & Repeat Key
- Support up to Two Keys Pressed Simultaneously
- Support Keypad-Scan Wakeup function

## Power Saving

- Support Three Kind of Power Saving Mode: Standby, Suspend, and Sleep Mode
- Support Wakeup Function by Host and External Event

## Clock Source

- Embedded Programmable PLL for Core Clock, LCD Panel's Pixel Clock, and Frame Buffer Clock

## Reset

- Provide Power On Reset Automatically
- Accept External Hardware Reset to Synchronize with System
- Software Command Reset

## Power Supply

- VDD: 3.3V +/- 0.3V
- Embedded 1.8V LDO

## Package

- LQFP 100/128-Pins, QFN 68-Pins

## Temperature

- -40°C~85°C

### 3. Pin Assignment

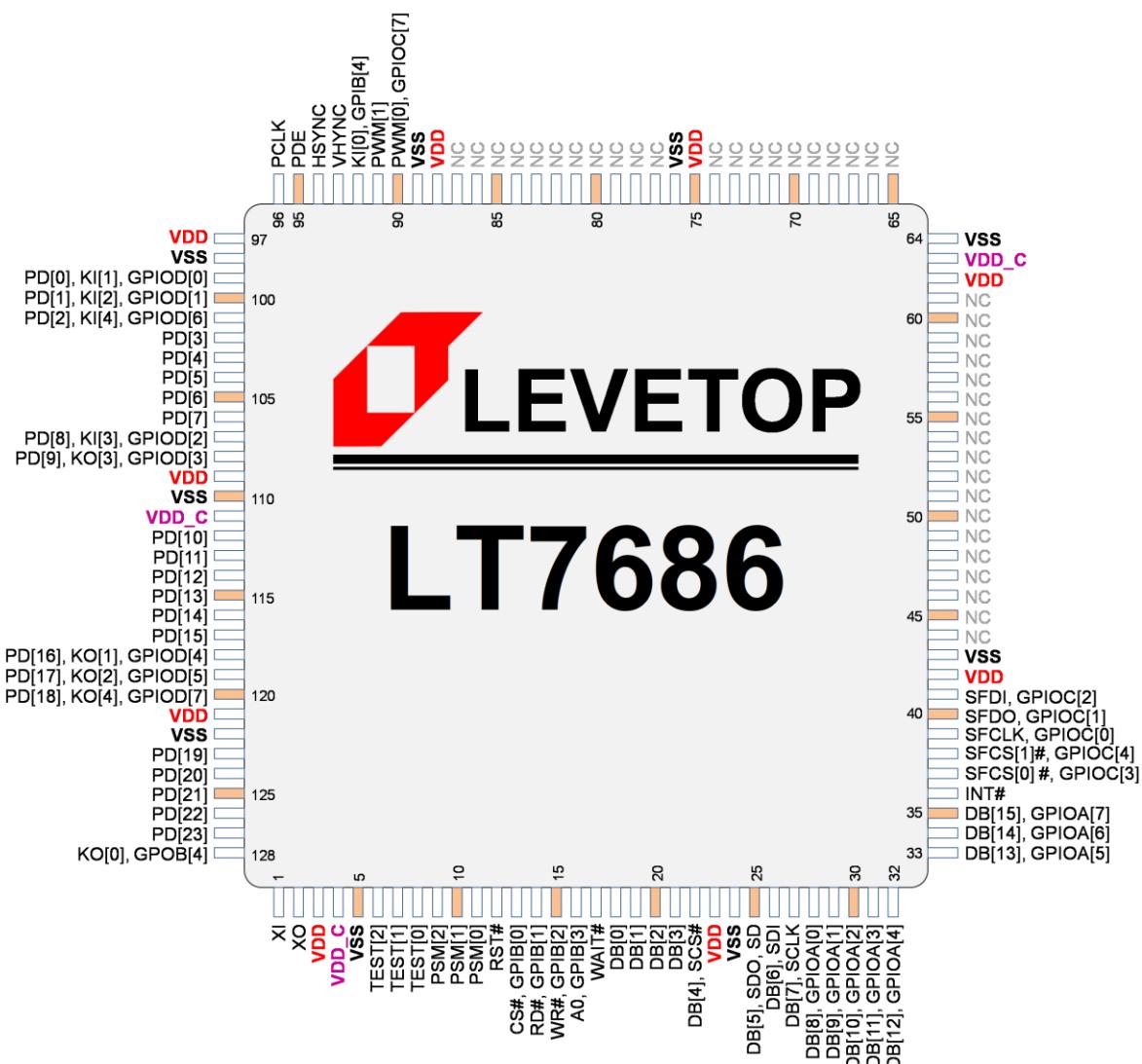
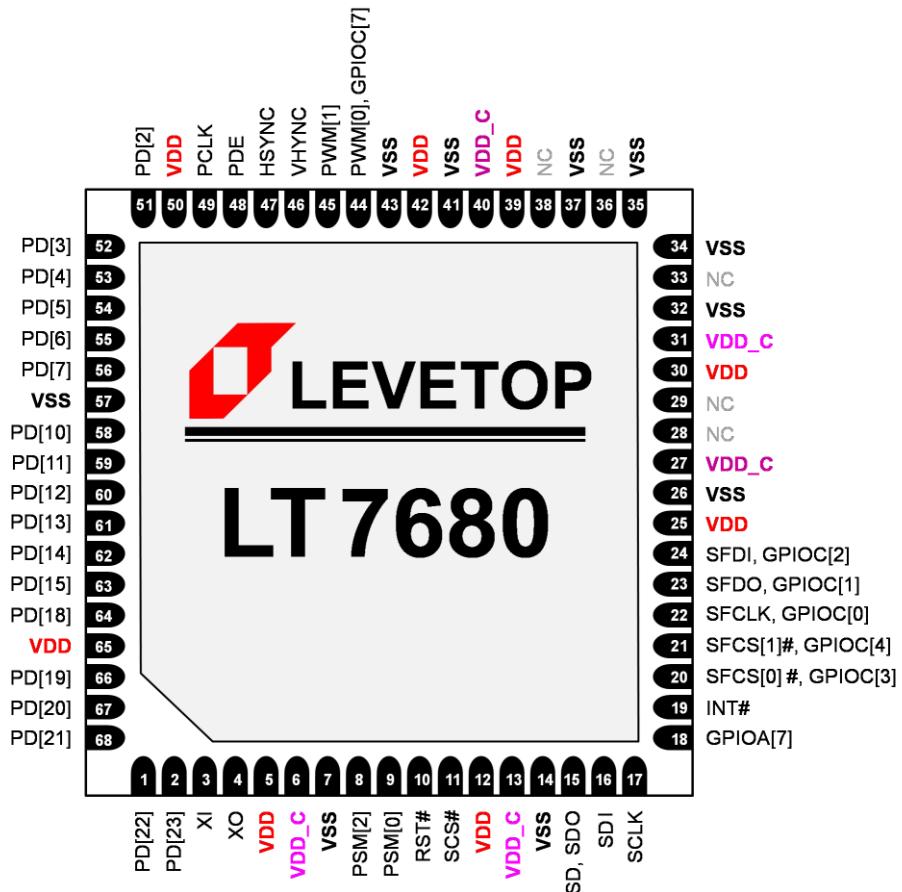


Figure 3-1: LT7681/LT7683/LT7686 Pin Assignment (LQFP-128Pin)



**Figure 3-2: LT7680A-R / LT7680B-R Pin Assignment (QFN-68Pin)**

## 4. Pin Description

### 4.1 LT7681/LT7683/LT7686 (128Pin-LQFP)

LT7681/7683/7686 are 128Pin LQFP type package. The following tables are pin description of these chips.

#### 4.1.1 Host Interface Select Signals (3 Pins)

Table 4-1: Host I/F Select Signals

Pin #	Pin Name	I/O	Pin Description													
9~11	PSM[2:0]	I	<b>Host Interface Selection</b> <table border="1"> <thead> <tr> <th>PSM[2:0]</th> <th>Host I/F Mode</th> </tr> </thead> <tbody> <tr> <td>0 0 X</td> <td>8bits or 16bits 8080 Parallel Interface Mode</td> </tr> <tr> <td>0 1 X</td> <td>8bits or 16bits 6800 Parallel Interface Mode</td> </tr> <tr> <td>1 0 0</td> <td>3-Wire SPI Mode</td> </tr> <tr> <td>1 0 1</td> <td>4-Wire SPI Mode</td> </tr> <tr> <td>1 1 X</td> <td>I2C Mode</td> </tr> </tbody> </table> <p>If Host interface is set to parallel mode, then PSM[0] pin is used as an external interrupt input pin.</p>		PSM[2:0]	Host I/F Mode	0 0 X	8bits or 16bits 8080 Parallel Interface Mode	0 1 X	8bits or 16bits 6800 Parallel Interface Mode	1 0 0	3-Wire SPI Mode	1 0 1	4-Wire SPI Mode	1 1 X	I2C Mode
PSM[2:0]	Host I/F Mode															
0 0 X	8bits or 16bits 8080 Parallel Interface Mode															
0 1 X	8bits or 16bits 6800 Parallel Interface Mode															
1 0 0	3-Wire SPI Mode															
1 0 1	4-Wire SPI Mode															
1 1 X	I2C Mode															

#### 4.1.2 Host Parallel I/F Signals (22 Pins)

Table 4-2: Host Parallel I/F Signals

Pin #	Pin Name	I/O	Pin Description
35~25, 22~18	DB[15:0]	IO	<b>Host Data Bus</b> These are data bus for data transfer between Host and LT768x. DB[15:8] will become GPIO (GPIOA[7:0]) when Host does not set to 8080/6800 16-bits data bus mode. DB[7:0] are multiplex pins that share with Serial Host control pins. When Host is set to serial mode, then DB[7:0] are defined as the control pins of serial host. Please refer to Host Interface section.
13	CS# GPIB[0]	I	<b>Chip Select Input</b> Low active chip select pin from Host. If host interface is set to serial mode, then this pin can be set as GPIB[0]. This pin has an internal pull-high resistor.
14	RD# EN GPIB[1]	I	<b>Read / Enable Input</b> <b>RD#:</b> When host interface is set to 8080 mode, this pin is a Read input signal, active low. <b>EN:</b> When host interface is set to 6800 mode, this pin is an Enable input signal, active high. If host interface is set to serial mode, then this pin can be set as GPIB[1]. This pin has an internal pull-high resistor.

Pin #	Pin Name	I/O	Pin Description
15	WR# RW# GPIB[2]	I	<p><b>Write / Read-Write Input</b>  <b>WR#:</b> When host interface is set to 8080 mode, this pin is a Write input signal, active low.  <b>RW#:</b> When host interface is set to 6800 mode, this pin is a Read-Write input signal. It actives high in 'Host's read cycle, and actives low in Host's write cycle.  If host interface is set to serial mode, then this pin will be set as GPIB[2]. This pin has an internal pull-high resistor.</p>
16	A0 GPIB[3]	I	<p><b>Command / Data Select Input</b>  The pin is used to select Command or Data cycle.  A0 = 0, Status Read or Command Write cycle is selected.  A0 = 1, Data Read or Data Write cycle is selected.  If host interface is set to serial mode, then this pin will be set as GPIB[3]. This pin has an internal pull-high resistor.</p>
36	INT#	O	<p><b>Interrupt Output Signal</b>  The interrupt output for indicating the status to the host.</p>
17	WAIT#	O	<p><b>Wait Output Signal</b>  When high, it indicates that the LT768x is ready to transfer data.  When low, then microprocessor is in wait state.</p>

#### 4.1.3 MCU Serial I/F Signals (8 Pins)

Table 4-3: Host Serial I/F Signals

Pin #	Pin Name	I/O	Pin Description
27	SCLK (DB[7])	I	<p><b>SPI or I2C Clock</b>  <b>SCLK:</b> Clock of 3-wire, 4-wire Serial or I2C interface.  This is a multiplex pin that shares with Parallel Host Data Bus DB[7].</p>
26	SDI I2C_SDA (DB[6])	I	<p><b>I2C Data / 4-wire SPI Data Input</b>  <b>SDI:</b> Data input pin of 4-wire SPI I/F. Connect to MCU's MOSI.  <b>I2C_SDA:</b> Bi-direction data pin of I2C I/F.  If using 3-Wire serial I/F, please connect this pin to GND. This is a multiplex pin that shares with Parallel Host Data Bus DB[6].</p>
25	SD SDO I2CA[5] (DB[5])	IO	<p><b>3-wire SPI Data / 4-wire SPI Data Output / I2C Slave Address Select</b>  <b>SD:</b> Bi-direction data pin of 3-wire SPI I/F.  <b>SDO:</b> Data output pin of 4-wire SPI I/F. Connect to MCU's MISO.  <b>I2CA[5]:</b> I2C device address bit[5] of I2C I/F.  This is a multiplex pin that shares with Parallel Host Data Bus DB[5].</p>

Pin #	Pin Name	I/O	Pin Description
22	SCS# I2CA[4] (DB[4])	I	<b>SPI Chip Select / I2C Slave Address Select</b> <b>SCS#</b> : Chip select pin for 3-wire and 4-wire serial I/F. <b>I2CA[4]</b> : I2C device address bit[4]. This is a multiplex pin that shares with Parallel Host Data Bus DB[4].
21~18	I2CA[3:0] (DB[3:0])	I	<b>I2C Slave Address Select</b> <b>I2CA[3:0]</b> : I2C device address bit [3:0]. These pins are not used in 3-Wire or 4-Wire I/F. Please connect them to GND. These are multiplex pins that share with Parallel Host Data Bus DB[3:0].

#### 4.1.4 External Serial Flash / SPI Master Signals (5 Pins)

Table 4-4: External Serial Flash Signals

Pin #	Pin Name	I/O	Pin Description
37	SFCS[0]# GPIOC[3]	IO	<b>Chip Select 0 for External Serial Flash or SPI device</b> SPI Chip select pin #0 for serial Flash or SPI device. If SPI master I/F is disabled then it can be programmed as GPIOC[3], and its default is input function.
38	SFCS[1]# GPIOC[4]	IO	<b>Chip Select 1 for External Serial Flash or SPI device</b> SPI Chip select pin #1 for serial Flash or SPI device. If SPI master I/F is disabled then it can be programmed as GPIOC[4], and its default is input function.
39	SFCLK GPIOC[0]	IO	<b>SPI Serial Clock</b> Serial clock output for serial Flash/ROM or SPI device. If SPI master I/F is disabled then it can be programmed as GPIO C[0], and its default is input function.
40	SFDO GPIOC[1]	IO	<b>Master Output Slave Input</b> <b>Single Mode</b> : Data input of serial Flash or SPI device. For LT768x, it is output. <b>Dual Mode</b> : The signal is used as bi-direction data #0(SIO0). Only valid in serial flash DMA mode. If SPI master I/F is disabled, then it can be programmed as GPIO C[1], and its default is input function.
41	SFDI GPIOC[2]	IO	<b>Master Input Slave Output</b> <b>Single Mode</b> : Data output of serial Flash or SPI device. For LT768, it is input. <b>Dual Mode</b> : The signal is used as bi-direction data #1(SIO1). Only valid in serial flash DMA mode. If SPI master I/F is disabled, then it can be programmed as GPIOC[2], and its default is input function.

#### 4.1.5 LCD Driver Signals (28 Pins)

Table 4-5: LCD Driver Signals

Pin #	Pin Name	I/O	Pin Description			
127~123, 120~112, 108~99	PD[23:0]	IO	<b>LCD Panel Data Bus</b>			
			TFT-LCD Interface			
			11b (GPIO)	10b (16bits)	01b (18bits)	00b (24bits)
			PD[0]	GPIOD[0] / KI[1]		B0
			PD[1]	GPIOD[1] / KI[2]		B1
			PD[2]	GPIOD[6] / KI[4]		B0
			PD[3]	GPIOE[0]	B0	B1
			PD[4]	GPIOE[1]	B1	B2
			PD[5]	GPIOE[2]	B2	B3
			PD[6]	GPIOE[3]	B3	B4
			PD[7]	GPIOE[4]	B4	B5
			PD[8]	GPIOD[2] / KI[3]		G0
			PD[9]	GPIOD[3] / KO[3]		G1
			PD[10]	GPIOE[5]	G0	G0
			PD[11]	GPIOE[6]	G1	G1
			PD[12]	GPIOE[7]	G2	G2
			PD[13]	GPIOF[0]	G3	G3
			PD[14]	GPIOF[1]	G4	G4
			PD[15]	GPIOF[2]	G5	G5
			PD[16]	GPIOD[4] / KO[1]		R0
			PD[17]	GPIOD[5] / KO[2]		R1
			PD[18]	GPIOD[7] / KO[4]		R0
			PD[19]	GPIOF[3]	R0	R1
			PD[20]	GPIOF[4]	R1	R2
			PD[21]	GPIOF[5]	R2	R3
			PD[22]	GPIOF[6]	R3	R4
			PD[23]	GPIOF[7]	R4	R5
These are multiplex pins that share with GPIO and Key-Matrix pins. If the setting of LCD I/F is 18bpp function mode, then PD[17:16 / 8:9 / 1:0] are defined as GPIO pins.						

Pin #	Pin Name	I/O	Pin Description
96	PCLK	O	<b>Panel Scan Clock</b> Generic TFT interface signal for panel scan clock. It derives from internal PLL.
93	VSYNC	O	<b>VSYNC Pulse</b> Generic TFT interface signal for vertical synchronous pulse.
94	Hsync	O	<b>Hsync Pulse</b> Generic TFT interface signal for horizontal synchronous pulse.
95	PDE	O	<b>Data Enable</b> Generic TFT interface signal for data valid or data enable.

#### 4.1.6 PWM Output Signals (2 Pins)

Table 4-6: PWM Output Signals

Pin #	Pin Name	I/O	Pin Description
90	PWM[0] INITDIS GPIOC[7] CCLK	IO	<b>PWM Output 0 / Initial Display Enable</b> <b>PWM[0]</b> : PWM's output signal. The output mode is decided by configuration register. This pin can be used as the control signal of TFT panel's back light. <b>INITDIS</b> : Pull-high this pin will enable Initial Display function. This pin has an internal pull-down, thus the Initial Display function is disabled by default in reset period. If PWM function is disabled, then it can be programmed as GPIO C[7], and the default is GPIOC[7] input function or output Core Clock - CCLK.
91	PWM[1]	IO	<b>PWM Output 1</b> PWM's output signal. The output mode and output function are decided by configuration register. This pin also can be used as the control signal of TFT panel's back light. When TEST[0] is set to high, then PWM[1] pin is external panel scan clock input

#### 4.1.7 GPIO Signals (28 Pins)

**Table 4-7: General Purpose I/O Signals**

Pin #	Pin Name	I/O	Pin Description
35~28	GPIOA[7:0]	IO	<b>GPIO A Group</b> These are general purpose I/O. These are multiplex pins that share with DB[15:8]. They are only available when the host interface is set to 8bits parallel mode or serial mode.
92, 128, 16~13	GPIB[4], GPOB[4], GPIB[3:0]	IO	<b>GPIO B Group</b> These are general purpose I/O. GPIB[3:0] are read only and available in serial host mode. GPIB[4] shares the same pin with KI[0]. GPOB[4] shares the same pin with KO[0]. GPIB[3:0] are multiplex pins that share with {A0, WR#, RD#, CS#}.
90, 38, 37, 41~39	GPIOC[7], GPIOC[4:0]	IO	<b>GPIO C Group</b> These are general purpose I/O. GPIOC are available when PWM and SPI Master functions are disabled. GPIOC[7] shares the same pin with PWM[0]. GPIOC[4:0] are multiplex pins that share with {SFCS1#, SFCS0#, SFDI, SFDO, SFCLK}
120, 101 119, 118 108, 107 100, 99	GPIOD[7:0]	IO	<b>GPIO D Group</b> These are general purpose I/O. GPIOD[7:0] are multiplex pins that share with PD[18, 2, 17, 16, 9, 8, 1, 0]. GPIOD[5,4,3,2,1,0] are available when LCD Panel interface is set to 16bits or 18bits. GPIOD[7,6] are available when LCD Panel interface is set to 16bits.

#### 4.1.8 Key-Matrix Signals (10 Pins)

**Table 4-8: Key-Matrix Signals**

Pin #	Pin Name	I/O	Pin Description
101, 107, 100, 99, 92	KI[4:0] I2CMCK	I	<b>Key-Matrix Data Pins</b> Keypad data inputs with internal pull-up resistor. KI[4:1] are multiplex pins that share with PD[8] and PD[2:0]. The Key-matrix function will be disable when LCD I/F is set to 24bits. KI[0] also provides the I2CMCK function of I2C Master.
120,108, 119,118, 128	KO[4:0] I2CMDA	O	<b>Key-Matrix Strobe Pins</b> Keypad strobe data outputs with Open-Drain. KO[4:1] are multiplex pins that share with PD[9] and PD[18:16]. KO[0] also provides the I2CMDA function of I2C Master.

#### 4.1.9 Power and Clock Signals (23 Pins)

Table 4-9: Power and Clock Signals

Pin #	Pin Name	I/O	Pin Description
1	XI	I	<b>Crystal / External Clock Input</b> This input pin is used for internal crystal circuit or external clock that generates clock source for PLL. It should be connected to external crystal or clock, and the suggested frequency is 4 ~ 12 MHz.
2	XO	O	<b>Crystal Output</b> This is an output pin for internal crystal circuit. It should be connected to external crystal circuit.
4, 63, 111	VDD_C	PWR	<b>Internal LDO Output</b> These pins must connect 1uF and 0.1uF capacitor to ground.
3, 23, 42, 62, 75, 88, 97, 109, 121	VDD	PWR	<b>3.3V Power Pins</b>
5, 24, 43, 64, 76, 89 98, 110, 122	VSS	PWR	<b>Ground(GND) Pins</b>

#### 4.1.10 Reset and Test Signals (4 Pins)

Table 4-10: Reset and Test Signals

Pin #	Pin Name	I/O	Pin Description
12	RST#	I/O	<b>Reset Signal Input</b> This is an active low Reset pin for LT768x. To avoid noise interferences that cause fake reset behavior, this pin will not be active unless the reset signal lasts at least 256 OSC clocks.
6~8	TEST[2:0]	I	<b>Test Input</b> These pins are used for testing and normally connected to GND. If TEST[0] keeps high, the internal PLL will be disable and the system clock is supplied by external clock. If TEST[2:1] keep 01b, then the SPI Master signals will keep floating. This feature allows external device to program Serial Flash directly. (i.e. ISP, In-System-Programming)

## 4.2 LT7680x-R (68Pin-QFN)

LT7680 is a 68Pin QFN type package chip. The following tables are the pin function list. For detail pin descriptions, please refer to the previous section - Pin Description -1.

### 4.2.1 Host Interface Select Signals (2 Pins)

Table 4-11: Host I/F Select Signals

Pin #	Pin Name	I/O	Pin Description	
<b>Host Interface Selection</b>				
8~9	PSM[2] PSM[0]	I	PSM[2] PSM[0]	Host I/F Mode
			1 0	3-Wire SPI Mode
			1 1	4-Wire SPI Mode
The PSM[2] pin must connect to Hi.				

### 4.2.2 Host Serial I/F Signals (5 Pins)

Table 4-12: Host Serial I/F Signals

Pin #	Pin Name	I/O	Pin Description
17	SCLK	I	<b>SPI Clock</b>
16	SDI	I	<b>4-wire SPI Data Input</b>
15	SD SDO	IO	<b>3-wire SPI Data</b> <b>4-wire SPI Data Output</b>
11	SCS#	I	<b>SPI Chip Select</b>
19	INT#	O	<b>Interrupt Output Signal</b>

### 4.2.3 LCD Driver Signals (22 Pins)

Table 4-13: LCD Driver Signals

Pin #	Pin Name	I/O	Pin Description
2~1, 68~66, 64, 63~58, 56~51	PD[23:18], PD[15:10], PD[7:2],	IO	<b>LCD Panel Data Bus</b>
49	PCLK	O	<b>Panel Scan Clock</b>
46	VSYNC	O	<b>VSYNC Pulse</b>
47	Hsync	O	<b>Hsync Pulse</b>
48	PDE	O	<b>Data Enable</b>

#### 4.2.4 External Serial Flash / SPI Master Signals (5 Pins)

Table 4-14: External Serial Flash Signals

Pin #	Pin Name	I/O	Pin Description
21~20	SFCS[1:0]#	IO	<b>Chip Select 0 for External Serial Flash or SPI device</b>
22	SFCLK	IO	<b>SPI Serial Clock</b>
23	SFDO	IO	<b>Master Output Slave Input</b>
24	SFDI	IO	<b>Master Input Slave Output</b>

#### 4.2.5 PWM Output Signals (2 Pins)

Table 4-15: PWM Output Signals

Pin #	Pin Name	I/O	Pin Description
45	PWM[0]	IO	<b>PWM Output 0 / Initial Display Enable</b>
44	PWM[1]	IO	<b>PWM Output 1</b>

#### 4.2.6 GPIO Signals (7 Pins)

Table 4-16: General Purpose I/O Signals

Pin #	Pin Name	I/O	Pin Description
18	GPIOA[7]	IO	<b>GPIO A Group</b>
44, 21, 20, 24, 23, 22	GPIOC[7] GPIOC[4:0]	IO	<b>GPIO C Group</b>

#### 4.2.7 Reset (1 Pins)

Table 4-17: Reset Signal

Pin #	Pin Name	I/O	Pin Description
10	RST#	I/O	<b>Reset Signal Input</b>

#### 4.2.8 Power and Clock Signals (26 Pins)

**Table 4-18: Power and Clock Signals**

Pin #	Pin Name	I/O	Pin Description
3	XI	I	<b>Crystal / External Clock Input</b>
4	XO	O	<b>Crystal Output</b>
6, 13, 27, 31, 40	VDD_C	PWR	<b>Internal LDO Output</b>
5, 12, 25, 30, 39, 42, 50, 65	VDD	PWR	<b>3.3V Power Pins</b>
7, 14, 26, 32, 34, 35, 37, 41, 43, 57,	VSS	PWR	<b>Ground(GND) Pins</b>
	Thermal Pad	-	The back of LT7689 Heat Sink Pad must be grounded.

## 5. Electrical Characteristics

### 5.1 Absolute Maximum Ratings

Table 5-1: Absolute Maximum Ratings

Symbol	Parameter	Value	Unit
V <sub>DD</sub>	Supply Voltage Range	-0.3 ~ 4.0	V
V <sub>IN</sub>	Input Voltage Range	-0.3 ~ VDD+0.3	V
V <sub>OUT</sub>	Output Voltage Range	-0.3 ~ VDD+0.3	V
P <sub>D</sub>	Power Dissipation	≤300	mW
T <sub>OPR</sub>	Operation Temperature Range	-45 ~ 85	°C
T <sub>ST</sub>	Storage Temperature	-45 ~ 125	°C
T <sub>SOL</sub>	Soldering Temperature	260	°C

**Note:**

If used beyond the absolute maximum ratings, LT768x may be permanently damaged. It is strongly recommended that the device is used within the electrical characteristics. If exposed to the condition not within the electrical characteristics, it may affect the reliability of the device. This specification does not guarantee the accuracy of the parameters without a given upper and lower limit value, but its typical value reasonably reflects the device performance.

### 5.2 Static Electrical Characteristics

Table 5-2: Electrical Characteristics

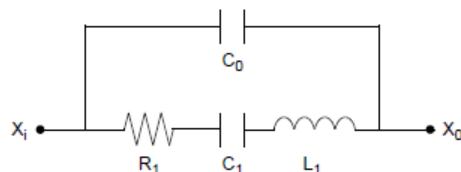
Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
V <sub>DD</sub>	System Voltage		3.0	3.3	3.6	V
C <sub>VDD</sub>	Loading Capacitor		1	-	10	μF
I <sub>OPR</sub>	Operation Current	Note 1		60		mA
I <sub>STB</sub>	Standby Mode	Note 1		30		mA
I <sub>SUSP</sub>	Suspend Mode	Note 1		10		mA
I <sub>SLP</sub>	Sleep Mode	Note 1		7		mA
T <sub>RMP</sub>	Power Ramp Up Time	VDD Ramp Up to 3.3 V			35	ms
<b>OSC / PLL</b>						
F <sub>Osc</sub>	Oscillator Clock	V <sub>DD</sub> = 3.3 V, Note 2		10		MHz
F <sub>VCO</sub>	VCO Output Clock Frequency		100		500	MHz
T <sub>LOCK</sub>	Lock Time	Note 3			500	us
CLK <sub>MPLL</sub>	MPLL Output Clock (MCLK)	VDD = 3.3 V			133	MHz
CLK <sub>CPLL</sub>	CPLL Output Clock (CCLK)	VDD = 3.3 V			100	MHz
CLK <sub>PPLL</sub>	PPLL Output Clock (PCLK)	VDD = 3.3 V			80	MHz
<b>Serial Host Interface</b>						
CLK <sub>SPI</sub>	SPI Input Clock				50	MHz

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
<b>Input/Output</b> (CMOS 3-state Output pad with Schmitt Trigger Input, Pull-Up/Down)						
$V_{IH}$	Input High Voltage		2		3.6	V
$V_{IL}$	Input Low Voltage		-0.3		0.8	V
$V_{OH}$	Output High Voltage		2.4			V
$V_{OL}$	Output Low Voltage				0.4	V
$R_{PU}$	Pull up Resistance		34	41	64	KΩ
$R_{PD}$	Pull down Resistance		33	44	79	KΩ
$V_{TP}$	Schmitt Trigger Low to High Threshold		1.5		2.1	V
$V_{TN}$	Schmitt Trigger High to Low Threshold		0.8		1.3	V
$V_{HVS}$	Hysteresis Voltage		200			mV
$I_{LEAK}$	Input Leakage Current		-10		+10	μA
$V_{SLEW}$	Rise/Fall Slew Rate			1.5		V/ns

**Note 1:** Condition:  $V_{DD} = 3.3V$ ,  $T_A = 25^\circ C$

**Note 2:** Measured on tester with 8 bit MPU interface and without extra load.

**Note 3:** Parasitic effect when the Crystal Oscillator is used.



Typical:  $R_1 = 50\Omega$  (25-100Ω),  $L_1 = 3.4mH$ ,  $C_1 = 13fF$ ,  $C_0 = 2.8pF$

**Figure 5-1: Equivalent Circuit**

**Note 4:** Time needed from power-up till the internal PLL have stable clock outputs.

### 5.3 Absolute Maximum Ratings(electrical sensitivity)

Based on specific criterias, the tests of ESD (HBM, MM) and LU tests are performed to stress the device for determining its performance in terms of electrical sensitivity.

#### Electrostatic discharge (ESD)

**Table 5-3 HBM:** Electrostatic discharges (MIL-STD ZAP 3 times, 1sec) are applied to the pins of each sample according to each pin combination. This test conforms to MIL-STD-883G Method 3015.7

Test Model: HBM	ESD Sensitivity Passed: <u><math>\pm 7000V</math></u>	MIL-STD Classification Class : <u>3A</u>
Test condition	Passed Volts	Class 0 : < 250V Class 1A : $\geq 250V, < 499V$ Class 1B : $\geq 500V, < 999V$ Class 1C : $\geq 1000V, < 1999V$ Class 2 : $\geq 2000V, < 3999V$ Class 3A : $\geq 4000V, < 7999V$ Class 3B : $\geq 8000V$
All other pin to VSS(+)	+8000V	
All other pin to VSS(-)	-8000V	
All other pin to VDD(+)	+8000V	
All other pin to VDD(-)	-8000V	
All other pin to VDD_C(+)	+7000V	
All other pin to VDD_C(-)	-8000V	
I/O to I/O(+)	+8000V	
I/O to I/O(-)	-8000V	

**Table 5-4 MM:** Electrostatic discharges (JEDEC ZAP 3 times, 1sec) are applied to the pins of each sample according to each pin combination. This test conforms to JEDEC EIA/JESD22-A115

Test Model: MM	ESD Sensitivity Passed: <u><math>\pm 400V</math></u>	JEDEC Classification Class: <u>C</u>
Test condition	Passed Volts	Class A : < 200V Class B : $\geq 200V, < 400V$ Class C : $\geq 400V$
All other pin to VSS(+)	+400V	
All other pin to VSS(-)	-400V	
All other pin to VDD(+)	+400V	
All other pin to VDD(-)	-400V	
All other pin to VDD_C(+)	+400V	
All other pin to VDD_C(-)	-400V	
I/O to I/O(+)	+400V	
I/O to I/O(-)	-400V	

**Table 5-5 Latch Up:** A supply overvoltage (3.5V~5.5V(+), Step: 0.5V(+)) is applied to each power supply pin; and a trigger current (50mA~150mA(±),Step:50mA(±) Limit:2V/3.6V) is applied to each input, output and configurable I/O pin. These tests conform to JEDEC STANDARD NO.78C SEPTMBER 2010

Trigger Model	Test Pin	Tested Result	I Trigger: Class I
I Trigger (+)	I/P3.6V	PASS+150mA	Class I Latch-up testing performed at room temperature
	O/P3.6V		
	O/P1.8V		
	I/O3.6V		
I Trigger (-)	I/P3.6V	PASS-150mA	
	O/P3.6V		
	O/P1.8V		
	I/O3.6V		
Over Volt Test	VDD	PASS+5.5V	

#### Laboratory Ambience Condition:

Temperature:  $23 \pm 5^\circ\text{C}$ ; Relative humidity:  $55\% \pm 10\%$ (RH)

## 6. Clock and Reset

### 6.1 Clock

LT768x has embedded PLL circuits to generate three clock source for internal circuit operations:

- CPLL : Provide **CCLK** for Host interface, BTE Engine, Graphics Engine, and Text DMA data transfer etc...
- MPLL : Provide **MCLK** for internal Display RAM
- PPLL : Provide **PCLK** for TFT-LCD's Scan Clock.

The three PLL are operation independent. The PLL output frequency is calculated from the following formula:

$$F_{OUT} = X_I * (N / R) \div OD$$

In above formula,  $X_I$  is the external Oscillator / Clock input. The input frequency " $X_I/R$ " is no less than 1MHz, and the default value is 1MHz. "R" is Input Divider Ratio, its value is between 2 ~ 31. "OD" is Output Divider Ratio that must be 1, 2 or 4. "N" is the Feedback Divider Ratio of Loop that indicated by 9bits, and its value is between 2 and 511.

**Table 6-1: PLL Register Setting (1)**

R[4:0]	Input Divider Ratio (R)	N[8:0]	Feedback Divider Ratio (N)
00010	2	000000010	2
00011	3	000000011	3
00101	4	000000101	4
:	:	:	:
:	:	:	:
:	:	:	:
11101	29	111111101	509
11110	30	111111110	510
11111	31	111111111	511

**Table 6-2: PLL Register Setting (2)**

OD[1:0]	Input Divider Ratio (OD)
00	1
01	2
10	3
11	4

For example, if XI is 10MHz, R[4:0] is 01010 (i.e. 10), N[8:0] is 100000000 (i.e. 256), OD[1:0] is 11 (i.e. 4), then:

$$F_{OUT} = 10\text{MHz} * (256 / 10) \div 4 = 64\text{MHz}$$

The design rules of the three clocks are::

1.  $CCLK * 2 \geq MCLK \geq CCLK$
2.  $CCLK \geq PCLK * 1.5$

In general, TFT manufacturers will advise the Pixel Clock(PCLK) for the best display effect, based on their TFT characteristics. Therefore, users can setup the register according to the requirements of the PCLK, and then apply the above rules to set up CCLK and MCLK. The PLL (Fout) should be set according to the LCD panel resolution.

For example, if the LCD panel resolution is 640\*480, and the recommended PCLK = 20MHz, then users may set MCLK = 40MHz, CCLK = 40MHz. The PLL output and the register settings will be as following:

<b>PCLK</b> = $XI * (N / R) \div OD$	REG[05h] : OD = 11b, R = 01010b
= $10\text{MHz} * (80 / 10) \div 4$	REG[06h] : N = 01010000b
= 20MHz	
<b>MCLK</b> = $XI * (N / R) \div OD$	REG[07h] : OD = 11b, R = 01010b
= $10\text{MHz} * (160 / 10) \div 4$	REG[08h] : N = 10100000b
= 40MHz	
<b>CCLK</b> = $XI * (N / R) \div OD$	REG[09h] : OD = 11b, R = 01010b
= $10\text{MHz} * (160 / 10) \div 4$	REG[0Ah] : N = 10100000b
= 40MHz	

Another example, if the LCD panel resolution is 800\*480, and the recommend PCLK = 25MHz, then users may set MCLK = 50MHz, CCLK = 50MHz. The PLL output and the register settings will be as following:

<b>PCLK</b> = $XI * (N / R) \div OD$	REG[05h] : OD = 11b, R = 01010b
= $10\text{MHz} * (100 / 10) \div 4$	REG[06h] : N = 01100100b
= 25MHz	
<b>MCLK</b> = $XI * (N / R) \div OD$	REG[07h] : OD = 11b, R = 01010b
= $10\text{MHz} * (200 / 10) \div 4$	REG[08h] : N = 11001000b
= 50MHz	
<b>CCLK</b> = $XI * (N / R) \div OD$	REG[09h] : OD = 11b, R = 01010b
= $10\text{MHz} * (200 / 10) \div 4$	REG[0Ah] : N = 11001000b
= 50MHz	

**Table 6-3: PLL Register Setting Example**

Registers	640*480	800*480
REG[05h], PPLLC1	11010100b	11010100b
REG[06h], PPLLC2	01010000b	01100100b
REG[07h], MPLLC1	11010100b	11010100b
REG[08h], MPLLC2	10100000b	11001000b
REG[09h], CPLLC1	11010100b	11010100b
REG[0Ah], CPLLC1	10100000b	11001000b

## 6.2 Reset

### 6.2.1 Power-on Reset

LT768x has an embedded Power-On-Reset (POR) circuit. POR can issue an active low signal to synchronize the whole system through RST# pin. When system power (3.3V) on, the internal reset will be activated for at least 256 OSC clocks until the internal power is stable.

### 6.2.2 External Reset

External reset signal RST# allows LT768x to synchronize with external systems. The external reset signal must be stabilized for at least 256 OSC clocks to be approved, as shown in Figure 6-1. The MCU should check the BIT1(working mode status indication bit) of the state register STSR before setting up LT768x to ensure that LT768x is currently in "Normal Operating State".

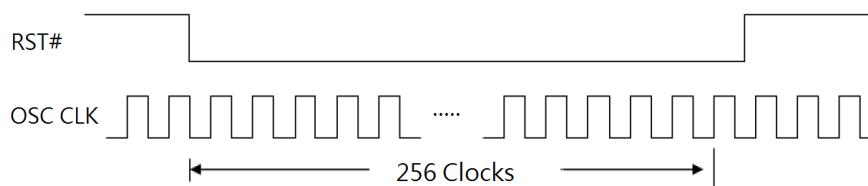


Figure 6-1: External Reset Signal

### 6.2.3 Software Reset

If the Host writes 1 to register REG[00h] bit0, LT768x will be reset by software. The software reset will only reset the internal state machine of LT768x, and the other registers values will not be affected or cleared. After the software reset is complete, the REG[00h] bit0 will be cleared to 0 automatically.

## 7. Host Interface

LT768x provides 8bits and 16bits parallel modes, serial SPI mode, and I2C mode for the communication with Host. These interface modes are setup by PSM[2:0] pins:

**Table 7-1: Host Interface Mode**

PSM[2:0]	Host Interface
0 0 X	8bits or 16bits 8080 Parallel Interface Mode
0 1 X	8bits or 16bits 6800 Parallel Interface Mode
1 0 0	3-Wire SPI Mode
1 0 1	4-Wire SPI Mode
1 1 X	I2C Mode

Because different MCU interfaces cannot be used at the same time, LT768x provides a shared pin mode. When using Serial Mode, the other parallel pins can be set as GPIO pins. Please refer to the following table:

**Table 7-2: The Pin Definition of Host Interface**

Pin Name	8080 I/F		6800 I/F		SPI 3-Wires	SPI 4-Wires	I2C
	8-bits	16-bits	8-bits	16-bits			
DB[15:8]	--		--		GPIOA[0:7]	GPIOA[0:7]	GPIOA[0:7]
DB[7]					SCLK	SCLK	SCLK
DB[6]					GND	SDI	I2C_SDA
DB[5]					SD	SDO	I2CA[5]
DB[4]					SCS#	SCS#	I2CA[4]
DB[3:0]					GND	GND	I2CA[3:0]
CS#	CS#		CS#		GPIOB[0]	GPIOB[0]	GPIOB[0]
RD#	RD#		EN		GPIOB[1]	GPIOB[1]	GPIOB[1]
WR#	WR#		RW#		GPIOB[2]	GPIOB[2]	GPIOB[2]
A0	A0		A0		GPIOB[3]	GPIOB[3]	GPIOB[3]
INT#	INT#		INT#		INT#	INT#	INT#
WAIT#	WAIT#		WAIT#		--	--	--

When using the Parallel Host mode, 8bits or 16bits mode is determined by the bit0 of Register REG[01h]. When the bit0=0, then 8bits mode is selected. If this bit0=1 then 16bits mode is selected.

LT768x series support different Host interfaces. For example, LT7680 is a 68pin QFN chip , which only supports serial SPI mode. The following table is the list of host interfaces supported by LT768x series:

**Table 7-3: Host Interface Supporting List of LT768x Series**

No.	Host Interface Mode	LT7681 LT7683 LT7686	LT7680A-R LT7680B-R
1	8bits 8080 Parallel Interface Mode	v	
2	16bits 8080 Parallel Interface Mode	v	
3	8bits 6800 Parallel Interface Mode	v	
4	16bits 6800 Parallel Interface Mode	v	
5	3-Wire SPI Mode	v	v
6	4-Wire SPI Mode	v	v
7	I2C Mode	v	

The Host interface is selected by pins PSM[2:0] except for LT7680. Please refer to previous Table 7-1. LT7680 only supports 3-Wires and 4-Wires SPI mode. For LT7680, the PSM[1] pin is already connected to GND inside the chip, whereas PSM[2] must be pull-up to VDD. When PSM[0] = 0, then the 3-wires Serial SPI mode is selected. When PSM[0] = 1, then the 4-wires serial mode is selected.

## 7.1 Parallel Host Interface

The followings are the application circuit and timing of 8080/6800 parallel interface:

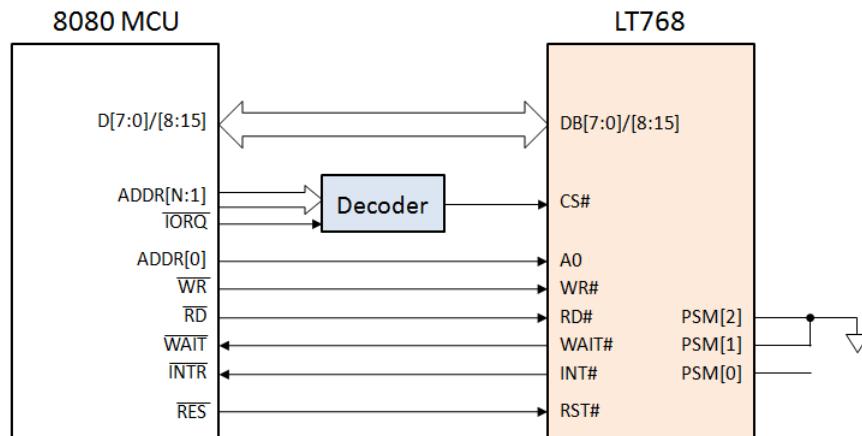


Figure 7-1: 8080 Parallel Mode Interface

The WAIT# signal is used to indicate whether LT768x is ready to transfer data or not. If WAIT# signal is not connected, then the Host access cycle time has to be longer than five CCLK clocks to avoid access fail. If the Host's Reset signal is active low, then it can be connected to RST# of LT768x. The RST# can also be controlled by the I/O pin of host, or be connected to an RC circuit that generate a low pulse. However, to confirm RST#'s active cycle is effective, the signal has to last at least 256 system clock cycle. While using LT768x, Host should first confirm the state register bit1, to find out whether the LT768x is in the standard operating state.

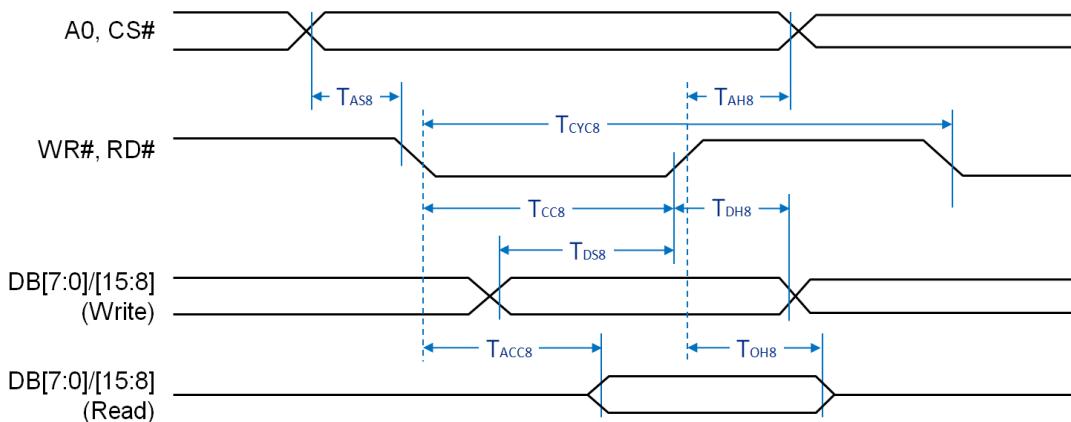


Figure 7-2: 8080 Parallel Mode Interface Timing

Table 7-4: 8080 Parallel Mode Interface Timing Parameter

Symbol	Parameter	Rating		Unit	Note
		Min.	Max.		
T <sub>CYC8</sub>	Cycle Time	50	--	ns	tc is one system clock period: tc = 1/SYS_CLK
T <sub>CC8</sub>	Strobe Pulse Width	20	--	ns	
T <sub>AS8</sub>	Address Setup Time	0	--	ns	
T <sub>AH8</sub>	Address Hold Time	10	--	ns	
T <sub>DS8</sub>	Data Setup Time	20	--	ns	
T <sub>DH8</sub>	Data Hold Time	10	--	ns	
T <sub>ACC8</sub>	Data Output Access Time	0	20	ns	
T <sub>OH8</sub>	Data Output Hold Time	0	20	ns	

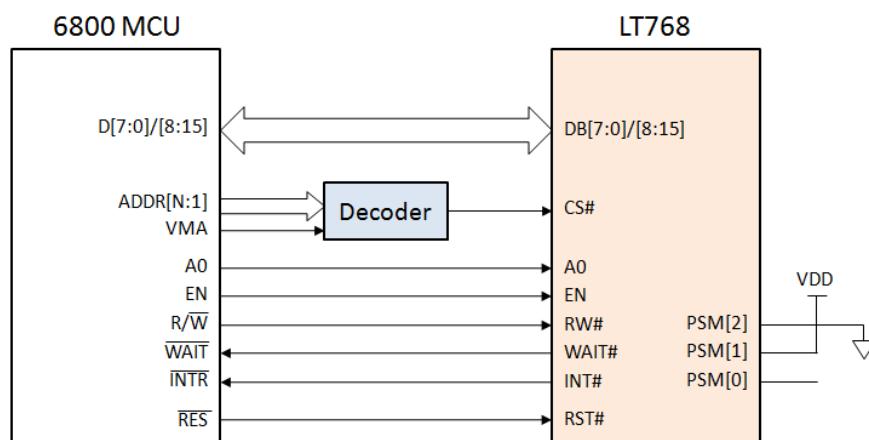


Figure 7-3: 6800 Parallel Mode Interface

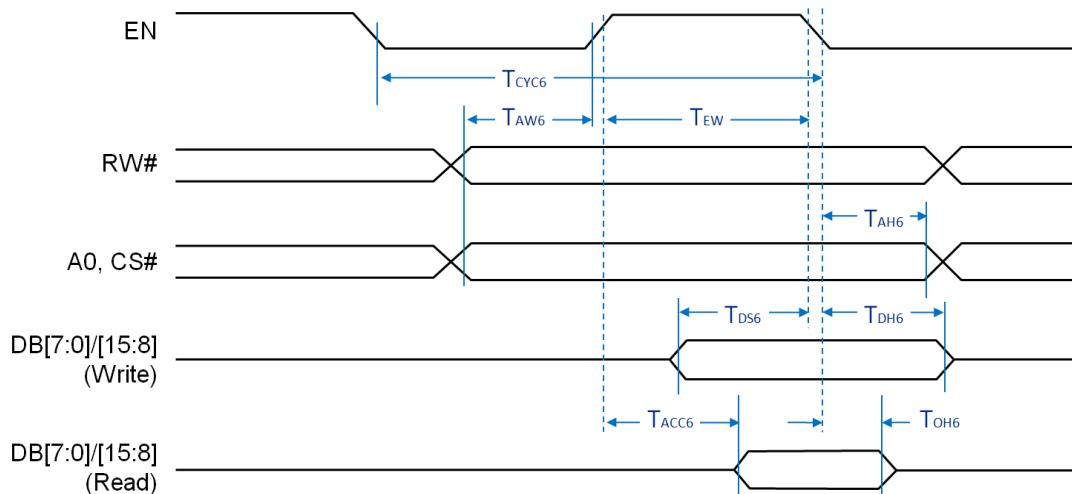


Figure 7-4: 6800 Parallel Mode Interface Timing

**Table 7-5: 6800 Parallel Mode Interface Timing Parameter**

Symbol	Parameter	Rating		Unit	Note
		Min.	Max.		
T <sub>CYC6</sub>	Cycle Time	50	--	ns	tc is one system clock period: tc = 1/SYS_CLK
T <sub>EW</sub>	Strobe Pulse Width	20	--	ns	
T <sub>AW6</sub>	Address Setup Time	0	--	ns	
T <sub>AH6</sub>	Address Hold Time	10	--	ns	
T <sub>DS6</sub>	Data Setup Time	20	--	ns	
T <sub>DH6</sub>	Data Hold Time	10	--	ns	
T <sub>ACC6</sub>	Data Output Access Time	0	20	ns	
T <sub>OH6</sub>	Data Output Hold Time	0	20	ns	

Host accesses LT768x's Registers and Display Memory through the host interface. LT768x has one Status Register and 256 Instruction Registers (i.e. .REG[00h] ~ REG[FF]). The access procedure is as following:

#### Register Write:

1. Address Write: Write the Register's Address. For example, 00h i.e. REG[00h], 01h i.e. REG[01h], 02h i.e. REG[02h] .....
2. Data Write: Write Data to the Register

#### Register Read:

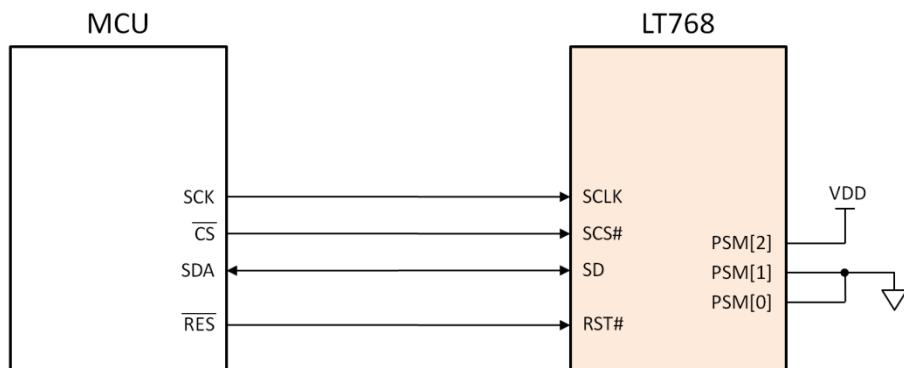
1. Address Write: Write the Register's Address
2. Data Write: Read Data from the Register

Displays Memory (Display RAM) is where the TFT screen image data is stored. Host writes data into Display RAM through interface. The procedure of accessing Display RAM is as following:

#### Display RAM Write:

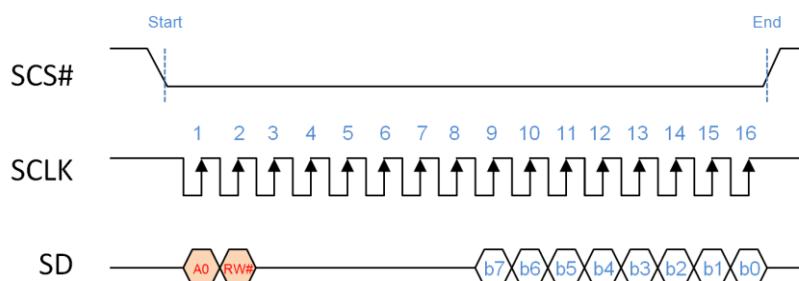
1. Set the Active Window Registers before writing any image data.
2. Perform a register write to Graphic R/W Position Register (REG[5Fh] ~ REG[62h]).
3. Repeat step 2 until all the Active Window & Graphic R/W Position Coordinates are set.
4. Perform an address write to point to Memory Data Port Register (REG[04h])
5. Perform data writes to fill the window. Each write to the Memory Data Port will auto-increment the internal memory address.

## 7.2 Serial Host Interface



**Figure 7-5: 3-Wire SPI Interface**

The above circuit is the LT768x's 3-Wires SPI interface with Host. SD signal is a bi-direction data pin for data access. The access timing and procedure are as below:



**Figure 7-6: 3-Wire SPI Interface Timing**

### Status Register Read:

1. Host drive SCS#(Low) and SCLK(SPI Clock).
2. Host drive A0(Low), then drive RW#(High).
3. LT768x will drive the Data of Status Register (b7 ~ b0) at 9<sup>th</sup> ~ 16<sup>th</sup> Clock. Then Host will get the content of Status Register.

### Write Register's Address:

1. Host drive SCS#(Low) and SCLK.
2. Host drive A0(Low), then drive RW#(Low).
3. Host drive the Register's Address (b0 ~ b7) at 9<sup>th</sup> ~ 16<sup>th</sup> Clock to LT768x.

### Write Data to Register or Memory:

1. Host drive SCS#(Low) and SCLK.
2. Host drive A0(High), then drive RW#(Low).
3. Host drive the Data at 9<sup>th</sup> ~ 16<sup>th</sup> Clock to LT768x. i.e. Data will be stored in Register or Memory.

### Read Register's Data:

1. Host drives SCS#(Low) and SCLK.
2. Host drives A0(High), then drives RW#(High).
3. LT768x will drive the Data of Register at 9<sup>th</sup> ~ 16<sup>th</sup> Clock. Then Host will get the content of Register.

The 4-Wires SPI is almost the same as 3-Wires. The difference is its input and output data lines are separated. The interface circuit diagram and Timing are as follows:

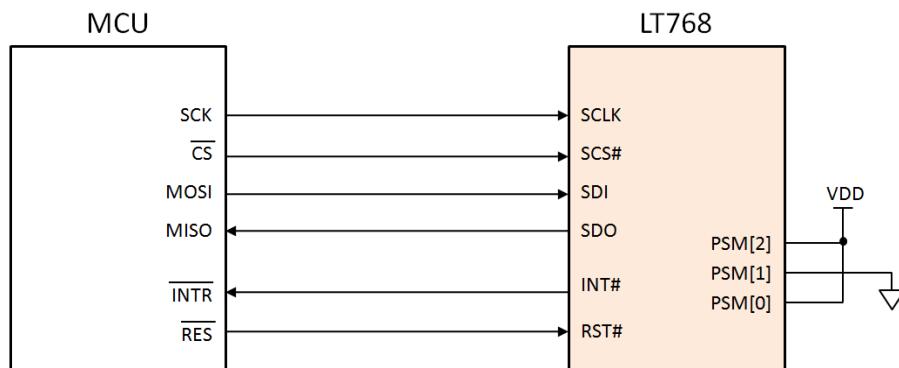


Figure 7-7: 4-Wire SPI Interface

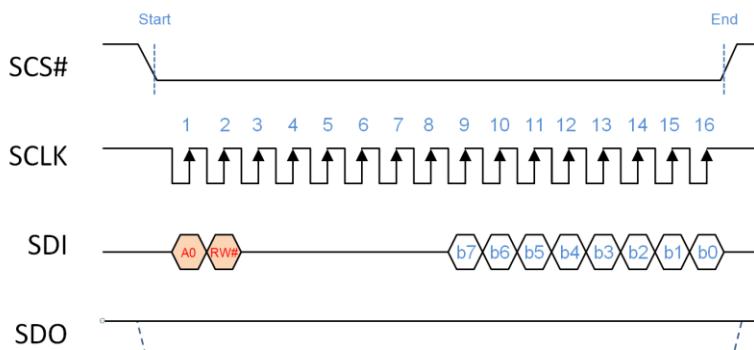
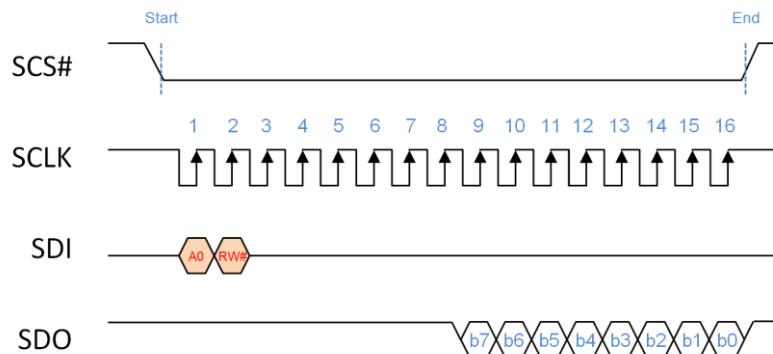


Figure 7-8: 4-Wire SPI Interface Write Timing

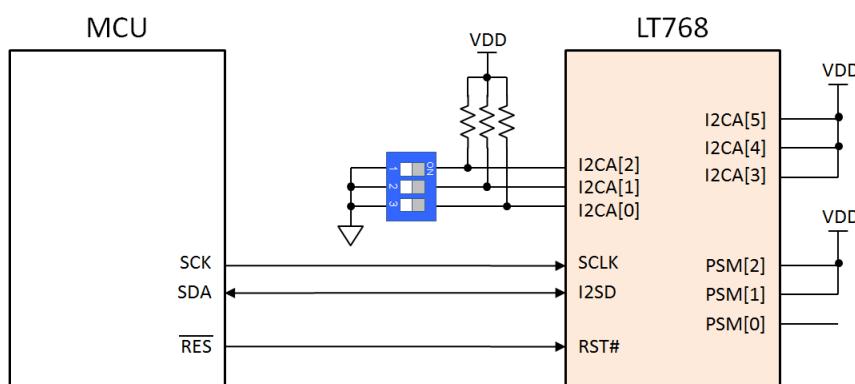
The above Timing diagram is the Write Cycle of 4-Wires SPI. When Host drives A0(Low) and RW#(Low), it means Host writes Register's Address. When Host drives A0(High) and RW#(Low), it means Host writes data to Register or Display RAM.

The following Timing diagram is the Read Cycle of 4-Wires SPI. When Host drives A0(Low) and RW#(High), it means Host wants to read the data of Status Register. LT768x will drive the Data of Status Register (b7 ~ b0) at 9<sup>th</sup> ~ 16<sup>th</sup> Clock. Then Host will get the data of Status Register. When Host drives A0(High), then RW#(High), it means Host wants to read the data of Command Register. LT768x will drive the Data of Command Register (b7 ~ b0) at 9<sup>th</sup> ~ 16<sup>th</sup> Clock for Host. Host will then get the content of Command Register.



**Figure 7-9: 4-Wire SPI Interface Read Timing**

The serial I2C interface is similar to 3-Wires SPI interface, yet I2C interface only needs 2 wires for data transfer. The following is the application circuit of I2C interface. Signals I2CA[5:0] are used to setup LT768x's Device ID, and to avoid confusing with other I2C devices. In this example circuit, I2CA[5:3] connect to VDD, and if all DIP Switches are "ON" state, then the I2C Device ID is 111000b. i.e. 38h. Therefore if Host drives I2C timing with "111000b" Device ID, then LT768x will communicate in the I2C cycle time with the Host.



**Figure 7-10: I2C Interface**

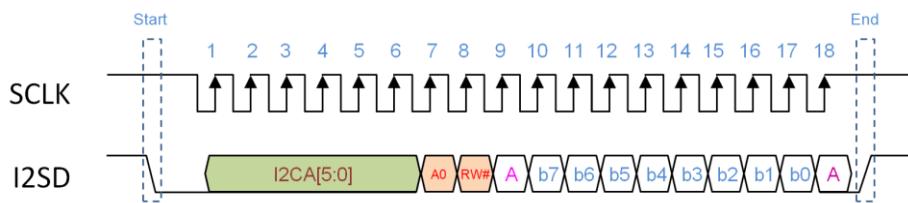


Figure 7-11: I2C Interface Timing

Figure 7-11 is the timing diagram of I2C interface. At first, Host has to find out the Device ID of the I2C device. Then release the Device ID's data in the five clocks of beginning. In the example of Figure 7-10, the Device ID data is 111000. The definition of A0 and RW# are the same as SPI interface. When Host releases A0(High) and RW#(Low), it means Host will write data(b7 ~ b0, at 10th ~ 17th Clock) to Command Register or Display RAM. If Host releases A0(High) and RW#(High), it means Host wants to read data from Command Register. LT768x will release the content(b7 ~ b0) of Command Register on 10th ~ 17th Clock. If Host releases A0(Low) and RW#(High), it means Host wants to read Status Data. LT768x will release the Status Data(b7 ~ b0) on 10th ~ 17th Clock. Host will then get the data of LT768x's Status Register.

### 7.3 Display Input Data Format

LT768x supports Monochrome, 256 color, 65K color and 16.7M color for TFT panel. The data for RGB colors are arranged in Display RAM as follows:

- 1bpp : Monochrome (1bit/pixel)
- 8bpp : Color RGB 3:3:2 (1 byte/pixel)
- 16bpp : Color RGB 5:6:5 (2bytes/pixel)
- 24bpp : Color RGB 8:8:8 (3bytes/pixel, or 4bytes/pixel)

The following examples are based on 8bits MCU, 16bits MCU, and display color (RGB) in different data format.

#### 7.3.1 Input Data without Opacity (RGB)

Table 7-6: 8bits MCU, 1bpp Monochrom Mode

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	P <sub>7</sub>	P <sub>6</sub>	P <sub>5</sub>	P <sub>4</sub>	P <sub>3</sub>	P <sub>2</sub>	P <sub>1</sub>	P <sub>0</sub>
2	P <sub>15</sub>	P <sub>14</sub>	P <sub>13</sub>	P <sub>12</sub>	P <sub>11</sub>	P <sub>10</sub>	P <sub>9</sub>	P <sub>8</sub>
3	P <sub>23</sub>	P <sub>22</sub>	P <sub>21</sub>	P <sub>20</sub>	P <sub>19</sub>	P <sub>18</sub>	P <sub>17</sub>	P <sub>16</sub>
4	P <sub>31</sub>	P <sub>30</sub>	P <sub>29</sub>	P <sub>28</sub>	P <sub>27</sub>	P <sub>26</sub>	P <sub>25</sub>	P <sub>24</sub>
5	P <sub>39</sub>	P <sub>38</sub>	P <sub>37</sub>	P <sub>36</sub>	P <sub>35</sub>	P <sub>34</sub>	P <sub>33</sub>	P <sub>32</sub>
6	P <sub>47</sub>	P <sub>46</sub>	P <sub>45</sub>	P <sub>44</sub>	P <sub>43</sub>	P <sub>42</sub>	P <sub>41</sub>	P <sub>40</sub>

Table 7-7: 8bits MCU, 8bpp Mode (RGB 3:3:2)

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>
2	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>
3	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>
4	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>
5	R <sub>4</sub> <sup>7</sup>	R <sub>4</sub> <sup>6</sup>	R <sub>4</sub> <sup>5</sup>	G <sub>4</sub> <sup>7</sup>	G <sub>4</sub> <sup>6</sup>	G <sub>4</sub> <sup>5</sup>	B <sub>4</sub> <sup>7</sup>	B <sub>4</sub> <sup>6</sup>
6	R <sub>5</sub> <sup>7</sup>	R <sub>5</sub> <sup>6</sup>	R <sub>5</sub> <sup>5</sup>	G <sub>5</sub> <sup>7</sup>	G <sub>5</sub> <sup>6</sup>	G <sub>5</sub> <sup>5</sup>	B <sub>5</sub> <sup>7</sup>	B <sub>5</sub> <sup>6</sup>

Table 7-8: 8bits MCU, 16bpp Mode (RGB 5:6:5)

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	G <sub>0</sub> <sup>4</sup>	G <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>2</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>	B <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>4</sup>	B <sub>0</sub> <sup>3</sup>
2	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	R <sub>0</sub> <sup>4</sup>	R <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>
3	G <sub>1</sub> <sup>4</sup>	G <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>2</sup>	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	B <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>4</sup>	B <sub>1</sub> <sup>3</sup>
4	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	R <sub>1</sub> <sup>4</sup>	R <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>
5	G <sub>2</sub> <sup>4</sup>	G <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>2</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>	B <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>4</sup>	B <sub>2</sub> <sup>3</sup>
6	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	R <sub>2</sub> <sup>4</sup>	R <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>

Table 7-9: 8bits MCU, 24bpp Mode (RGB 8:8:8)

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>	B <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>4</sup>	B <sub>0</sub> <sup>3</sup>	B <sub>0</sub> <sup>2</sup>	B <sub>0</sub> <sup>1</sup>	B <sub>0</sub> <sup>0</sup>
2	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>4</sup>	G <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>2</sup>	G <sub>0</sub> <sup>1</sup>	G <sub>0</sub> <sup>0</sup>
3	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	R <sub>0</sub> <sup>4</sup>	R <sub>0</sub> <sup>3</sup>	R <sub>0</sub> <sup>2</sup>	R <sub>0</sub> <sup>1</sup>	R <sub>0</sub> <sup>0</sup>
4	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	B <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>4</sup>	B <sub>1</sub> <sup>3</sup>	B <sub>1</sub> <sup>2</sup>	B <sub>1</sub> <sup>1</sup>	B <sub>1</sub> <sup>0</sup>
5	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>4</sup>	G <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>2</sup>	G <sub>1</sub> <sup>1</sup>	G <sub>1</sub> <sup>0</sup>
6	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	R <sub>1</sub> <sup>4</sup>	R <sub>1</sub> <sup>3</sup>	R <sub>1</sub> <sup>2</sup>	R <sub>1</sub> <sup>1</sup>	R <sub>1</sub> <sup>0</sup>

Table 7-10: 16bits MCU, 1bpp Mono Mode -1

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P <sub>7</sub>	P <sub>6</sub>	P <sub>5</sub>	P <sub>4</sub>	P <sub>3</sub>	P <sub>2</sub>	P <sub>1</sub>	P <sub>0</sub>
2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P <sub>15</sub>	P <sub>14</sub>	P <sub>13</sub>	P <sub>12</sub>	P <sub>11</sub>	P <sub>10</sub>	P <sub>9</sub>	P <sub>8</sub>
3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P <sub>23</sub>	P <sub>22</sub>	P <sub>21</sub>	P <sub>20</sub>	P <sub>19</sub>	P <sub>18</sub>	P <sub>17</sub>	P <sub>16</sub>
4	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P <sub>31</sub>	P <sub>30</sub>	P <sub>29</sub>	P <sub>28</sub>	P <sub>27</sub>	P <sub>26</sub>	P <sub>25</sub>	P <sub>24</sub>
5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P <sub>39</sub>	P <sub>38</sub>	P <sub>37</sub>	P <sub>36</sub>	P <sub>35</sub>	P <sub>34</sub>	P <sub>33</sub>	P <sub>32</sub>
6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P <sub>47</sub>	P <sub>46</sub>	P <sub>45</sub>	P <sub>44</sub>	P <sub>43</sub>	P <sub>42</sub>	P <sub>41</sub>	P <sub>40</sub>

Table 7-11: 16bits MCU, 1bpp Mono Mode -2

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	P <sub>15</sub>	P <sub>14</sub>	P <sub>13</sub>	P <sub>12</sub>	P <sub>11</sub>	P <sub>10</sub>	P <sub>9</sub>	P <sub>8</sub>	P <sub>7</sub>	P <sub>6</sub>	P <sub>5</sub>	P <sub>4</sub>	P <sub>3</sub>	P <sub>2</sub>	P <sub>1</sub>	P <sub>0</sub>
2	P <sub>31</sub>	P <sub>30</sub>	P <sub>29</sub>	P <sub>28</sub>	P <sub>27</sub>	P <sub>26</sub>	P <sub>25</sub>	P <sub>24</sub>	P <sub>23</sub>	P <sub>22</sub>	P <sub>21</sub>	P <sub>20</sub>	P <sub>19</sub>	P <sub>18</sub>	P <sub>17</sub>	P <sub>16</sub>
3	P <sub>47</sub>	P <sub>46</sub>	P <sub>45</sub>	P <sub>44</sub>	P <sub>43</sub>	P <sub>42</sub>	P <sub>41</sub>	P <sub>40</sub>	P <sub>39</sub>	P <sub>38</sub>	P <sub>37</sub>	P <sub>36</sub>	P <sub>35</sub>	P <sub>34</sub>	P <sub>33</sub>	P <sub>32</sub>
4	P <sub>63</sub>	P <sub>62</sub>	P <sub>61</sub>	P <sub>60</sub>	P <sub>59</sub>	P <sub>58</sub>	P <sub>57</sub>	P <sub>56</sub>	P <sub>55</sub>	P <sub>54</sub>	P <sub>53</sub>	P <sub>52</sub>	P <sub>51</sub>	P <sub>50</sub>	P <sub>49</sub>	P <sub>48</sub>
5	P <sub>79</sub>	P <sub>78</sub>	P <sub>77</sub>	P <sub>76</sub>	P <sub>75</sub>	P <sub>74</sub>	P <sub>73</sub>	P <sub>72</sub>	P <sub>71</sub>	P <sub>70</sub>	P <sub>69</sub>	P <sub>68</sub>	P <sub>67</sub>	P <sub>66</sub>	P <sub>65</sub>	P <sub>64</sub>
6	P <sub>95</sub>	P <sub>94</sub>	P <sub>93</sub>	P <sub>92</sub>	P <sub>91</sub>	P <sub>90</sub>	P <sub>89</sub>	P <sub>88</sub>	P <sub>87</sub>	P <sub>86</sub>	P <sub>85</sub>	P <sub>84</sub>	P <sub>83</sub>	P <sub>82</sub>	P <sub>81</sub>	P <sub>80</sub>

Table 7-12: 16bits MCU, 8bpp Mode -1 (RGB 3:3:2)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>
2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>
3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>
4	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>
5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R <sub>4</sub> <sup>7</sup>	R <sub>4</sub> <sup>6</sup>	R <sub>4</sub> <sup>5</sup>	G <sub>4</sub> <sup>7</sup>	G <sub>4</sub> <sup>6</sup>	G <sub>4</sub> <sup>5</sup>	B <sub>4</sub> <sup>7</sup>	B <sub>4</sub> <sup>6</sup>
6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R <sub>5</sub> <sup>7</sup>	R <sub>5</sub> <sup>6</sup>	R <sub>5</sub> <sup>5</sup>	G <sub>5</sub> <sup>7</sup>	G <sub>5</sub> <sup>6</sup>	G <sub>5</sub> <sup>5</sup>	B <sub>5</sub> <sup>7</sup>	B <sub>5</sub> <sup>6</sup>

Table 7-13: 16bits MCU, 8bpp Mode -2 (RGB 3:3:2)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>
2	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>
3	R <sub>5</sub> <sup>7</sup>	R <sub>5</sub> <sup>6</sup>	R <sub>5</sub> <sup>5</sup>	G <sub>5</sub> <sup>7</sup>	G <sub>5</sub> <sup>6</sup>	G <sub>5</sub> <sup>5</sup>	B <sub>5</sub> <sup>7</sup>	B <sub>5</sub> <sup>6</sup>	R <sub>4</sub> <sup>7</sup>	R <sub>4</sub> <sup>6</sup>	R <sub>4</sub> <sup>5</sup>	G <sub>4</sub> <sup>7</sup>	G <sub>4</sub> <sup>6</sup>	G <sub>4</sub> <sup>5</sup>	B <sub>4</sub> <sup>7</sup>	B <sub>4</sub> <sup>6</sup>
4	R <sub>7</sub> <sup>7</sup>	R <sub>7</sub> <sup>6</sup>	R <sub>7</sub> <sup>5</sup>	G <sub>7</sub> <sup>7</sup>	G <sub>7</sub> <sup>6</sup>	G <sub>7</sub> <sup>5</sup>	B <sub>7</sub> <sup>7</sup>	B <sub>7</sub> <sup>6</sup>	R <sub>6</sub> <sup>7</sup>	R <sub>6</sub> <sup>6</sup>	R <sub>6</sub> <sup>5</sup>	G <sub>6</sub> <sup>7</sup>	G <sub>6</sub> <sup>6</sup>	G <sub>6</sub> <sup>5</sup>	B <sub>6</sub> <sup>7</sup>	B <sub>6</sub> <sup>6</sup>
5	R <sub>9</sub> <sup>7</sup>	R <sub>9</sub> <sup>6</sup>	R <sub>9</sub> <sup>5</sup>	G <sub>9</sub> <sup>7</sup>	G <sub>9</sub> <sup>6</sup>	G <sub>9</sub> <sup>5</sup>	B <sub>9</sub> <sup>7</sup>	B <sub>9</sub> <sup>6</sup>	R <sub>8</sub> <sup>7</sup>	R <sub>8</sub> <sup>6</sup>	R <sub>8</sub> <sup>5</sup>	G <sub>8</sub> <sup>7</sup>	G <sub>8</sub> <sup>6</sup>	G <sub>8</sub> <sup>5</sup>	B <sub>8</sub> <sup>7</sup>	B <sub>8</sub> <sup>6</sup>
6	R <sub>11</sub> <sup>7</sup>	R <sub>11</sub> <sup>6</sup>	R <sub>11</sub> <sup>5</sup>	G <sub>11</sub> <sup>7</sup>	G <sub>11</sub> <sup>6</sup>	G <sub>11</sub> <sup>5</sup>	B <sub>11</sub> <sup>7</sup>	B <sub>11</sub> <sup>6</sup>	R <sub>10</sub> <sup>7</sup>	R <sub>10</sub> <sup>6</sup>	R <sub>10</sub> <sup>5</sup>	G <sub>10</sub> <sup>7</sup>	G <sub>10</sub> <sup>6</sup>	G <sub>10</sub> <sup>5</sup>	B <sub>10</sub> <sup>7</sup>	B <sub>10</sub> <sup>6</sup>

**Note:** The Mode-1 and Mode 2 is determined by bit[7:6] of Register[02h], please refer to Chapter 19 Reregister Description.

Table 7-14: 16bits MCU, 16bpp Mode (RGB 5:6:5)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	R <sub>0</sub> <sup>4</sup>	R <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>4</sup>	G <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>2</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>	B <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>4</sup>	B <sub>0</sub> <sup>3</sup>
2	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	R <sub>1</sub> <sup>4</sup>	R <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>4</sup>	G <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>2</sup>	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	B <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>4</sup>	B <sub>1</sub> <sup>3</sup>
3	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	R <sub>2</sub> <sup>4</sup>	R <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>4</sup>	G <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>2</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>	B <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>4</sup>	B <sub>2</sub> <sup>3</sup>
4	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	R <sub>3</sub> <sup>4</sup>	R <sub>3</sub> <sup>3</sup>	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	G <sub>3</sub> <sup>4</sup>	G <sub>3</sub> <sup>3</sup>	G <sub>3</sub> <sup>2</sup>	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>	B <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>4</sup>	B <sub>3</sub> <sup>3</sup>
5	R <sub>4</sub> <sup>7</sup>	R <sub>4</sub> <sup>6</sup>	R <sub>4</sub> <sup>5</sup>	R <sub>4</sub> <sup>4</sup>	R <sub>4</sub> <sup>3</sup>	G <sub>4</sub> <sup>7</sup>	G <sub>4</sub> <sup>6</sup>	G <sub>4</sub> <sup>5</sup>	G <sub>4</sub> <sup>4</sup>	G <sub>4</sub> <sup>3</sup>	G <sub>4</sub> <sup>2</sup>	B <sub>4</sub> <sup>7</sup>	B <sub>4</sub> <sup>6</sup>	B <sub>4</sub> <sup>5</sup>	B <sub>4</sub> <sup>4</sup>	B <sub>4</sub> <sup>3</sup>
6	R <sub>5</sub> <sup>7</sup>	R <sub>5</sub> <sup>6</sup>	R <sub>5</sub> <sup>5</sup>	R <sub>5</sub> <sup>4</sup>	R <sub>5</sub> <sup>3</sup>	G <sub>5</sub> <sup>7</sup>	G <sub>5</sub> <sup>6</sup>	G <sub>5</sub> <sup>5</sup>	G <sub>5</sub> <sup>4</sup>	G <sub>5</sub> <sup>3</sup>	G <sub>5</sub> <sup>2</sup>	B <sub>5</sub> <sup>7</sup>	B <sub>5</sub> <sup>6</sup>	B <sub>5</sub> <sup>5</sup>	B <sub>5</sub> <sup>4</sup>	B <sub>5</sub> <sup>3</sup>

Table 7-15: 16bits MCU, 24bpp Mode -1 (RGB 8:8:8)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>4</sup>	G <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>2</sup>	G <sub>0</sub> <sup>1</sup>	G <sub>0</sub> <sup>0</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>	B <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>4</sup>	B <sub>0</sub> <sup>3</sup>	B <sub>0</sub> <sup>2</sup>	B <sub>0</sub> <sup>1</sup>	B <sub>0</sub> <sup>0</sup>
2	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	B <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>4</sup>	B <sub>1</sub> <sup>3</sup>	B <sub>1</sub> <sup>2</sup>	B <sub>1</sub> <sup>1</sup>	B <sub>1</sub> <sup>0</sup>	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	R <sub>0</sub> <sup>4</sup>	R <sub>0</sub> <sup>3</sup>	R <sub>0</sub> <sup>2</sup>	R <sub>0</sub> <sup>1</sup>	R <sub>0</sub> <sup>0</sup>
3	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	R <sub>1</sub> <sup>4</sup>	R <sub>1</sub> <sup>3</sup>	R <sub>1</sub> <sup>2</sup>	R <sub>1</sub> <sup>1</sup>	R <sub>1</sub> <sup>0</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>4</sup>	G <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>2</sup>	G <sub>1</sub> <sup>1</sup>	G <sub>1</sub> <sup>0</sup>
4	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>4</sup>	G <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>2</sup>	G <sub>2</sub> <sup>1</sup>	G <sub>2</sub> <sup>0</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>	B <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>4</sup>	B <sub>2</sub> <sup>3</sup>	B <sub>2</sub> <sup>2</sup>	B <sub>2</sub> <sup>1</sup>	B <sub>2</sub> <sup>0</sup>
5	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>	B <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>4</sup>	B <sub>3</sub> <sup>3</sup>	B <sub>3</sub> <sup>2</sup>	B <sub>3</sub> <sup>1</sup>	B <sub>3</sub> <sup>0</sup>	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	R <sub>2</sub> <sup>4</sup>	R <sub>2</sub> <sup>3</sup>	R <sub>2</sub> <sup>2</sup>	R <sub>2</sub> <sup>1</sup>	R <sub>2</sub> <sup>0</sup>
6	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	R <sub>3</sub> <sup>4</sup>	R <sub>3</sub> <sup>3</sup>	R <sub>3</sub> <sup>2</sup>	R <sub>3</sub> <sup>1</sup>	R <sub>3</sub> <sup>0</sup>	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	G <sub>3</sub> <sup>4</sup>	G <sub>3</sub> <sup>3</sup>	G <sub>3</sub> <sup>2</sup>	G <sub>3</sub> <sup>1</sup>	G <sub>3</sub> <sup>0</sup>

Table 7-16: 16bits MCU, 24bpp Mode -2 (RGB 8:8:8)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>4</sup>	G <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>2</sup>	G <sub>0</sub> <sup>1</sup>	G <sub>0</sub> <sup>0</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>	B <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>4</sup>	B <sub>0</sub> <sup>3</sup>	B <sub>0</sub> <sup>2</sup>	B <sub>0</sub> <sup>1</sup>	B <sub>0</sub> <sup>0</sup>
2	n/a	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	R <sub>0</sub> <sup>4</sup>	R <sub>0</sub> <sup>3</sup>	R <sub>0</sub> <sup>2</sup>	R <sub>0</sub> <sup>1</sup>	R <sub>0</sub> <sup>0</sup>							
3	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>4</sup>	G <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>2</sup>	G <sub>1</sub> <sup>1</sup>	G <sub>1</sub> <sup>0</sup>	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	B <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>4</sup>	B <sub>1</sub> <sup>3</sup>	B <sub>1</sub> <sup>2</sup>	B <sub>1</sub> <sup>1</sup>	B <sub>1</sub> <sup>0</sup>
4	n/a	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	R <sub>1</sub> <sup>4</sup>	R <sub>1</sub> <sup>3</sup>	R <sub>1</sub> <sup>2</sup>	R <sub>1</sub> <sup>1</sup>	R <sub>1</sub> <sup>0</sup>							
5	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>4</sup>	G <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>2</sup>	G <sub>2</sub> <sup>1</sup>	G <sub>2</sub> <sup>0</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>	B <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>4</sup>	B <sub>2</sub> <sup>3</sup>	B <sub>2</sub> <sup>2</sup>	B <sub>2</sub> <sup>1</sup>	B <sub>2</sub> <sup>0</sup>
6	n/a	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	R <sub>2</sub> <sup>4</sup>	R <sub>2</sub> <sup>3</sup>	R <sub>2</sub> <sup>2</sup>	R <sub>2</sub> <sup>1</sup>	R <sub>2</sub> <sup>0</sup>							

### 7.3.2 Input Data with Opacity ( $\alpha$ RGB)

LT768x provides a palette of 64 simultaneous colors from a total of 4096 different colors with opacity attribute for OSD (On-Screen-Display) application. Users may load preferred color into embedded color palette then pick it up by index color. The  $\alpha$  value stands for opacity.

**Table 7-17: 8bits MCU, 8bpp Mode ( $\alpha$ Index 2:6)**

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	$\alpha_1^3$	$\alpha_1^2$						Index color of pixel 0
2	$\alpha_3^3$	$\alpha_3^2$						Index color of pixel 1
3	$\alpha_5^3$	$\alpha_5^2$						Index color of pixel 2
4	$\alpha_7^3$	$\alpha_7^2$						Index color of pixel 3
5	$\alpha_9^3$	$\alpha_9^2$						Index color of pixel 4
6	$\alpha_{11}^3$	$\alpha_{11}^2$						Index color of pixel 5

$\alpha_x^3 \alpha_x^2 : 0 \rightarrow 100\%, 1 \rightarrow 20/32, 2 \rightarrow 11/32, 3 \rightarrow 0$

**Table 7-18: 8bits MCU, 16bpp Mode ( $\alpha$ RGB 4:4:4:4)**

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	$G_0^7$	$G_0^6$	$G_0^5$	$G_0^4$	$B_0^7$	$B_0^6$	$B_0^5$	$B_0^4$
2	$\alpha_0^3$	$\alpha_0^2$	$\alpha_0^1$	$\alpha_0^0$	$R_0^7$	$R_0^6$	$R_0^5$	$R_0^4$
3	$G_1^7$	$G_1^6$	$G_1^5$	$G_1^4$	$B_1^7$	$B_1^6$	$B_1^5$	$B_1^4$
4	$\alpha_1^3$	$\alpha_1^2$	$\alpha_1^1$	$\alpha_1^0$	$R_1^7$	$R_1^6$	$R_1^5$	$R_1^4$
5	$G_2^7$	$G_2^6$	$G_2^5$	$G_2^4$	$B_2^7$	$B_2^6$	$B_2^5$	$B_2^4$
6	$\alpha_2^3$	$\alpha_2^2$	$\alpha_2^1$	$\alpha_2^0$	$R_2^7$	$R_2^6$	$R_2^5$	$R_2^4$

$\alpha_x^3 \alpha_x^2 \alpha_x^1 \alpha_x^0 : 0 \rightarrow 100\%, 1 \rightarrow 30/32, 2 \rightarrow 28/32, 3 \rightarrow 26/32,$   
 $4 \rightarrow 24/32, \dots, 12 \rightarrow 8/32, 13 \rightarrow 6/32, 14 \rightarrow 4/32, 15 \rightarrow 0.$

**Table 7-19: 16bits MCU, Index Mode (Index 2:6)**

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0		
1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	$\alpha_0^3$	$\alpha_0^2$	Index color of pixel 0							
2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	$\alpha_1^3$	$\alpha_1^2$	Index color of pixel 1							
3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	$\alpha_2^3$	$\alpha_2^2$	Index color of pixel 2							
4	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	$\alpha_3^3$	$\alpha_3^2$	Index color of pixel 3							
5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	$\alpha_4^3$	$\alpha_4^2$	Index color of pixel 4							
6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	$\alpha_5^3$	$\alpha_5^2$	Index color of pixel 5							

$\alpha_x^3 \alpha_x^2 : 0 \rightarrow 0, 1 \rightarrow 11/32, 2 \rightarrow 20/32, 3 \rightarrow 100\%$

Table 7-20: 16bits MCU, 12bpp Mode ( $\alpha$ RGB 4:4:4:4)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	$\alpha_0^3$	$\alpha_0^2$	$\alpha_0^1$	$\alpha_0^0$	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	R <sub>0</sub> <sup>4</sup>	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>4</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>	B <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>4</sup>
2	$\alpha_1^3$	$\alpha_1^2$	$\alpha_1^1$	$\alpha_1^0$	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	R <sub>1</sub> <sup>4</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>4</sup>	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	B <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>4</sup>
3	$\alpha_2^3$	$\alpha_2^2$	$\alpha_2^1$	$\alpha_2^0$	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	R <sub>2</sub> <sup>4</sup>	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>4</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>	B <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>4</sup>
4	$\alpha_3^3$	$\alpha_3^2$	$\alpha_3^1$	$\alpha_3^0$	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	R <sub>3</sub> <sup>4</sup>	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	G <sub>3</sub> <sup>4</sup>	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>	B <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>4</sup>
5	$\alpha_4^3$	$\alpha_4^2$	$\alpha_4^1$	$\alpha_4^0$	R <sub>4</sub> <sup>7</sup>	R <sub>4</sub> <sup>6</sup>	R <sub>4</sub> <sup>5</sup>	R <sub>4</sub> <sup>4</sup>	G <sub>4</sub> <sup>7</sup>	G <sub>4</sub> <sup>6</sup>	G <sub>4</sub> <sup>5</sup>	G <sub>4</sub> <sup>4</sup>	B <sub>4</sub> <sup>7</sup>	B <sub>4</sub> <sup>6</sup>	B <sub>4</sub> <sup>5</sup>	B <sub>4</sub> <sup>4</sup>
6	$\alpha_5^3$	$\alpha_5^2$	$\alpha_5^1$	$\alpha_5^0$	R <sub>5</sub> <sup>7</sup>	R <sub>5</sub> <sup>6</sup>	R <sub>5</sub> <sup>5</sup>	R <sub>5</sub> <sup>4</sup>	G <sub>5</sub> <sup>7</sup>	G <sub>5</sub> <sup>6</sup>	G <sub>5</sub> <sup>5</sup>	G <sub>5</sub> <sup>4</sup>	B <sub>5</sub> <sup>7</sup>	B <sub>5</sub> <sup>6</sup>	B <sub>5</sub> <sup>5</sup>	B <sub>5</sub> <sup>4</sup>

$\alpha_x^3 \alpha_x^2 \alpha_x^1 \alpha_x^0 : 0 \rightarrow 0, 1 \rightarrow 2/32, 2 \rightarrow 4/32, 3 \rightarrow 6/32, 4 \rightarrow 8/32, \dots, 12 \rightarrow 24/32, 13 \rightarrow 26/32,$   
 $14 \rightarrow 28/32, 15 \rightarrow 100\%.$

## 8. Display Memory

LT768x has embedded 128Mb Display RAM. The Host(MCU) can save the displayed data to the internal Display RAM through designated instructions. LT768x's internal display engine will keep reading the image data stored in the Display RAM (first layer only), and sending them to the TFT driver for display. The amount of the image layers is related to the Display RAM capacity, picture resolution, and color depth, as shown in the following table:

**Table 8-1: LT768x' Model vs. Embedded Display RAM Capacity**

Model	Display RAM Capacity	Resolution (Max.)	Color (Max.)	Image Layer (@Max Color)
LT7681	128Mb	640*480	16.7M	18
LT7683	128Mb	1024*768	16.7M	10 (65K)
LT7686	128Mb	1280*1024	16.7M	4
LT7680A-R	128Mb	1280*1024	65K / 262K	6 (65K)
LT7680B-R	128Mb	480*320	65K / 262K	54 (65K)

The embedded Display RAM of LT768x is a High-Speed SDRAM (Synchronous Dynamic Random Access Memory). Before the Display RAM can be accessed, the relevant register initialization must be done based on the type of Display RAM. Please refer to the initialization steps as follows:

- Setup Register REG[E0h] according to the Display RAM capacity and model. The value of REG[E0h] must be set according to the LT768x model used to avoid display anomalies and image confusion. Please refer to Chapter 19, Table 19-6.
- Setup Register REG[E1H], REG[E2h], REG[E3h] including the CAS latency, refresh interval etc. The typical Display RAM refresh interval is 64ms. The value of these registers are recommended according to the LT768x model used. Please refer to Chapter 19, Table 19-7.
- Set Register REG[E4h] Bit0 to 1, and start Display RAM initialization process.
- Read Register REG[E4h] bit0, if it becomes 1, it means the initialization is complete.

## 8.1 Display RAM Data Structure

The image data stored in the Display RAM will be stored in different arrangement formats depending on the color (1bpp, 8bpp, 16bpp, 24bpp). Therefore, the Host must write the image data according to these formats, the following tables explain 8/16/24bpp RGB data arrangement format:

### 8.1.1 8bpp Display Data (RGB 3:3:2)

Table 8-2: 8bpp Display Data (RGB 3:3:2)

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>
0002h	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>
0004h	R <sub>5</sub> <sup>7</sup>	R <sub>5</sub> <sup>6</sup>	R <sub>5</sub> <sup>5</sup>	G <sub>5</sub> <sup>7</sup>	G <sub>5</sub> <sup>6</sup>	G <sub>5</sub> <sup>5</sup>	B <sub>5</sub> <sup>7</sup>	B <sub>5</sub> <sup>6</sup>	R <sub>4</sub> <sup>7</sup>	R <sub>4</sub> <sup>6</sup>	R <sub>4</sub> <sup>5</sup>	G <sub>4</sub> <sup>7</sup>	G <sub>4</sub> <sup>6</sup>	G <sub>4</sub> <sup>5</sup>	B <sub>4</sub> <sup>7</sup>	B <sub>4</sub> <sup>6</sup>
0006h	R <sub>7</sub> <sup>7</sup>	R <sub>7</sub> <sup>6</sup>	R <sub>7</sub> <sup>5</sup>	G <sub>7</sub> <sup>7</sup>	G <sub>7</sub> <sup>6</sup>	G <sub>7</sub> <sup>5</sup>	B <sub>7</sub> <sup>7</sup>	B <sub>7</sub> <sup>6</sup>	R <sub>6</sub> <sup>7</sup>	R <sub>6</sub> <sup>6</sup>	R <sub>6</sub> <sup>5</sup>	G <sub>6</sub> <sup>7</sup>	G <sub>6</sub> <sup>6</sup>	G <sub>6</sub> <sup>5</sup>	B <sub>6</sub> <sup>7</sup>	B <sub>6</sub> <sup>6</sup>
0008h	R <sub>9</sub> <sup>7</sup>	R <sub>9</sub> <sup>6</sup>	R <sub>9</sub> <sup>5</sup>	G <sub>9</sub> <sup>7</sup>	G <sub>9</sub> <sup>6</sup>	G <sub>9</sub> <sup>5</sup>	B <sub>9</sub> <sup>7</sup>	B <sub>9</sub> <sup>6</sup>	R <sub>8</sub> <sup>7</sup>	R <sub>8</sub> <sup>6</sup>	R <sub>8</sub> <sup>5</sup>	G <sub>8</sub> <sup>7</sup>	G <sub>8</sub> <sup>6</sup>	G <sub>8</sub> <sup>5</sup>	B <sub>8</sub> <sup>7</sup>	B <sub>8</sub> <sup>6</sup>
000Ah	R <sub>11</sub> <sup>7</sup>	R <sub>11</sub> <sup>6</sup>	R <sub>11</sub> <sup>5</sup>	G <sub>11</sub> <sup>7</sup>	G <sub>11</sub> <sup>6</sup>	G <sub>11</sub> <sup>5</sup>	B <sub>11</sub> <sup>7</sup>	B <sub>11</sub> <sup>6</sup>	R <sub>10</sub> <sup>7</sup>	R <sub>10</sub> <sup>6</sup>	R <sub>10</sub> <sup>5</sup>	G <sub>10</sub> <sup>7</sup>	G <sub>10</sub> <sup>6</sup>	G <sub>10</sub> <sup>5</sup>	B <sub>10</sub> <sup>7</sup>	B <sub>10</sub> <sup>6</sup>

### 8.1.2 16bpp Display Data (RGB 5:6:5)

Table 8-3: 16bpp Display Data (RGB 5:6:5)

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	R <sub>0</sub> <sup>4</sup>	R <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>4</sup>	G <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>2</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>	B <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>4</sup>	B <sub>0</sub> <sup>3</sup>
0002h	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	R <sub>1</sub> <sup>4</sup>	R <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>4</sup>	G <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>2</sup>	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	B <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>4</sup>	B <sub>1</sub> <sup>3</sup>
0004h	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	R <sub>2</sub> <sup>4</sup>	R <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>4</sup>	G <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>2</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>	B <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>4</sup>	B <sub>2</sub> <sup>3</sup>
0006h	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	R <sub>3</sub> <sup>4</sup>	R <sub>3</sub> <sup>3</sup>	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	G <sub>3</sub> <sup>4</sup>	G <sub>3</sub> <sup>3</sup>	G <sub>3</sub> <sup>2</sup>	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>	B <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>4</sup>	B <sub>3</sub> <sup>3</sup>
0008h	R <sub>4</sub> <sup>7</sup>	R <sub>4</sub> <sup>6</sup>	R <sub>4</sub> <sup>5</sup>	R <sub>4</sub> <sup>4</sup>	R <sub>4</sub> <sup>3</sup>	G <sub>4</sub> <sup>7</sup>	G <sub>4</sub> <sup>6</sup>	G <sub>4</sub> <sup>5</sup>	G <sub>4</sub> <sup>4</sup>	G <sub>4</sub> <sup>3</sup>	G <sub>4</sub> <sup>2</sup>	B <sub>4</sub> <sup>7</sup>	B <sub>4</sub> <sup>6</sup>	B <sub>4</sub> <sup>5</sup>	B <sub>4</sub> <sup>4</sup>	B <sub>4</sub> <sup>3</sup>
000Ah	R <sub>5</sub> <sup>7</sup>	R <sub>5</sub> <sup>6</sup>	R <sub>5</sub> <sup>5</sup>	R <sub>5</sub> <sup>4</sup>	R <sub>5</sub> <sup>3</sup>	G <sub>5</sub> <sup>7</sup>	G <sub>5</sub> <sup>6</sup>	G <sub>5</sub> <sup>5</sup>	G <sub>5</sub> <sup>4</sup>	G <sub>5</sub> <sup>3</sup>	G <sub>5</sub> <sup>2</sup>	B <sub>5</sub> <sup>7</sup>	B <sub>5</sub> <sup>6</sup>	B <sub>5</sub> <sup>5</sup>	B <sub>5</sub> <sup>4</sup>	B <sub>5</sub> <sup>3</sup>

### 8.1.3 24bpp Display Data (RGB 8:8:8)

Table 8-4: 24bpp Display Data (RGB 8:8:8)

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>4</sup>	G <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>2</sup>	G <sub>0</sub> <sup>1</sup>	G <sub>0</sub> <sup>0</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>	B <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>4</sup>	B <sub>0</sub> <sup>3</sup>	B <sub>0</sub> <sup>2</sup>	B <sub>0</sub> <sup>1</sup>	B <sub>0</sub> <sup>0</sup>
0002h	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	B <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>4</sup>	B <sub>1</sub> <sup>3</sup>	B <sub>1</sub> <sup>2</sup>	B <sub>1</sub> <sup>1</sup>	B <sub>1</sub> <sup>0</sup>	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	R <sub>0</sub> <sup>4</sup>	R <sub>0</sub> <sup>3</sup>	R <sub>0</sub> <sup>2</sup>	R <sub>0</sub> <sup>1</sup>	R <sub>0</sub> <sup>0</sup>
0004h	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	R <sub>1</sub> <sup>4</sup>	R <sub>1</sub> <sup>3</sup>	R <sub>1</sub> <sup>2</sup>	R <sub>1</sub> <sup>1</sup>	R <sub>1</sub> <sup>0</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>4</sup>	G <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>2</sup>	G <sub>1</sub> <sup>1</sup>	G <sub>1</sub> <sup>0</sup>
0006h	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>4</sup>	G <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>2</sup>	G <sub>2</sub> <sup>1</sup>	G <sub>2</sub> <sup>0</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>	B <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>4</sup>	B <sub>2</sub> <sup>3</sup>	B <sub>2</sub> <sup>2</sup>	B <sub>2</sub> <sup>1</sup>	B <sub>2</sub> <sup>0</sup>
0008h	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>	B <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>4</sup>	B <sub>3</sub> <sup>3</sup>	B <sub>3</sub> <sup>2</sup>	B <sub>3</sub> <sup>1</sup>	B <sub>3</sub> <sup>0</sup>	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	R <sub>2</sub> <sup>4</sup>	R <sub>2</sub> <sup>3</sup>	R <sub>2</sub> <sup>2</sup>	R <sub>2</sub> <sup>1</sup>	R <sub>2</sub> <sup>0</sup>
000Ah	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	R <sub>3</sub> <sup>4</sup>	R <sub>3</sub> <sup>3</sup>	R <sub>3</sub> <sup>2</sup>	R <sub>3</sub> <sup>1</sup>	R <sub>3</sub> <sup>0</sup>	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	G <sub>3</sub> <sup>4</sup>	G <sub>3</sub> <sup>3</sup>	G <sub>3</sub> <sup>2</sup>	G <sub>3</sub> <sup>1</sup>	G <sub>3</sub> <sup>0</sup>

### 8.1.4 Index Display with Opacity ( $\alpha$ RGB 2:2:2:2)

**Table 8-5: Index Display with Opacity ( $\alpha$ RGB 2:2:2:2)**

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	$\alpha_1^3$	$\alpha_1^2$	Index color of pixel 1				$\alpha_0^3$	$\alpha_0^2$	Index color of pixel 0							
0002h	$\alpha_3^3$	$\alpha_3^2$	Index color of pixel 3				$\alpha_2^3$	$\alpha_2^2$	Index color of pixel 2							
0004h	$\alpha_5^3$	$\alpha_5^2$	Index color of pixel 5				$\alpha_4^3$	$\alpha_4^2$	Index color of pixel 4							
0006h	$\alpha_7^3$	$\alpha_7^2$	Index color of pixel 7				$\alpha_6^3$	$\alpha_6^2$	Index color of pixel 6							
0008h	$\alpha_9^3$	$\alpha_9^2$	Index color of pixel 9				$\alpha_8^3$	$\alpha_8^2$	Index color of pixel 8							
000Ah	$\alpha_{11}^3$	$\alpha_{11}^2$	Index color of pixel 11				$\alpha_{10}^3$	$\alpha_{10}^2$	Index color of pixel 10							

$\alpha_x^3 \alpha_x^2 : 0 \rightarrow 0, 1 \rightarrow 11/32, 2 \rightarrow 20/32, 3 \rightarrow 100\%$

### 8.1.5 12bpp Display with Opacity ( $\alpha$ RGB 4:4:4:4)

**Table 8-6: 12bpp Display with Opacity ( $\alpha$ RGB 4:4:4:4)**

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	$\alpha_0^3$	$\alpha_0^2$	$\alpha_0^1$	$\alpha_0^0$	$R_0^7$	$R_0^6$	$R_0^5$	$R_0^4$	$G_0^7$	$G_0^6$	$G_0^5$	$G_0^4$	$B_0^7$	$B_0^6$	$B_0^5$	$B_0^4$
0002h	$\alpha_1^3$	$\alpha_1^2$	$\alpha_1^1$	$\alpha_1^0$	$R_1^7$	$R_1^6$	$R_1^5$	$R_1^4$	$G_1^7$	$G_1^6$	$G_1^5$	$G_1^4$	$B_1^7$	$B_1^6$	$B_1^5$	$B_1^4$
0004h	$\alpha_2^3$	$\alpha_2^2$	$\alpha_2^1$	$\alpha_2^0$	$R_2^7$	$R_2^6$	$R_2^5$	$R_2^4$	$G_2^7$	$G_2^6$	$G_2^5$	$G_2^4$	$B_2^7$	$B_2^6$	$B_2^5$	$B_2^4$
0006h	$\alpha_3^3$	$\alpha_3^2$	$\alpha_3^1$	$\alpha_3^0$	$R_3^7$	$R_3^6$	$R_3^5$	$R_3^4$	$G_3^7$	$G_3^6$	$G_3^5$	$G_3^4$	$B_3^7$	$B_3^6$	$B_3^5$	$B_3^4$
0008h	$\alpha_4^3$	$\alpha_4^2$	$\alpha_4^1$	$\alpha_4^0$	$R_4^7$	$R_4^6$	$R_4^5$	$R_4^4$	$G_4^7$	$G_4^6$	$G_4^5$	$G_4^4$	$B_4^7$	$B_4^6$	$B_4^5$	$B_4^4$
000Ah	$\alpha_5^3$	$\alpha_5^2$	$\alpha_5^1$	$\alpha_5^0$	$R_5^7$	$R_5^6$	$R_5^5$	$R_5^4$	$G_5^7$	$G_5^6$	$G_5^5$	$G_5^4$	$B_5^7$	$B_5^6$	$B_5^5$	$B_5^4$

$\alpha_x^3 \alpha_x^2 \alpha_x^1 \alpha_x^0 : 0 \rightarrow 0, 1 \rightarrow 2/32, 2 \rightarrow 4/32, 3 \rightarrow 6/32, 4 \rightarrow 8/32, \dots, 12 \rightarrow 24/32, 13 \rightarrow 26/32, 14 \rightarrow 28/32, 15 \rightarrow 100\%.$

## 8.2 Color Palette RAM

**Table 8-7: Color Palette RAM**

Addr	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	$R_0^7$	$R_0^6$	$R_0^5$	$R_0^4$	$G_0^7$	$G_0^6$	$G_0^5$	$G_0^4$	$B_0^7$	$B_0^6$	$B_0^5$	$B_0^4$
0002h	$R_1^7$	$R_1^6$	$R_1^5$	$R_1^4$	$G_1^7$	$G_1^6$	$G_1^5$	$G_1^4$	$B_1^7$	$B_1^6$	$B_1^5$	$B_1^4$
0004h	$R_2^7$	$R_2^6$	$R_2^5$	$R_2^4$	$G_2^7$	$G_2^6$	$G_2^5$	$G_2^4$	$B_2^7$	$B_2^6$	$B_2^5$	$B_2^4$
0006h	$R_3^7$	$R_3^6$	$R_3^5$	$R_3^4$	$G_3^7$	$G_3^6$	$G_3^5$	$G_3^4$	$B_3^7$	$B_3^6$	$B_3^5$	$B_3^4$
0008h	$R_4^7$	$R_4^6$	$R_4^5$	$R_4^4$	$G_4^7$	$G_4^6$	$G_4^5$	$G_4^4$	$B_4^7$	$B_4^6$	$B_4^5$	$B_4^4$
000Ah	$R_5^7$	$R_5^6$	$R_5^5$	$R_5^4$	$G_5^7$	$G_5^6$	$G_5^5$	$G_5^4$	$B_5^7$	$B_5^6$	$B_5^5$	$B_5^4$

## 9. LCD Interface

LT768x Supports 16, 18, 24bits RGB interface of TFT Panel, no matter the color depth is 24bpp (RGB 8:8:8), 16bpp (RGB 5:6:5) or 8bpp (RGB 3:3:2), the display data can be sent to the TFT Driver on the TFT panel through these RGB interfaces. The LT768x LCD Display data that correspond to the RGB data are shown in the below Table 9-1. The Host can setup REG[01h] Bit[4:3] to select 16bits, 18bits or 24bits resolution.

The RGB data supported by different models of LT768x are also different. For example, when using LT7680A, even if REG[01h] bit[4:3] is set to 00b (24bits), the shown color depth will still be 18bits only. Please refer to Table 9-2 for RGB data supported by different models of LT768x.

**Table 9-1: RGB Interface VS. RGB Display Data**

LCD Data Bus	TFT-LCD Interface		
	REG[01h] bit[4:3] = 10b (16bits)	REG[01h] bit[4:3] = 01b (18bits)	REG[01h] bit[4:3] = 00b (24bits)
PD[0]			B0
PD[1]			B1
PD[2]		B0	B2
PD[3]	B0	B1	B3
PD[4]	B1	B2	B4
PD[5]	B2	B3	B5
PD[6]	B3	B4	B6
PD[7]	B4	B5	B7
PD[8]			G0
PD[9]			G1
PD[10]	G0	G0	G2
PD[11]	G1	G1	G3
PD[12]	G2	G2	G4
PD[13]	G3	G3	G5
PD[14]	G4	G4	G6
PD[15]	G5	G5	G7
PD[16]			R0
PD[17]			R1
PD[18]		R0	R2
PD[19]	R0	R1	R3
PD[20]	R1	R2	R4
PD[21]	R2	R3	R5
PD[22]	R3	R4	R6
PD[23]	R4	R5	R7

Table 9-2: RGB Data Signals of LT768x

Model	LCD Data Bus	RGB Signal Number	Colors
LT7681	PD[23~0]	R:G:B = 8:8:8	16.7M Color
LT7683	PD[23~0]	R:G:B = 8:8:8	16.7M Color
LT7686	PD[23~0]	R:G:B = 8:8:8	16.7M Color
LT7680A	PD[23~18], PD[15~10], PD[7~23]	R:G:B = 6:6:6	262K Color
LT7680B	PD[23~18], PD[15~10], PD[7~23]	R:G:B = 6:6:6	262K Color

Figure 9-1 is the timing diagram of the output signals from LT768x to the TFT-LCD. In addition to the RGB data lines mentioned above, LT768x also provides PCLK (Panel Scan Clock), VSYNC Pulse, HSYNC Pulse and Data Enable signals. The frequency of PCLK is setup by REG[05h] and REG[06h]. Please refer to the description in Section 6.1 and Chapter 19th.

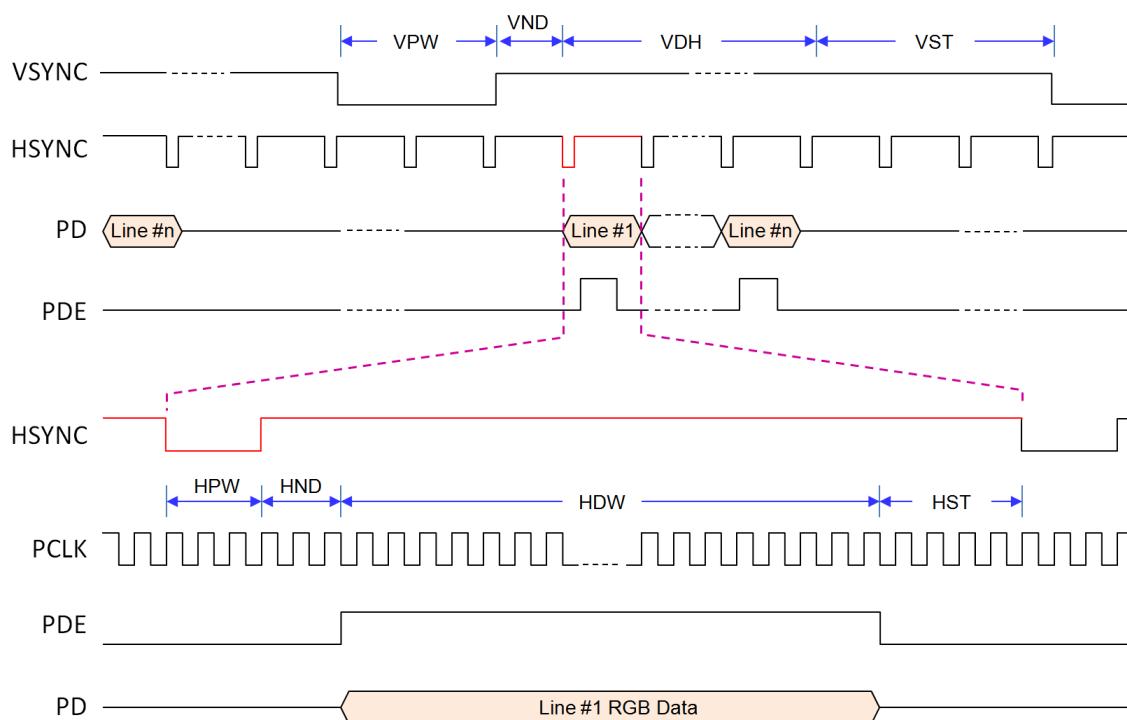


Figure 9-1: TFT-LCD Interface Timing

## 10. Display Function

### 10.1 Color Bar

LT768x provides a color bar display, which can be used as a display test and does not require display memory. The function can be performed by setting REG[12h] bit5 to 1.

Row number	
#1 ~ #32	Color set to R(00h), G(00h), B(00h)
#33 ~ #64	Color set to R(00h), G(00h), B(FFh)
#65 ~ #96	Color set to R(00h), G(FFh), B(00h)
#97 ~ #128	Color set to R(00h), G(FFh), B(FFh)
#129 ~ #160	Color set to R(FFh), G(00h), B(00h)
#161 ~ #192	Color set to R(FFh), G(00h), B(FFh)
#193 ~ #224	Color set to R(FFh), G(FFh), B(00h)
#225 ~ #256	Color set to R(FFh), G(FFh), B(FFh)
:	:
:	:
:	:
Last Row	( Repeat above eight Color)

Figure 10-1: Color Bar

### 10.2 Main Window

The LCD main window size can be defined by setting Registers REG[14h] ~ REG[1Fh]. Users can save images to memory buffers first. By setting up the related Registers (REG[20h] ~ REG[29h]), users can then show the specific images they want.

#### 10.2.1 Configure Display Image Buffer

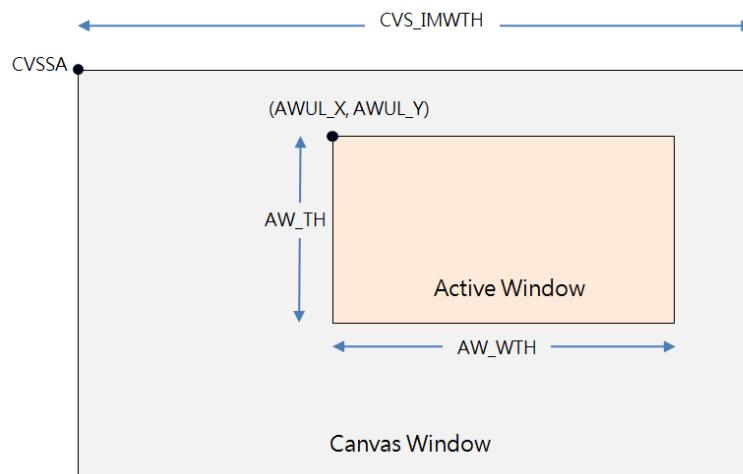
Display RAM is used to store images, and how many images can be stored is determined by the image resolution and the color depth. For example, if the image resolution is 800\*480, with 65K color (16bits), then 21 images can be stored in 128Mbits Display RAM:

$$\text{Number of Image} = (128 \times 1024 \times 1024) / (800 \times 480 \times 16) = 21.8$$

If the image resolution is 480\*272, with 65K color (16bits), then 64 image can be stored in 128Mbits Display RAM:

$$\text{Number of Image} = (128 \times 1024 \times 1024) / (480 \times 272 \times 16) = 64.3$$

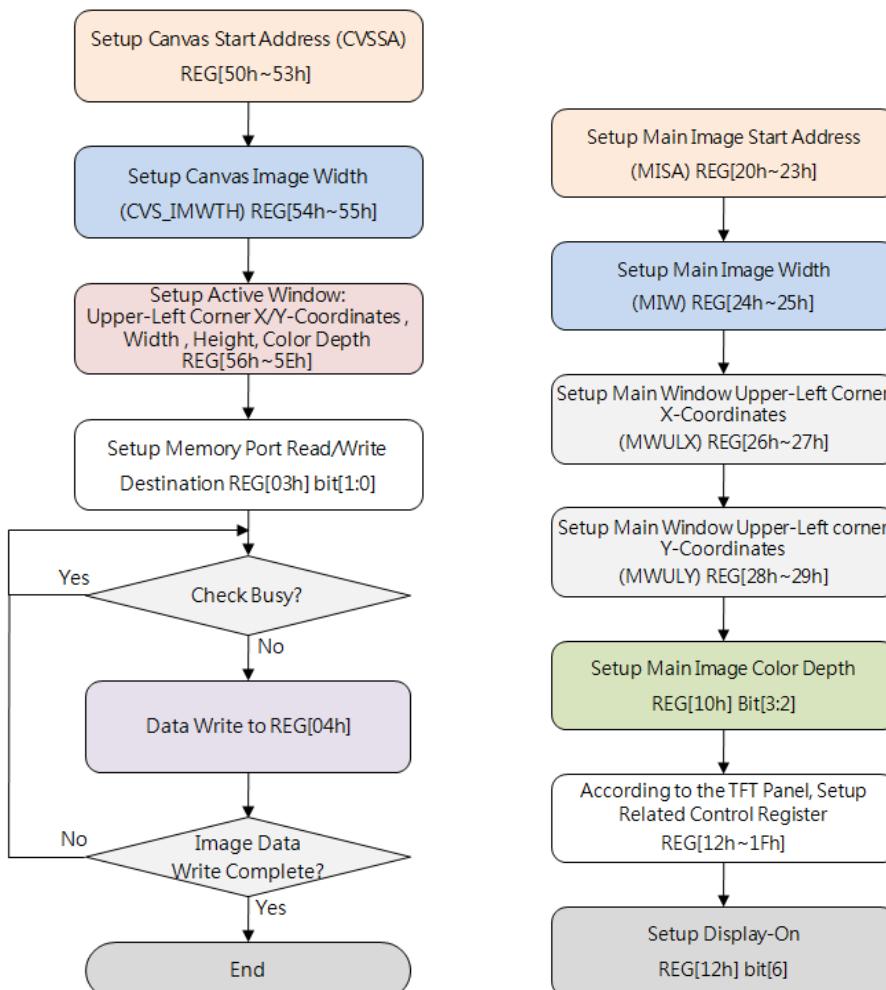
In LT768x, the Starting Position and Width of the Canvas, and the Working window range must be set before writing the image data to Display RAM. Please refer to the description of the registers REG[50H] ~ REG[5Eh] in Chapter 19.



**Figure 10-2: Canvas Window and Active Window**

### 10.2.2 Write Image Data to Display Image Buffer

The following diagram is a flowchart about writing image data to Display RAM and displaying the main window image on an LCD screen:



**Figure 10-3: Write Image Data to Display Image Buffer**

### 10.2.3 Display Main Window Image

Main window displays the selected image by setting the Registers REG[20h] ~ REG[29h]. The following figure is the process for setting up the main window:

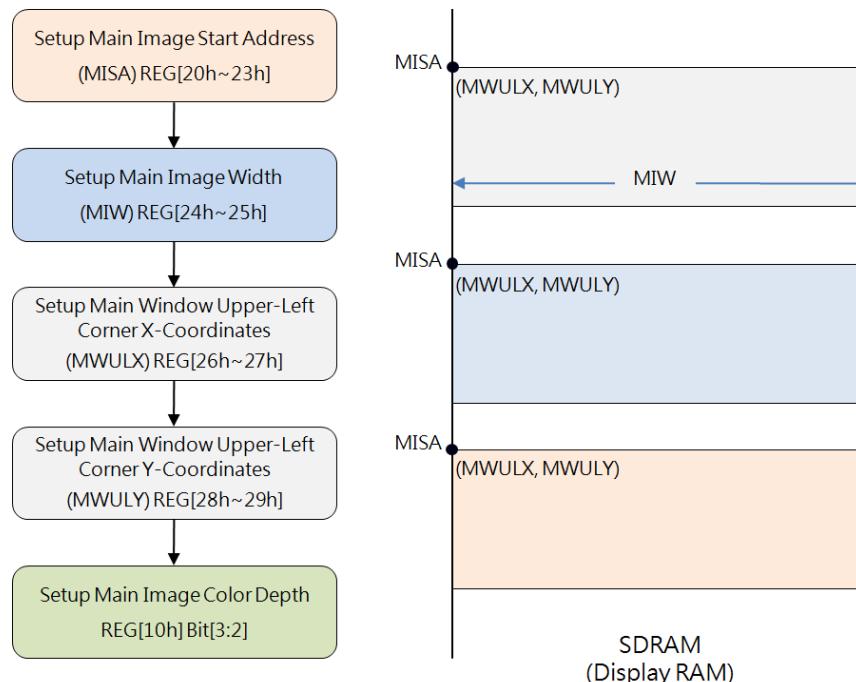


Figure 10-4: Setup and Select Main Window Image

### 10.3 Picture-In-Picture (PIP)

LT768x 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 size and location of the picture window is set by the register REG[2Ah] ~ REG[3BH] and REG[11h]. The parameters of PIP-1 and PIP-2 are using the same registers, whereas REG[10H] bit4 is used to decide whether the parameters are for PIP-1 or PIP-2. In addition, before using the PIP function, the relevant parameters of the display window must be set.

The unit of PIP windows sizes and start positions is 4 pixels in horizontal, and 1 pixel in vertical. In PIP mode, LT768x does not support the overlapping of transparent display, and when REG[12h] bit3 VDIR = 1, then the function of PIP windows, graphics cursors, and text cursors will be automatically prohibited.

#### 10.3.1 PIP Window Setting

To apply PIP feature, the Host has to setup the PIP Image Start Address, Image Width, Display X/Y coordinates, Image X/Y coordinates, PIP Windows Color Depth, PIP Window Width and PIP Window Height registers. The following figure is the flowchart for setting the PIP and displaying it on the main window.

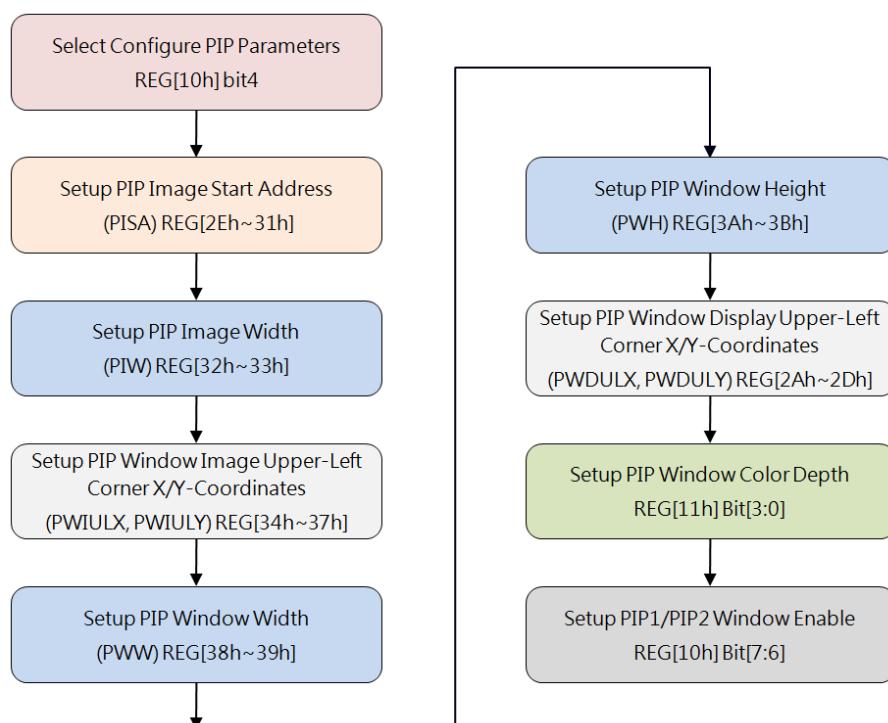


Figure 10-5: Initialize PIP Function

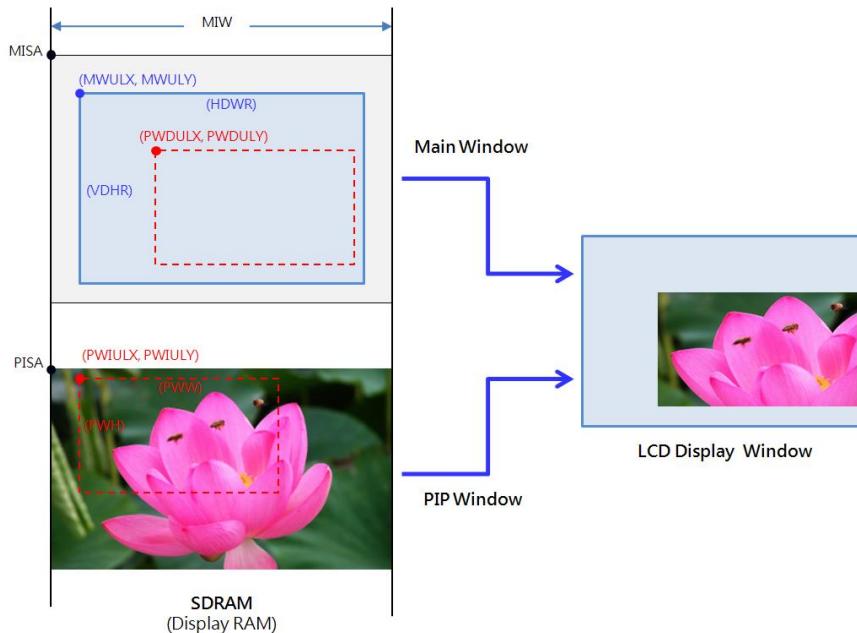


Figure 10-6: PIP Example

### 10.3.2 PIP Display Position and PIP Image Position

As described in the previous section of the PIP display flowchart, it is known that the final position of the PIP on the LCD screen can be changed by setting up PWDULX and PWDULY. Setting PISA, PIW, PWIULX, PWIULY can change the position of the PIP images that users want to display. These actions do not change any image data that exists in Display RAM, but can easily change the displayed picture on the panel. The following example shows a main window with a PIP window. Users can display different PIP images by changing the PIP Position Register (PWDULX and PWDULY) of the PIP window.

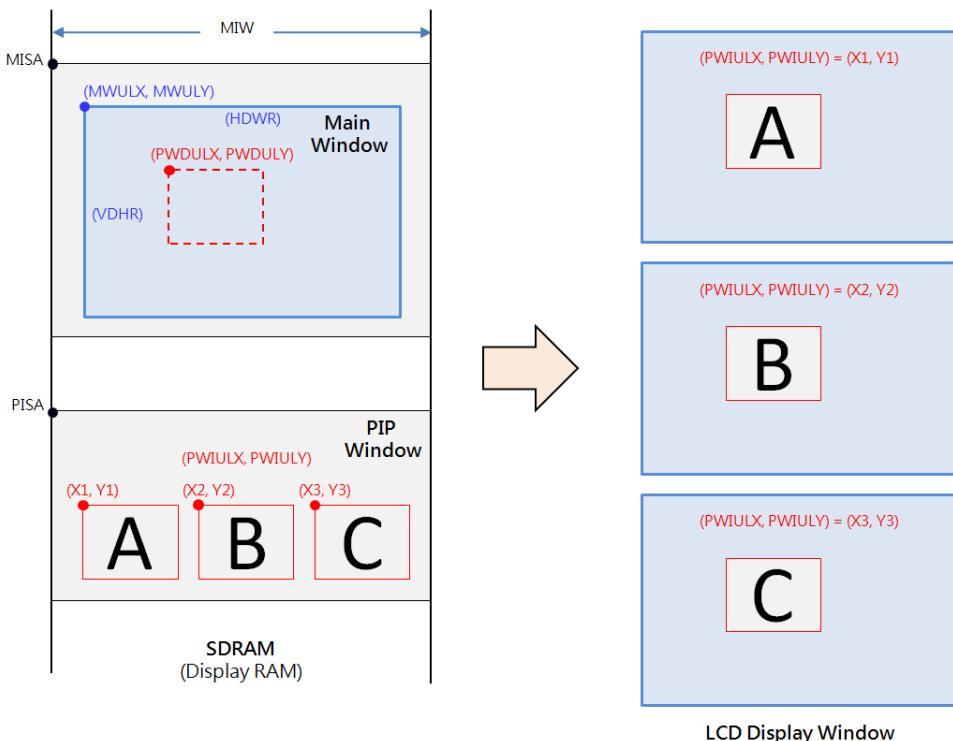


Figure 10-7: Select Different PIP image on the Screen

## 10.4 Image Rotate and Mirror

Usually LCD displays are refreshed horizontally (from left to right and then from top to bottom), and the displayed images are stored in the same way. LT768x provides Rotate and Mirror functions. The rotate feature rotates the displayed image in the counterclockwise direction of 90° or 180°. The mirror feature is the image that is displayed from right to left. LT768x has embedded a hardware controller to execute Rotation and Mirroring features, therefore, it can greatly save the Host processing time.

The REG[02H] bit[2:1] are used to control the Memory Store Direction. These two bits are available only for Graphic Mode.

- 00b: Left → Right, then Top → Bottom (Original)
- 01b: Right → Left, then Top → Bottom (Horizontal flip)
- 10b: Top → Bottom, then Left → Right (Rotate right 90° & Horizontal flip)
- 11b: Bottom → Top, then Left → Right (Rotate left 90°)

Followings are some examples for Image Rotate and Mirror:



Figure 10-8: Original Image – without Rotation

### 1. When VDIR (REG[12h] bit3)= 0

When setting REG[02H] bit[2:1] to 00b, its definition is to write image data from left to right and then top to bottom. This will show the original image as Figure 10-8. If setting REG[02H] bit[2:1] to 01b, it means writing image data from right to left and then from top to bottom. So the image displayed will be a horizontal mirror image, as shown in Figure 10-9.



Figure 10-9: Horizontal Mirror Image

When REG[02H] bit[2:1] is set to 10b, it means writing image data from top to bottom and then left to right. So the displayed image will be rotated to the right 90° and then flipped horizontally as shown in Figure 10-10.



Figure 10-10: Rotated to the Right 90° and Flipped Horizontally

When REG[02H] bit[2:1] is set to 11b, the image is written from bottom to top and then left to right. So the display image will rotate 90° to the left as shown in Figure 10-11.



Figure 10-11: Rotate 90° to the Left

## 2. When VDIR (REG[12h] bit3) = 1,

When REG[02h] bit[2:1] is set to 00b, the display image will be as following Figure:



Figure 10-12: Flip with Vertical

When REG[02h] bit[2:1] is set to 01b, the display image will be rotated 180°:



**Figure 10-13: Rotated 180°**

When REG[02h] bit[2:1] is set to 10b, The displayed image will rotate 90° to the left:



**Figure 10-14: Rotate 90° to the Left**

When REG[02h] bit[2:1] is set to 11b, the display image will be as following Figure:



**Figure 10-15: Rotated to the Left 90° then Vertically Mirrored**

## 11. Geometric Drawing Engine

### 11.1 Drawing Circle and Ellipse

LT768x supports the function of drawing Circles and Ellipses. As long as the Host sets the Circle or the Ellipse Center (REG[7Bh ~ 7Eh]), Radius (REG[77h ~ 7Ah]), and the Color of the Circle (REG[D2h ~ D4h]), then specifies REG[76h] bit[5:4] to 00b, and finally enables the Drawing Circle function (REG[76h] bit7 = 1), then LT768x will draw a Circle or Ellipse on the screen automatically.

Host can also set REG[76h] Bit6 to 1 to fill a Circle or an Ellipse. Therefore, Host does not need to consume many resources to calculate the location or to write series of data to the display memory. An Ellipse has two radius. To draw a Circle, simply set the long radius and short radius to the same value (R1 = R2).

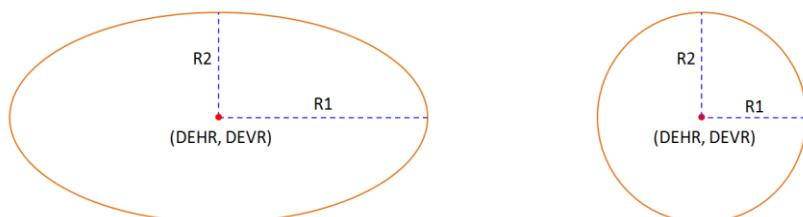


Figure 11-1: Drawing Ellipse and Circle

The flowchart for drawing circles and drawing ellipses is as follows. The left flowchart is to draw a circle or ellipses without fill, and the right flowchart is with fill. Please note the center point of the Circle must be located within the Active Window.

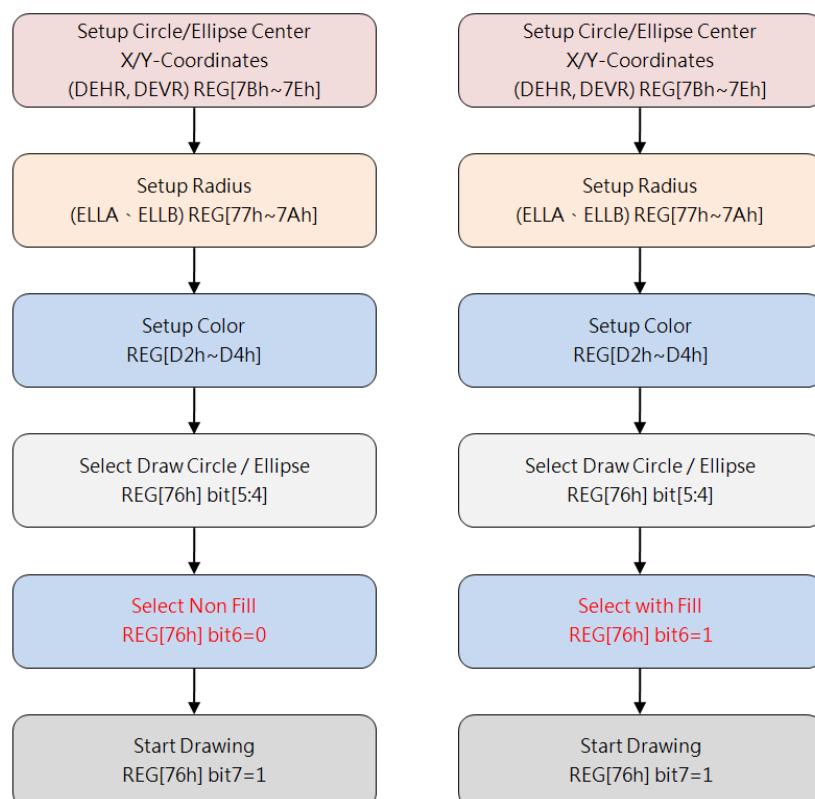


Figure 11-2: Flowchart of Drawing Circle and Ellipse

## 11.2 Drawing Curve

LT768x supports the function of drawing Curves. As long as the Host sets the Curve Center (REG[7Bh ~ 7Eh]), Radius (REG[77h ~ 7Ah]), and the Color of the Curve (REG[D2h ~ D4h]), then specifies REG[76h] bit[5:4] to 01b, and finally enables the Drawing Curve function (REG[76h] bit7 = 1), then LT768x will draw a one-fourth Curve on the screen automatically.

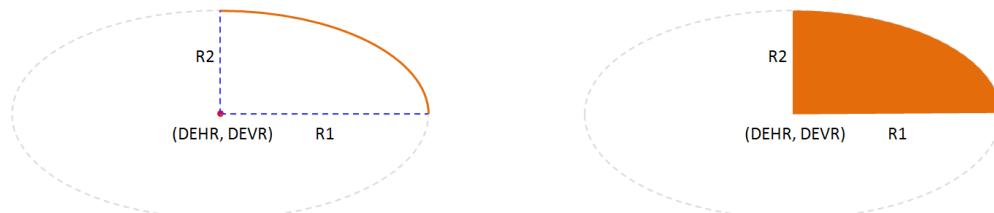


Figure 11-3: Drawing Curve and one-fourth Ellipse

The flowchart for drawing a Curve or drawing one-fourth Ellipses is as follows. The left flowchart is to draw a Curve or one-fourth Ellipse without fill, and the right flowchart is with fill. Please note the center point of the Curve must be located within the Active Window.

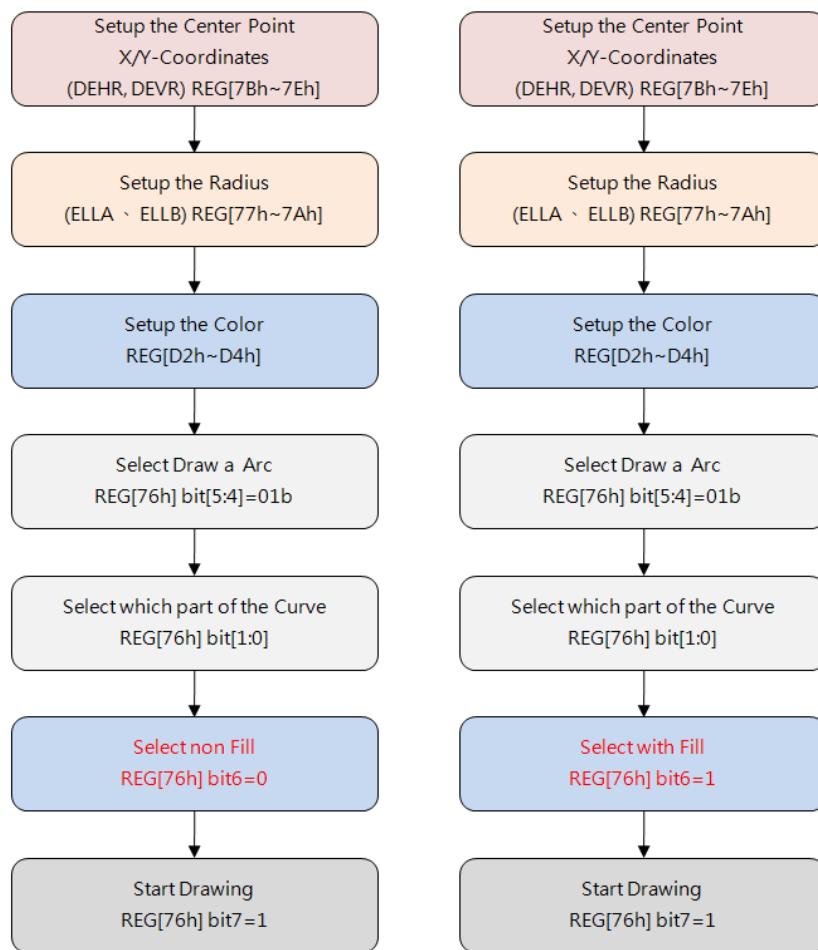
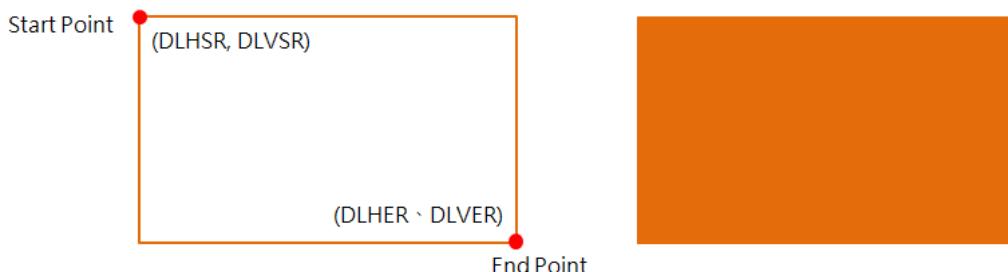


Figure 11-4: Flowchart of Drawing Curve and one-fourth Ellipse

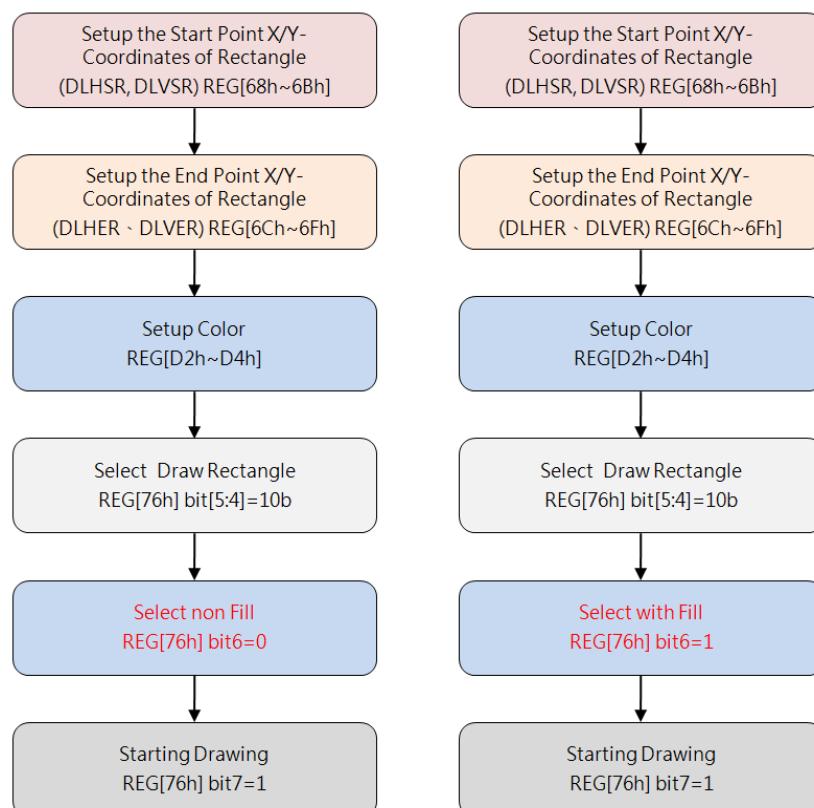
### 11.3 Drawing Rectangle

To draw a Rectangle, the Host has to set the Start Point Coordinates (REG[68h ~ 6Bh]), Stop Point Coordinates (REG[6Ch ~ 6Fh]), and the Color of the Rectangle (REG[D2h ~ D4h]), then specify REG[76h] bit[5:4] to 10b, and finally enables the drawing function (REG[76h] bit7 = 1), then LT768x will draw a Rectangle on the screen automatically. Host can also set REG[76h] Bit6 to 1 to fill the Rectangle with a specified color.



**Figure 11-5: Drawing Rectangle**

The following figure is the flowchart of drawing Rectangles. The left flowchart is to draw a Rectangle without fill, and the right flowchart is with fill. Please note the Start and End points must be located within the Active Window.



**Figure 11-6: Flowchart of Drawing a Rectangle**

## 11.4 Draw Line

To draw a Line, the Host has to set the Start Point Coordinates (REG[68h ~ 6Bh]), End Point Coordinates (REG[6Ch ~ 6Fh]), and the Color of the Line (REG[D2h ~ D4h]), then specify REG[67h] bit1 to 0, and finally enable the Line Drawing function (REG[67h] bit7 = 1), then LT768x will draw a Line on the screen automatically.

The following figure is the flowchart of Line drawing. Please note the Start and End point must be located within the Active Window.

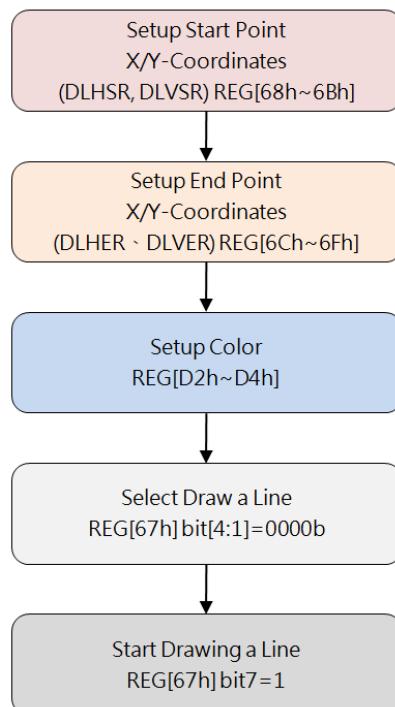
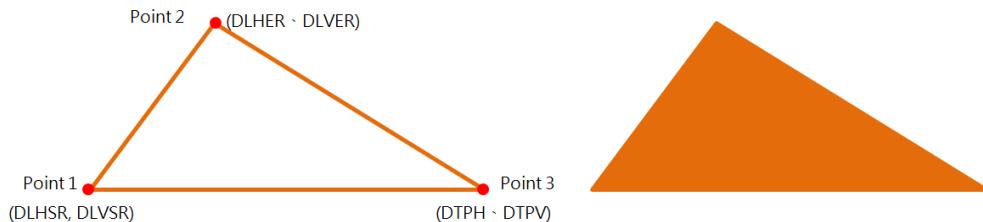


Figure 11-7: Flowchart of Drawing a Line

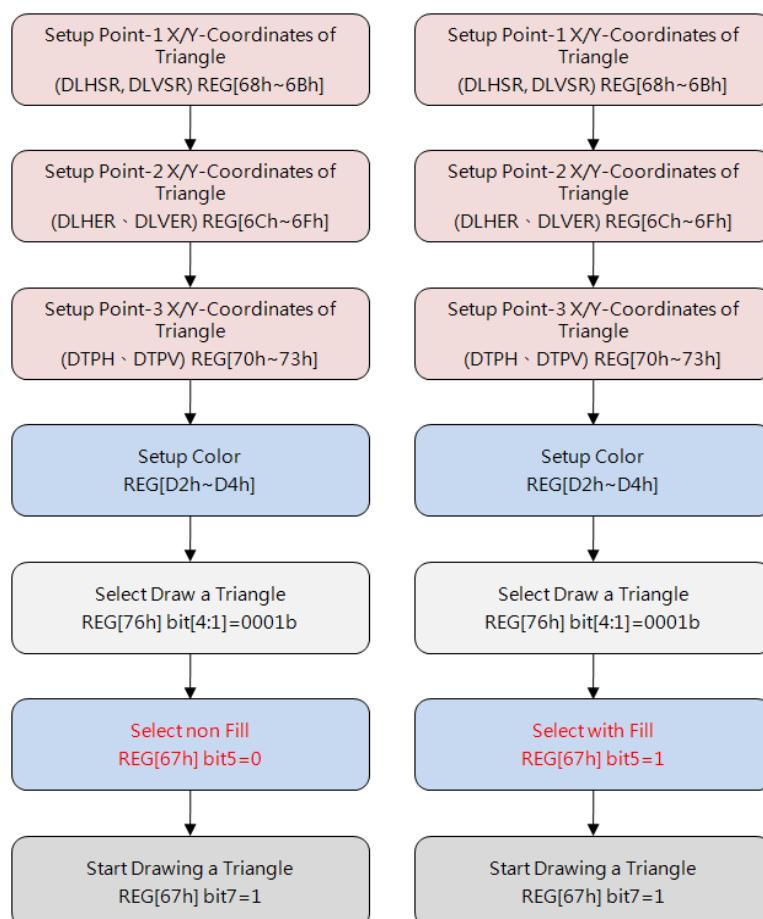
## 11.5 Drawing Triangle

To draw a triangle, the Host has to set the 1<sup>ST</sup> Point Coordinates (REG[68h ~ 6Bh]) of Triangle, the 2<sup>nd</sup> Point Coordinates (REG[6Ch ~ 6Fh]), 3<sup>rd</sup> Point Coordinates (REG[70h ~ 73h]), and the Color of the Triangle (REG[D2h ~ D4h]), then specify REG[67h] bit1 to 1, and finally enable the drawing function (REG[67h] bit7 = 1), then LT768x will draw a triangle on the screen automatically. Host can also set REG[67h] bit5 to 1 to fill the Rectangle with a specified color.



**Figure 11-8: Drawing Triangle**

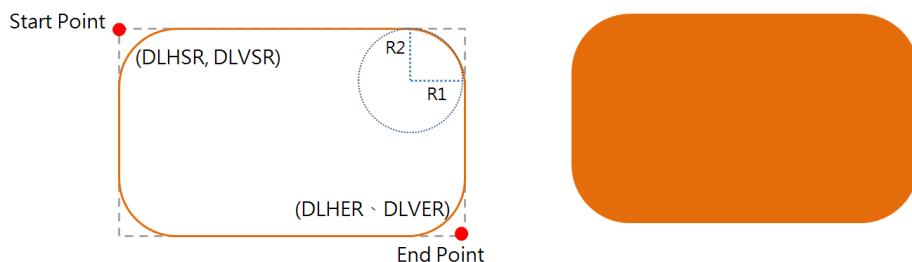
The following figure is the flowchart of drawing triangles. The left flowchart is to draw a triangle without fill, and the right flowchart is with fill. Please note the three points must be located within the Active Window.



**Figure 11-9: Flowchart of Drawing a Triangle**

## 11.6 Drawing Rounded-Rectangle

To draw a rounded-rectangle, the Host has to set the Start Point Coordinates (REG[68h ~ 6Bh]), End Point Coordinates (REG[6Ch ~ 6Fh]), the Rounded Radius (REG[77h ~ 7Ah]), and the Color (REG[D2h ~ D4h]), then specify REG[76h] bit[5:4] to 11b, and finally enable the drawing function (REG[76h] bit7 = 1), then LT768x will draw a rounded-rectangle on the screen automatically. Host can also set REG[76h] Bit6 to 1 to fill the rounded-rectangle with a specified color. The following figure is a schematic of a Rounded-Rectangle, in which the R1 represents the long axis radius, and the R2 represents the short axis radius.

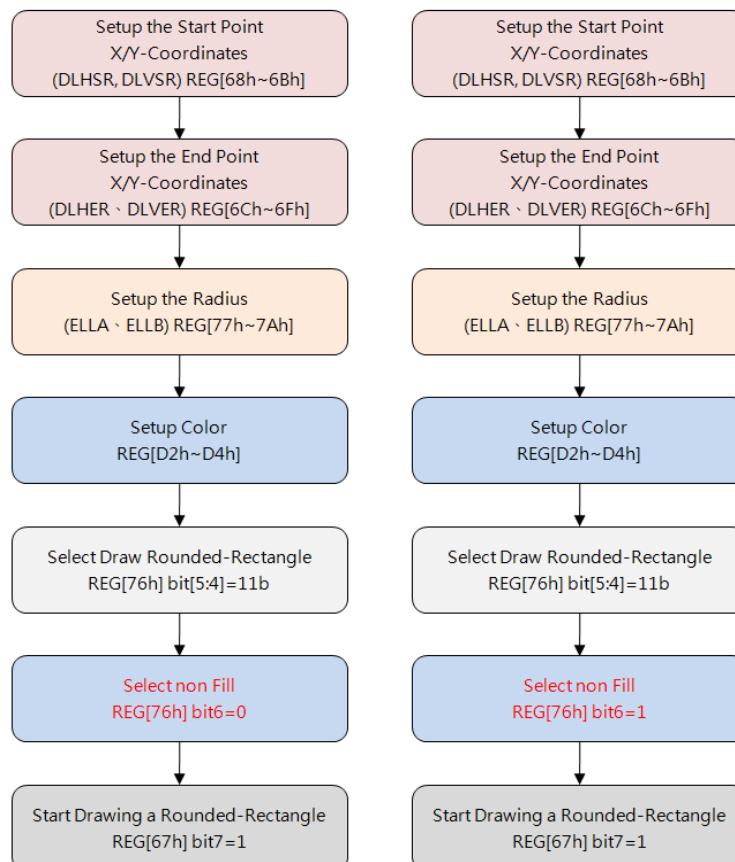


**Figure 11-10: Drawing Rounded Triangle**

**Note1:** DLHER-DLHSR must be larger than  $2 \times R1 + 1$

**Note2:** DLVER-DLVSR must be larger than  $2 \times R2 + 1$

The following figure is the flowchart of drawing a rounded-rectangle. The left flowchart is to draw a rounded-rectangle without fill, and the right flowchart is with fill. Please note the Start and End point must be located within the Active Window.



**Figure 11-11: Flowchart of Drawing a Rounded-Rectangle**

## 12. Block Transfer Engine (BTE)

LT768x has an embedded high performance hardware engine - Block Transfer Engine (BTE), it is designed to accelerate data Loading, Transferring, and additional Logic Processing. BTE supports 13 basic BTE Operations (Table 12-1), as well as additional functions of Raster Operation (ROP, Table 12-2), Chroma Key, Color Expansion, and so on. After getting properly set and enabled, BTE can perform the required operations and functions automatically and rapidly regardless of MCU. BTE can extremely ease MCU loading and greatly improve the performance for the whole system.

If a data block needs to be loaded, moved, and processed at the same time, users can get it done by BTE. Through BTE, users can implement many functions by the combinations of basic BTE Operations and various available logic combinations of Source and Destination. Once the BTE is properly set and started, it will keep active (busy) until the task is done. There are two ways to get BTE working status, one is to monitor Hardware Interruption: connecting INT# to MCU, once an interrupt is issued, the MCU responds and starts the routine to get the Interruption Source by checking REG[0Ch]; the other is to check the corresponding Flag bits of register BTE\_CTRL0 (REG[90h]) bit4, or Status Register (STSR) bit3.

**Table 12-1: BTE Operation**

BTE Operation Code REG[91h] Bits [3:0]	BTE Operation Description
0000b	MCU Write with ROP.
0010b	Memory Copy (move) with ROP.
0100b	MCU Write with chroma keying (without ROP)
0101b	Memory Copy (move) with chroma keying (without ROP)
0110b	Pattern Fill with ROP
0111b	Pattern Fill with chroma keying (without ROP)
1000b	MCU Write with Color Expansion (without ROP)
1001b	MCU Write with Color Expansion and chroma keying (without ROP)
1010b	Memory Copy with opacity (without ROP)
1011b	MCU Write with opacity (without ROP)
1100b	Solid Fill (without ROP)
1110b	Memory Copy with Color Expansion (without ROP)
1111b	Memory Copy with Color Expansion and chroma keying (without ROP)
Other Combinations	Reserved

Table 12-2: ROP Function

ROP Function Code REG[91h] bit[7:4]	Function Description (Boolean)
0000b	0 (Blackness)
0001b	$\sim S_0 \cdot \sim S_1$ or $\sim (S_0 + S_1)$
0010b	$\sim S_0 \cdot S_1$
0011b	$\sim S_0$
0100b	$S_0 \cdot \sim S_1$
0101b	$\sim S_1$
0110b	$S_0 \wedge S_1$
0111b	$\sim S_0 + \sim S_1$ or $\sim (S_0 \cdot S_1)$
1000b	$S_0 \cdot S_1$
1001b	$\sim (S_0 \wedge S_1)$
1010b	$S_1$
1011b	$\sim S_0 + S_1$
1100b	$S_0$
1101b	$S_0 + \sim S_1$
1110b	$S_0 + S_1$
1111b	1 (Whiteness)

**Note:**

1. Source 0 (S0) Data: comes from MCU Writing or Memory
2. Source 1 (S1) Data: comes from Memory,
3. Destination (DT) Data: be written to Memory
4. Memory means internal Display RAM
5. Memory, S0, S1, and DT will always be quoted in this Chapter. For Example: If ROP function REG[91h] bit[7:4]=0xCh, then “DT = S0”, it means “Transfer Source 0 data to Destination”; If ROP function REG[91h] bit[7:4]=0xEh, then “DT = S0 + S1”, it means “Source 0 data + Source 1 data, and then transfer the result to the Destination”.

Table 12-3: Color Expansion Function

ROP Function REG[91h] bit[7:4]	Start Bit Position for Color Expansion	
	BTE operation code = 1000/1001/1110/1111	
	16bits MCU Interface	8bits MCU Interface
0000b	Bit0	Bit0
0001b	Bit1	Bit1
0010b	Bit2	Bit2
0011b	Bit3	Bit3
0100b	Bit4	Bit4
0101b	Bit5	Bit5
0110b	Bit6	Bit6
0111b	Bit7	Bit7
1000b	Bit8	Invalid
1001b	Bit9	Invalid
1010b	Bit10	Invalid
1011b	Bit11	Invalid
1100b	Bit12	Invalid
1101b	Bit13	Invalid
1110b	Bit14	Invalid
1111b	Bit15	Invalid

## 12.1 BTE Basic Settings

The BTE basic settings, including Memory Start Address, Image Width, and Start Point Position (Coordinate) for each Source and Destination, can be defined by following registers:

1. S0 Address Registers: REG [93h], REG[94h], REG[95h], REG[96h], REG[97h], REG [98h], REG[99h], REG[9Ah], REG[9Bh], REG[9Ch]
2. S1 Address Registers: REG [9Dh], REG[9Eh], REG[9Fh], REG[A0h], REG [A1h], REG[A2h], REG[A3h], REG[A4h], REG[A5h], REG[A6h]
3. DT Address Registers: REG [A7h], REG[A8h], REG[A9h], REG[AAh], REG [ABh], REG[ACh], REG[ADh], REG[AEh], REG[AFh], REG[B0h]

## 12.2 Color Palette RAM

LT768x has an embedded Color Palette RAM for 8-bits Alpha Blend function. Real colors can be retrieved by searching Color Palette RAM. As shown in Figure 12-1, Color Palette RAM is organized by 64\*12bits. Since MCU writes 8 bits data at a time, when it writes every even byte, only the lower 4 bits will be stored to the RAM. When using Color Palette RAM, the Register REG[03h] bit [1:0] has to be set to 11b. Color Palette RAM must be written by 128 Bytes (8-bits per Byte) sequence at a time, and it is not readable. The diagram Figure 12-2 shows the initialization flow.

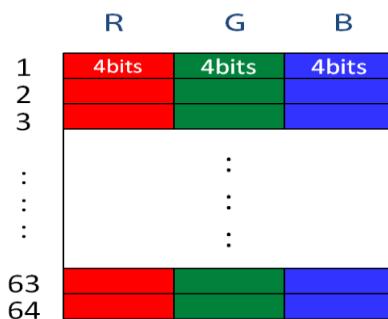


Figure 12-1: Color Palette RAM Organization

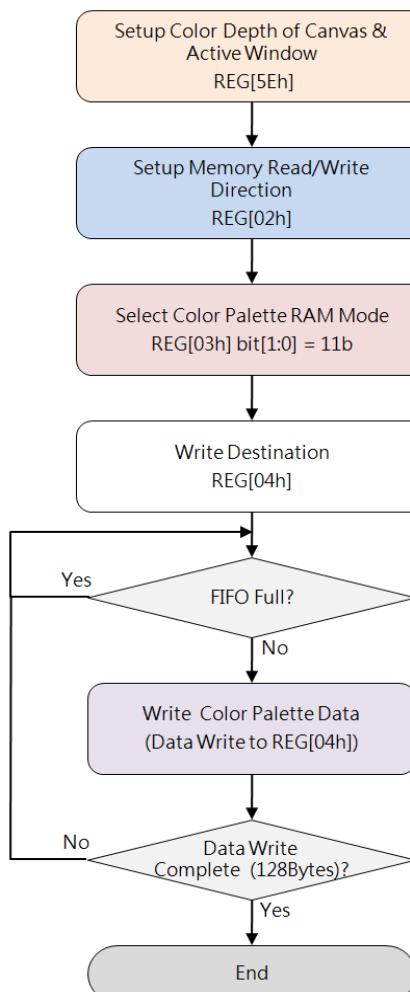


Figure 12-2: Flow Chart of Color Palette RAM Initialization

## 12.3 BTE Operation Overview

### 12.3.1 MCU Write with ROP

This operation supports 16 ROP functions. BTE engine will perform the ROP function automatically then write the result data to DT.

### 12.3.2 Memory Copy with ROP

This operation supports 16 ROP functions, but supports data transfer in positive direction only.

### 12.3.3 Solid Fill

This operation fills a specified rectangle area of DT with a solid color data defined in the Foreground Color Register.

### 12.3.4 Pattern Fill

This operation fills DT with an 8\*8 or 16\*16 pixel pattern.

### 12.3.5 Pattern Fill with Chroma Key

This operation fills DT with an 8\*8 or 16\*16 pixel pattern, combined with Chroma Key function but ROP function. If the pattern color is equal to the Chroma Key color which is defined in Background Color Register, then the destination BTE Window will not be changed.

### 12.3.6 MCU Write with Chroma Key

This operation supports data transfer from S0 (MCU Write) to DT. If S0 Color is equal to the Chroma Key Color which is defined in Background Color Register, then the destination BTE Window will not be changed. No ROP is applied for this operation.

### 12.3.7 Memory Copy with Chroma Key

This operation supports data transfer in positive direction only, and requires all source data and destination data located in Memory. If S0 Color is equal to Chroma Key color which is defined in Background Color Register, then the destination BTE Window will not be changed. No ROP is applied for this operation.

### 12.3.8 MCU Write with Color Expansion

This operation performs Color Expansion for the S0 (the data from MCU), from Monochrome 1 bpp format to Color 8/16/24 bpp format. The source data\_1b will expand to the color data defined in the Foreground Color Register. The source data\_0b will expand to the color data defined in the Background Color Register. If background transparency is enabled, then the color of destination BTE Window will remain no change.

**Note:** No matter background transparency is enabled or not, the color data set in Foreground Color Register (D2h~D4h) must be different from the color data set in Background Color Register (D5h~D7h).

### **12.3.9 Memory Copy with Color Expansion**

This operation performs Color Expansion for Source 0 data from Memory, please refer to 12.3.8 for other description and note.

### **12.3.10 Memory Copy with Opacity**

This operation performs Alpha Blending for S0 data and S1 data and transfers the result data to the destination BTE Window. It requires that S0, S1, and DT are located in Memory. The Alpha Blending has 2 modes, Picture Mode: for all pixels, blending by the same Alpha level. Pixel Mode: for each pixel, blending by individual Alpha Level of each pixel.

### **12.3.11 MCU Write with Opacity**

This operation performs Alpha Blending for S0 data and S1 data and transfers the result data to destination BTE Window. It requires S0 coming from MCU and S1/DT from Memory. Please refer to 12.3.10 for the descriptions of 2 modes.

## 12.4 BTE Memory Access Method

BTE accesses data of Sources and Destination by Block Method, and the size of the Data Block is defined by BTE Window. Below diagram shows how users can define S0 / S1 / DT / BTE\_Window, and shows the directions of Memory Access:

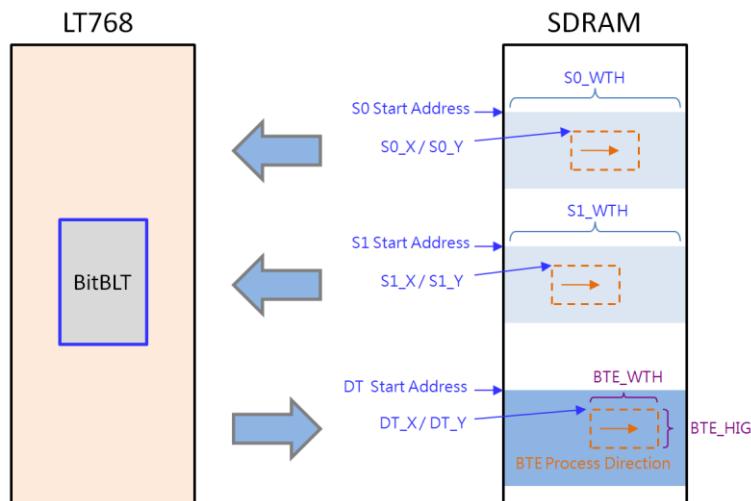


Figure 12-3: BTE Memory Access Method

## 12.5 BTE Chroma Key (Transparency Color) Function

If Chroma Key function is enabled, BTE will take Background Color (defined in Background Color Register) as the Chroma Key (Transparent Color), and compare the Chroma Key Data against the S0 Data one by one, if the compared result is equal, then BTE will not overwrite DT, otherwise write S0 Data to DT.

The description below shows the available bits of the Background Color Registers against different Color Depth:

### ■ Source Color Depth = 256

- Compare S0 Red color vs. REG[D5h] bit [7:5],
- Compare S0 Green color vs. REG [D6h] bit [7:5],
- Compare S0 Blue color vs. REG [D7h] bit [7:6]

### ■ Source Color Depth = 65,536

- Compare S0 Red color vs. REG[D5h] bit [7:3],
- Compare S0 Green color vs. REG [D6h] bit [7:2],
- Compare S0 Blue color vs. REG [D7h] bit [7:3]

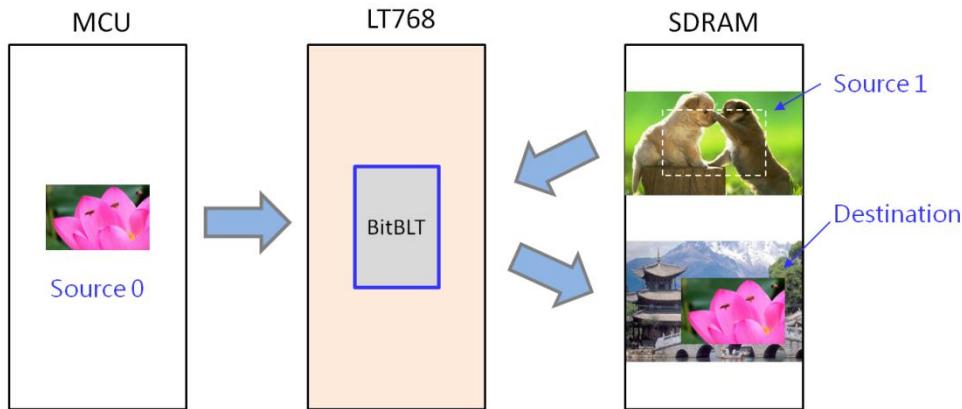
### ■ Source Color Depth = 16,777,216

- Compare S0 Red color vs. REG[D5h] bit [7:0],
- Compare S0 Green color vs. REG [D6h] bit [7:0],
- Compare S0 Blue color vs. REG [D7h] bit [7:0]

## 12.6 BTE Operation Detail

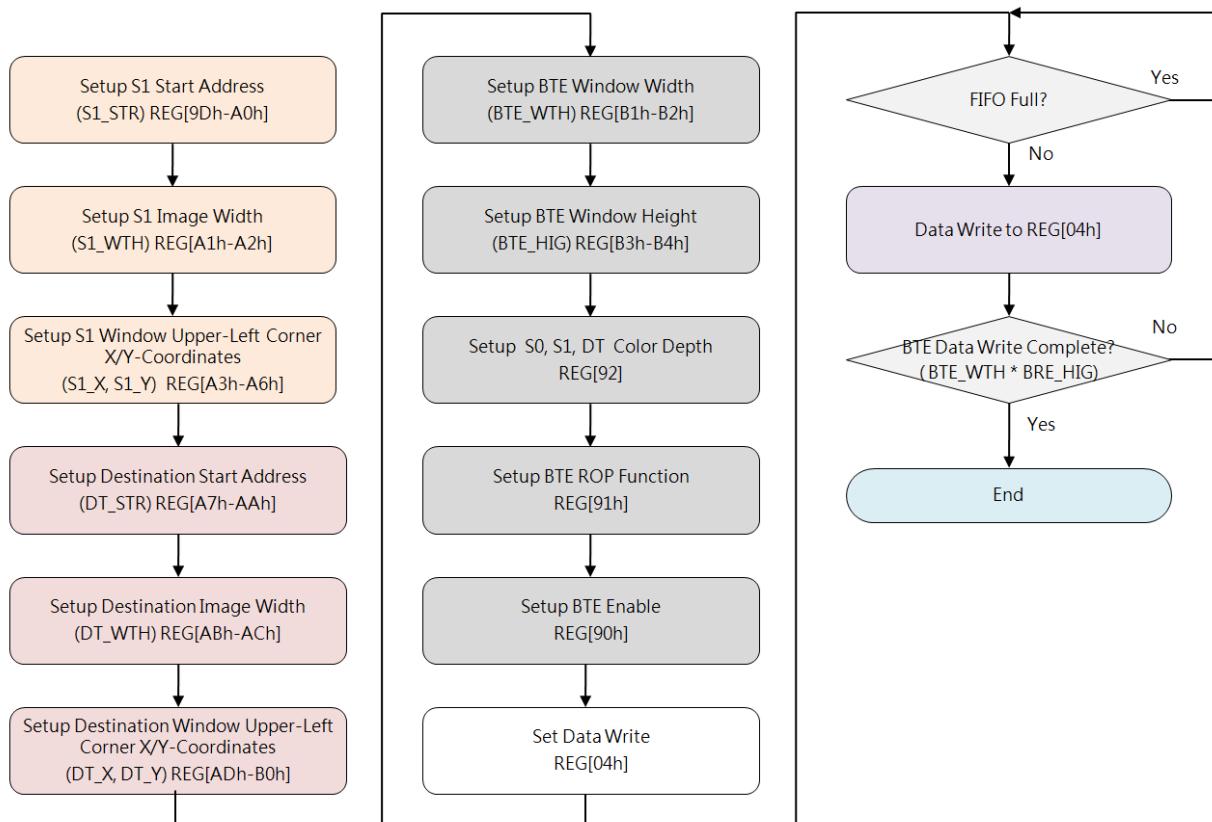
### 12.6.1 MCU Write with ROP

This operation is to transfer S0 (from MCU Write) to DT, and supports all 16 ROP functions. Below diagram is an example of the operation result while BTE Control Register REG[91h] is set as 0xC0h.



**Figure 12-4: Example of MCU Write with ROP**

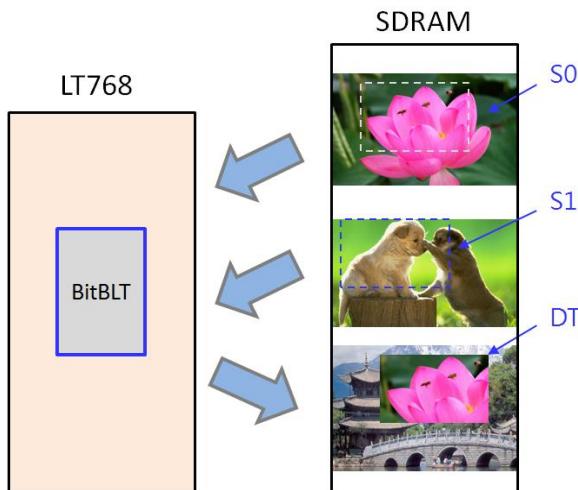
The suggested programming steps and register settings are listed below for reference.



**Figure 12-5: Flow Chart of MCU Write with ROP**

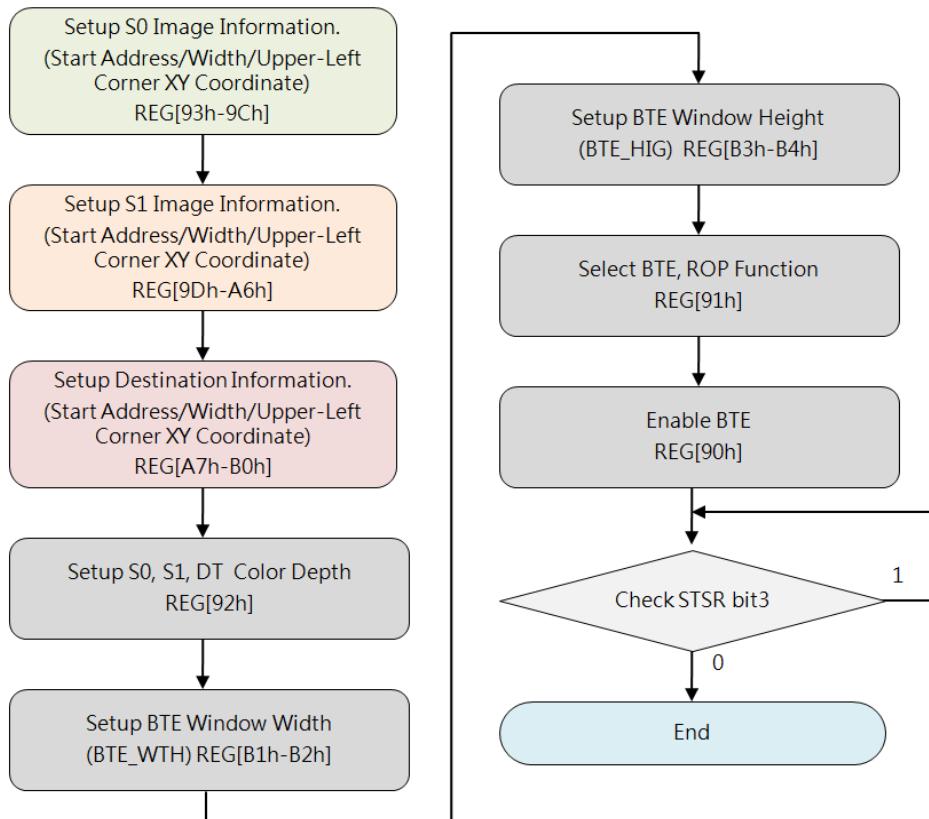
### 12.6.2 Memory Copy (move) with ROP

This operation performs Memory Copy from S0 (from Memory) to DT. Below diagram is an example of the operation result while BTE Control Register REG[91h] is set as 0xC2h.



**Figure 12-6: Example of Memory Copy with ROP**

There are two ways to get BTE status. Figure 12-7 shows one way to get it by checking the Status Register STSR bit3, and Figure 12-8 shows another by checking hardware interruption.



**Figure 12-7: Flow Chart 1 of Memory Copy with ROP**

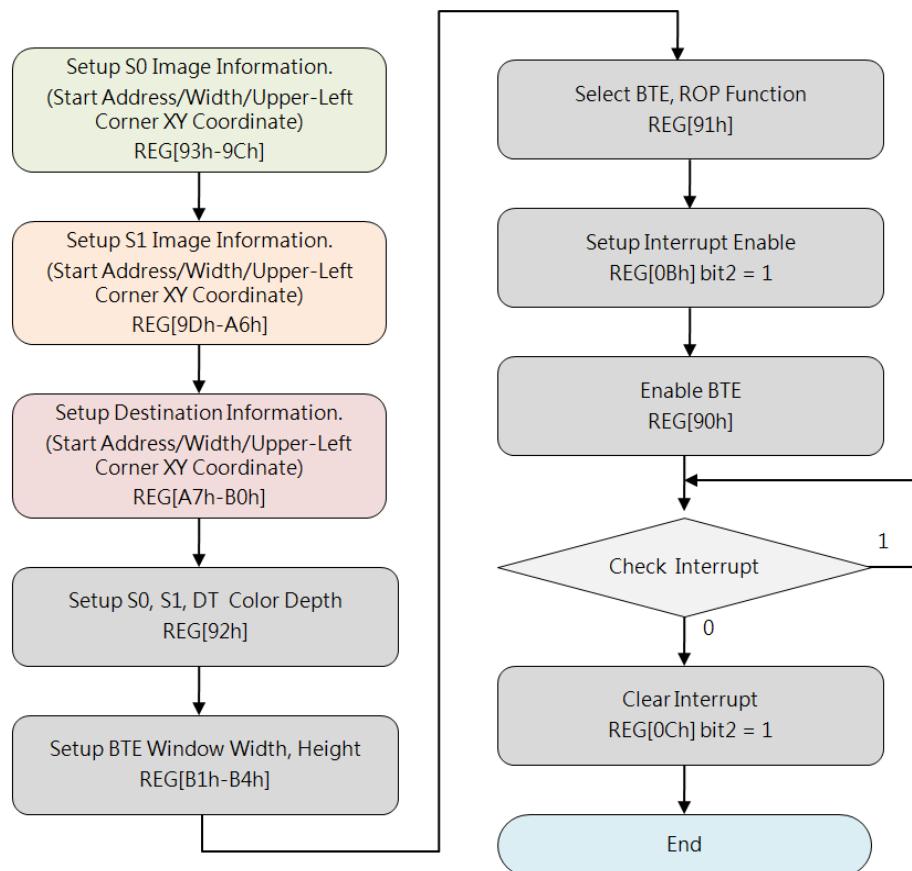
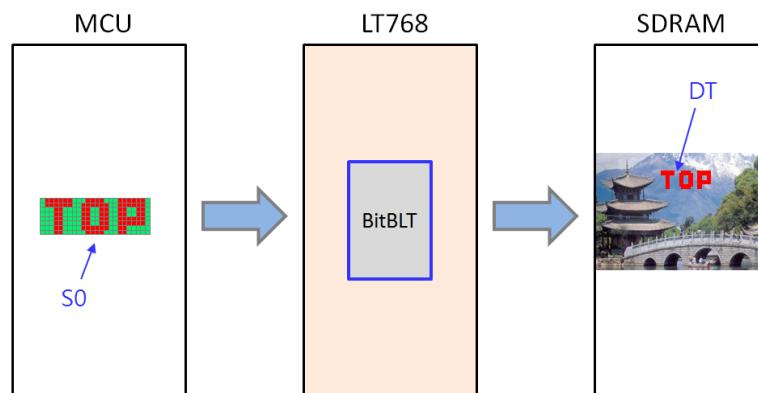


Figure 12-8: Flow Chart 2 of Memory Copy with ROP

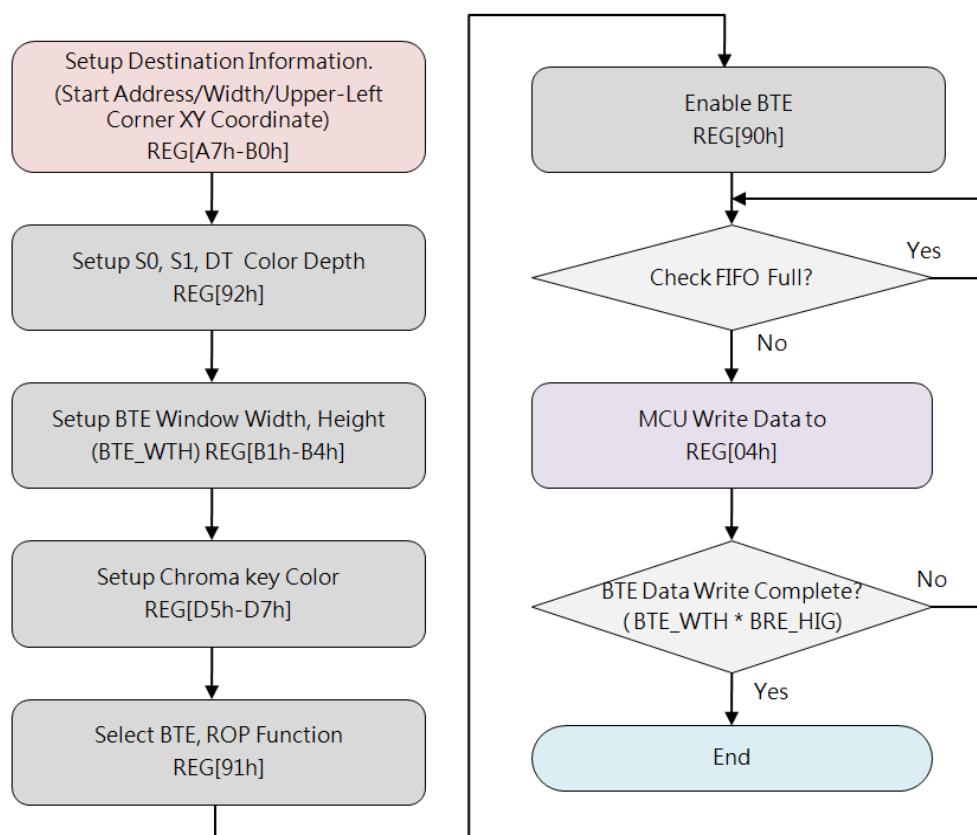
### 12.6.3 MCU Write with Chroma Key (w/o ROP)

This operation is to transfer S0 (from MCU Write) to DT, and supports Chroma Key function (referring to 12.5).

If S0 Color Data is equal to the Chroma Key (the color Data defined in the Background Color Registers REG[D5h-D7h] ), BTE will take that color as Transparent, which means BTE will not write that color data to DT. Below example shows GREEN is the background color of RED “TOP”, and GREEN is set as Chroma Key, therefore BTE will write RED “TOP” only to DT:



**Figure 12-9: Example of MCU Write with Chroma Key**

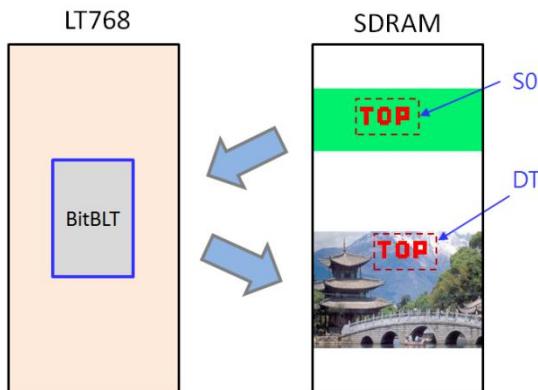


**Figure 12-10: Flow Chart of MCU Write with Chroma Key**

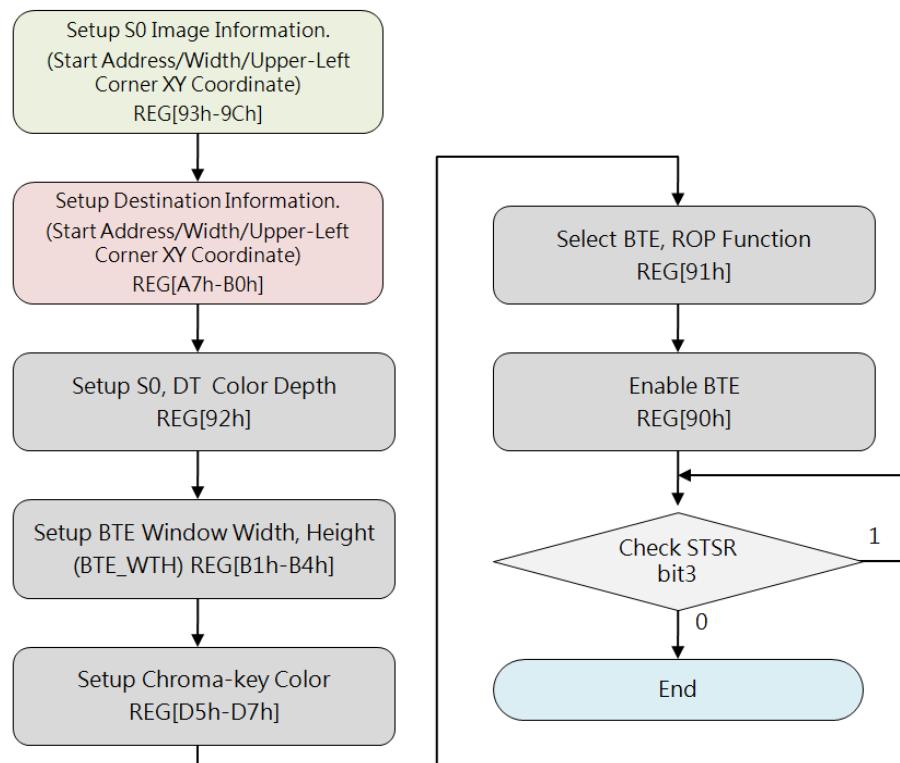
#### 12.6.4 Memory Copy with Chroma Key (w/o ROP)

This operation performs Memory Copy from S0 (from Memory) to DT, and supports Chroma Key function (referring to Section 12.5).

The difference comparing with “MCU Write with Chroma Key” is that S0 comes from Memory not from MCU, please refer to Section 12.6.3 for more descriptions and Figure 12-11 for an example:



**Figure 12-11: Example of Memory Copy with Chroma Key**

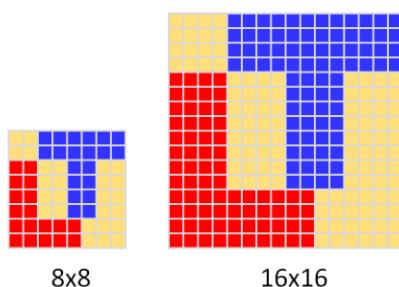


**Figure 12-12: Flow Chart of Memory Copy with Chroma Key**

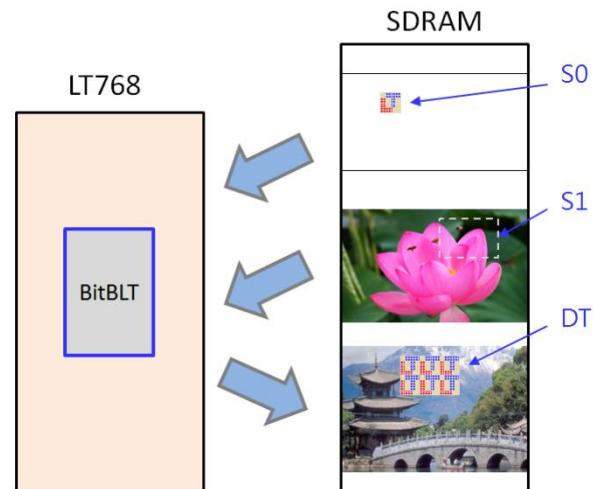
### 12.6.5 Pattern Fill with ROP

This operation is to fill a pattern (as S0) repeatedly to a specified Rectangular Area of Destination (also known as BTE Window). The pattern should be an array of 8x8 or 16x16 pixels stored in Memory. The pattern can be logically processed with S1 by using one of the 16 ROP functions. This operation can be used to speed up duplicating a matrix of pattern into an area, such as background paste function.

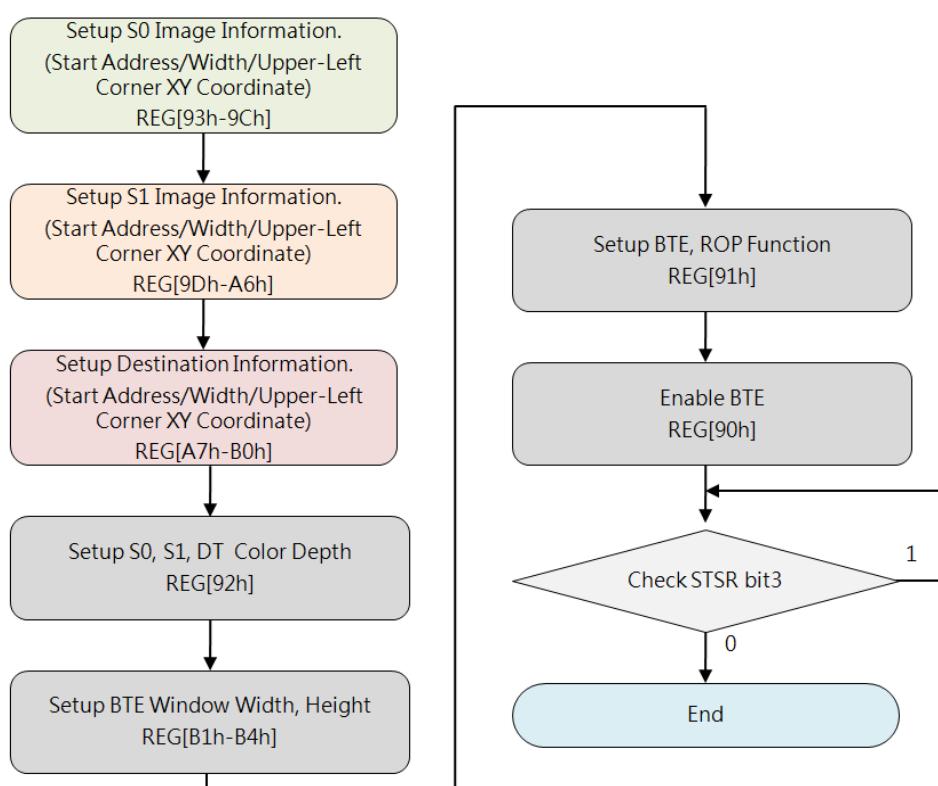
Below example shows a six fills with 8x8 Pattern, ROP REG[91h] bit[7:4] is set as 0xCh:



**Figure 12-13: Pattern Format**



**Figure 12-14: Example of Pattern Fill with ROP**



**Figure 12-15: Flow Chart of Pattern Fill with ROP**

### 12.6.6 Pattern Fill with Chroma Key

This operation is to fill a pattern (as S0) repeatedly to a specified Rectangular Area of DT (also known as BTE Window), and supports Chroma Key function (referring to Section 12.5). If any color data of the pattern is equal to the Chroma Key data, BTE will not write that part of data to DT.

Below example shows ORANGE is the background color of RED "T", and ORANGE is set as the Chroma Key. Therefore, BTE will fill RED "T" only to DT one by one:

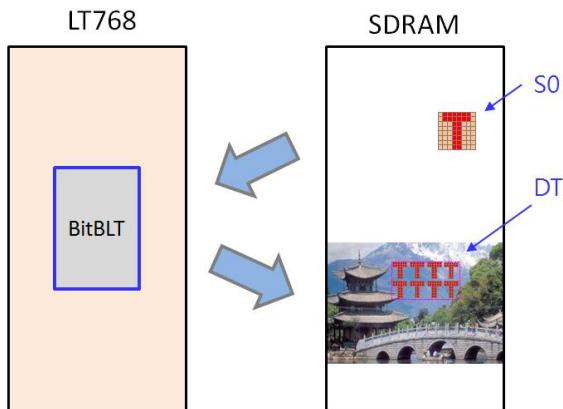


Figure 12-16: Example of Pattern Fill Chroma Key

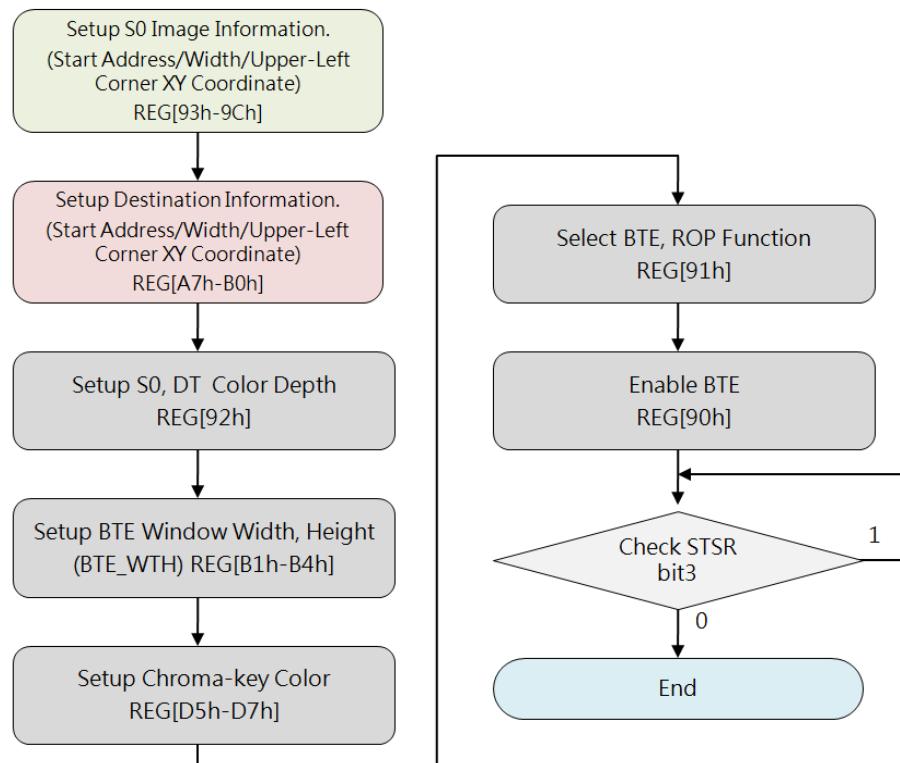


Figure 12-17: Flow Chart of Pattern Fill with Chroma Key

### 12.6.7 MCU Write with Color Expansion

This operation performs Color Expansion for S0 (from MCU Write). It is useful to translate bit-wise monochrome data to byte-wise color data. In this operation, S0 color depth REG[92h] bit[6:5] is ignored, BTE takes MCU Interface Width as S0 color depth (Word Width), that is, if MCU Interface is 8 bits, then the color depth is 8-bits, and if MCU Interface is 16 bits, then the color depth is 16-bits. Users should set needed start bit to REG[91h] bit[7:4] against the corresponding color depth, so bit[7] ~ bit[0] are available for 8-bits depth and bit[15] ~ bit[0] are available for 16-bits depth. BTE disassembles Word by Word to Bit sequences (from LSB to MSB) against the ROW Line of source image (from left to right), and sequentially expands bit by bit until the BTE Width reaches ending. The result to DT is that data\_1b expanded to Foreground Color and data\_0b expanded to Background Color. Any bit before the Start Bit and other Bits uncovered by the BTE Window will be discarded.

Below example shows expanding data\_1b to RED (Foreground Color) and data\_0b to BLUE (Background Color), so the result to DT is RED “TOP” together with BLUE background:

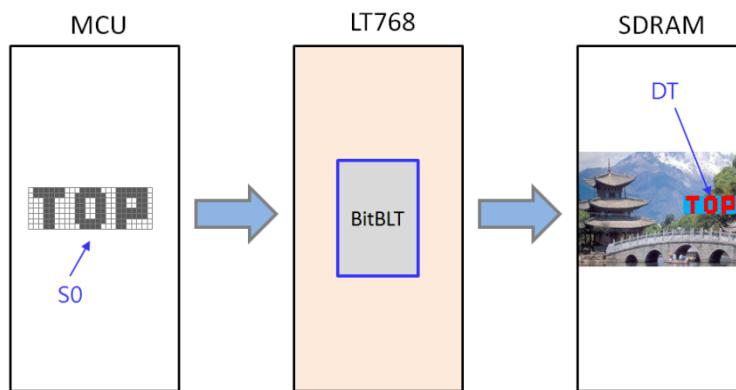
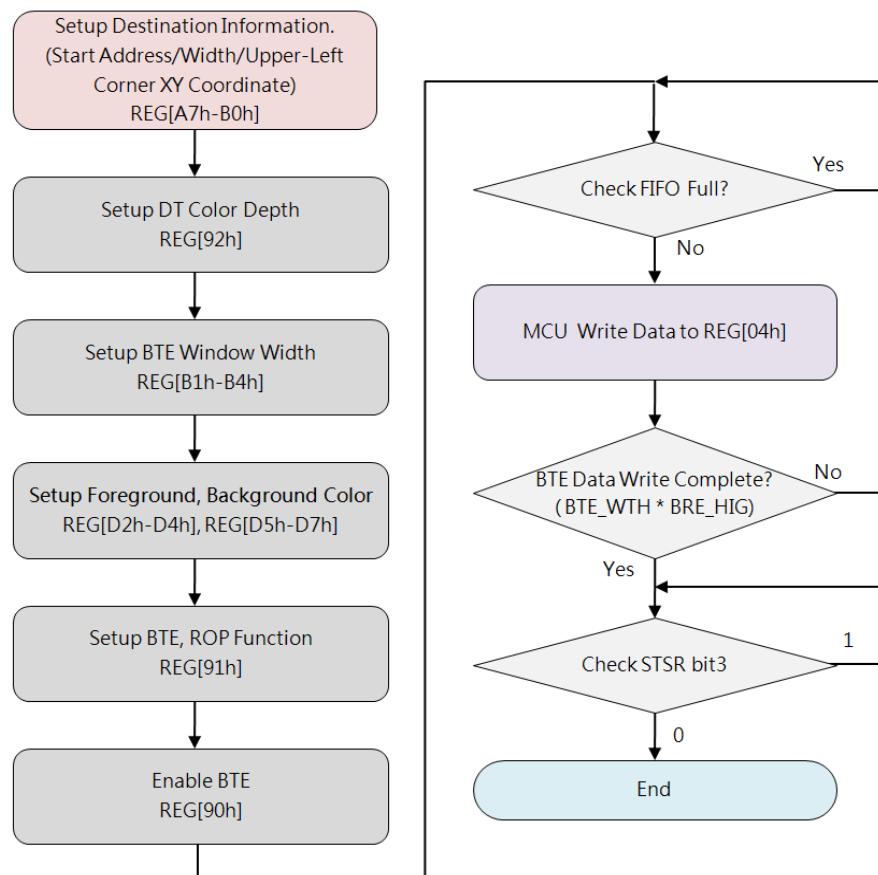
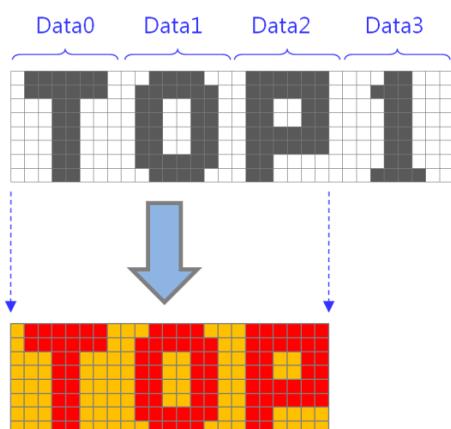


Figure 12-18: Example of MCU Write with Color Expansion

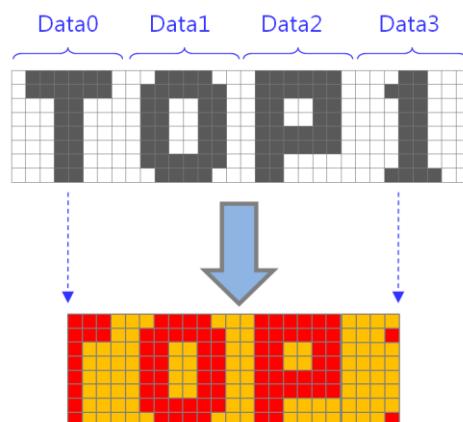


**Figure 12-19: Flow Chart of MCU Write with Color Expansion**

For 8-bits MCU Interface, if ROP = 7, Foreground Color is set to RED, Background Color is set to KHAKI, and BTE Width is set to 23, then the color expansion result is as shown in Figure 12-20. If ROP = 3, then the result is shown as Figure 12-21.



**Figure 12-20: Example 1 of MCU Write with Color Expansion (ROP=7)**

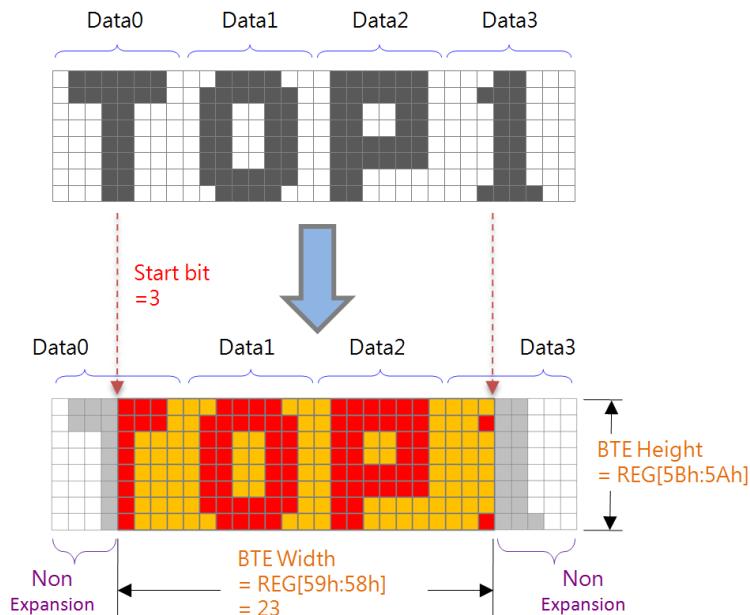


**Figure 12-21: Example 2 of MCU Write with Color Expansion (ROP=3)**

**Note:**

1. Sent Word\_ Numbers\_per\_Row  

$$= [\text{BTE\_Width} + (\text{MCU\_Interface\_bits} - \text{Start\_bit} - 1)] / (\text{MCU\_ Interface\_bits})$$
  
 (Take integer with unconditional carry)
2. Word\_Number\_Total = (Word\_ Numbers\_per\_Row) \* BTE Height

**Figure 12-22: Data Format for Color Expansion**

Example 1: "BTE Width"= 50, "MCU I/F bits" = 8 bits,

If Start bit=7, then:

$$\text{Sent Data Numbers per Row} = [\{ 50 + (8 - 7 - 1) \} / 8] = 7 \text{ Bytes}$$

If Start bit=4, then:

$$\text{Sent Data Numbers per Row} = [\{ 50 + (8 - 4 - 1) \} / 8] = 7 \text{ Bytes}$$

Example 2: "BTE Width"= 50, "MCU I/F bits" = 16 bits,

If Start bit =15, then:

$$\text{Sent Data Numbers per Row} = [\{ 50 + (16 - 15 - 1) \} / 16] = 4 \text{ Bytes}$$

If Start bit=0, then:

$$\text{Sent Data Numbers per Row} = [\{ 50 + (16 - 0 - 1) \} / 16] = 5 \text{ Bytes}$$

### 12.6.8 MCU Write with Color Expansion and Chroma key

This operation is similar to MCU Write with Color Expansion, but the difference is that data\_0b will be discarded, BTE expands only data\_1b to Foreground Color.

Below example shows that data\_1b is expanded to RED (Foreground Color) and data\_0b is discarded, so the result to DT is RED “TOP” with TRANSPARENT background:

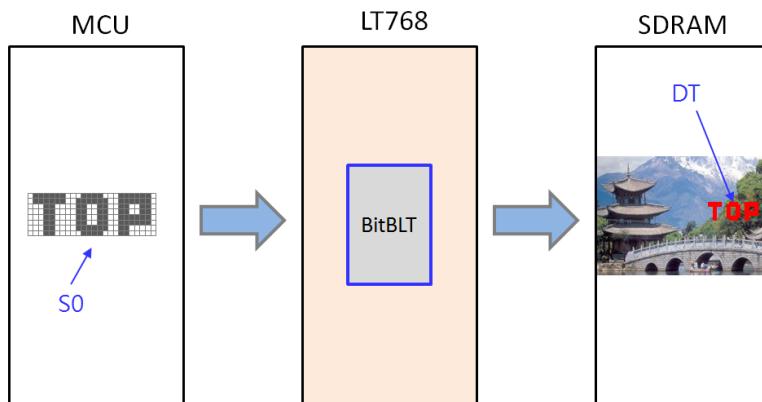


Figure 12-23: Example of MCU Write with Color Expansion and Chroma key

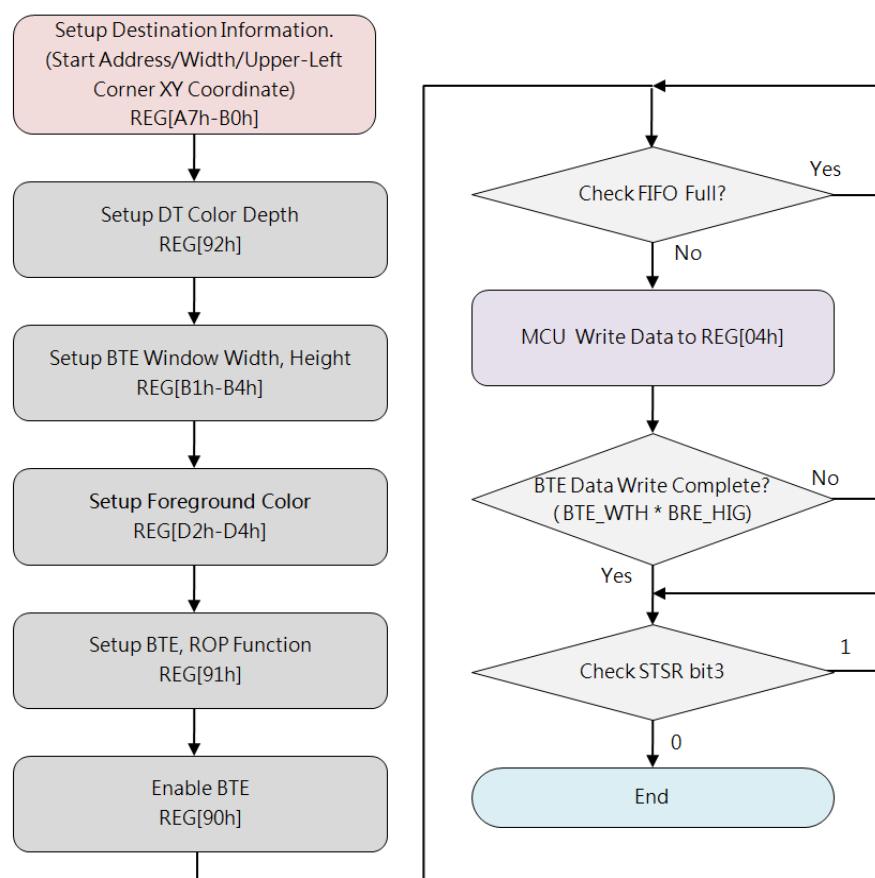


Figure 12-24: Flow Chart of MCU Write with Color Expansion and Chroma key

### 12.6.9 Memory Copy with Opacity

This operation performs blending S0 and S1 data and transferring the result to DT. Two modes are available: Picture Mode and Pixel Mode.

Picture Mode is for 8bpp/16bpp/24bpp format and it uses the same Opacity Value (Alpha Level) for the whole bitmap picture. Opacity Value of Picture Mode is defined in REG[B5h].

Pixel Mode is for 8bpp/16bpp format and it uses individual Opacity Value of S1, each pixel of S1 has its own Opacity Value, such as: for each 16bpp data, the bit[15:12] is Opacity Value, and the bit[11:0] is color data; for each 8bpp data of S1, the bit[7:6] is Opacity Value, and the bit[5:0] is the Index (Address) of Palette Color RAM pointing to initialized 12-bits Color Depth data.

- **Picture Mode:**

Alpha\_Level = REG[B5h]

DT Data = ( S0 \* Alpha\_Level ) + ( S1 \* ( 1- Alpha\_Level ) )

- **Pixel Mode 8bpp:**

Alpha\_Level = S1\_Bit[7:6]

DT Data = ( S0 \* Alpha\_Level ) + ( Palette\_Color\_RAM[ S1\_Bit[5:0] ] \* ( 1 – Alpha\_Level ) )

- **Pixel Mode 16bpp:**

Alpha\_Level = S1\_Bit[15:12]

DT Data = ( S0 \* Alpha\_Level ) + ( S1\_Bit[11:0] \* ( 1 – Alpha\_Level ) )

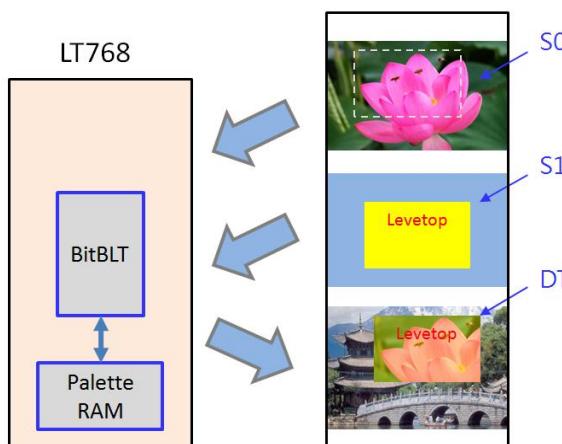


Figure 12-25: Example of Pixel Mode - 8bpp

Table 12-4: Alpha Blending of Pixel Mode - 8bpp

Bit[7:6]	Alpha Level
0h	0
1h	10/32
2h	21/32
3h	1

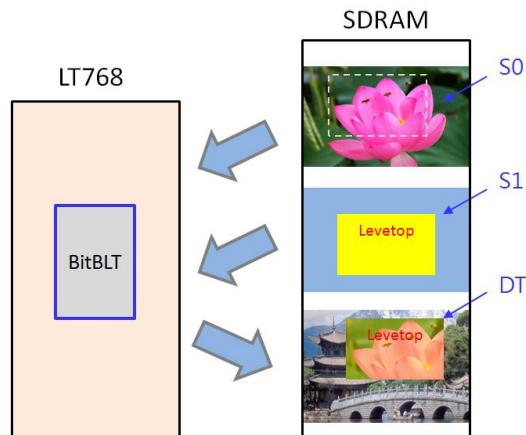


Figure 12-26: Example of Pixel Mode - 16bpp

Table 12-5: Alpha Level of Pixel Mode - 16bpp

Bit[15:12]	Alpha Level
0h	0
1h	2/32
2h	4/32
3h	6/32
4h	8/32
5h	10/32
6h	12/32
7h	14/32
8h	16/32
9h	18/32
Ah	20/32
Bh	22/32
Ch	24/32
Dh	26/32
Eh	28/32
Fh	1

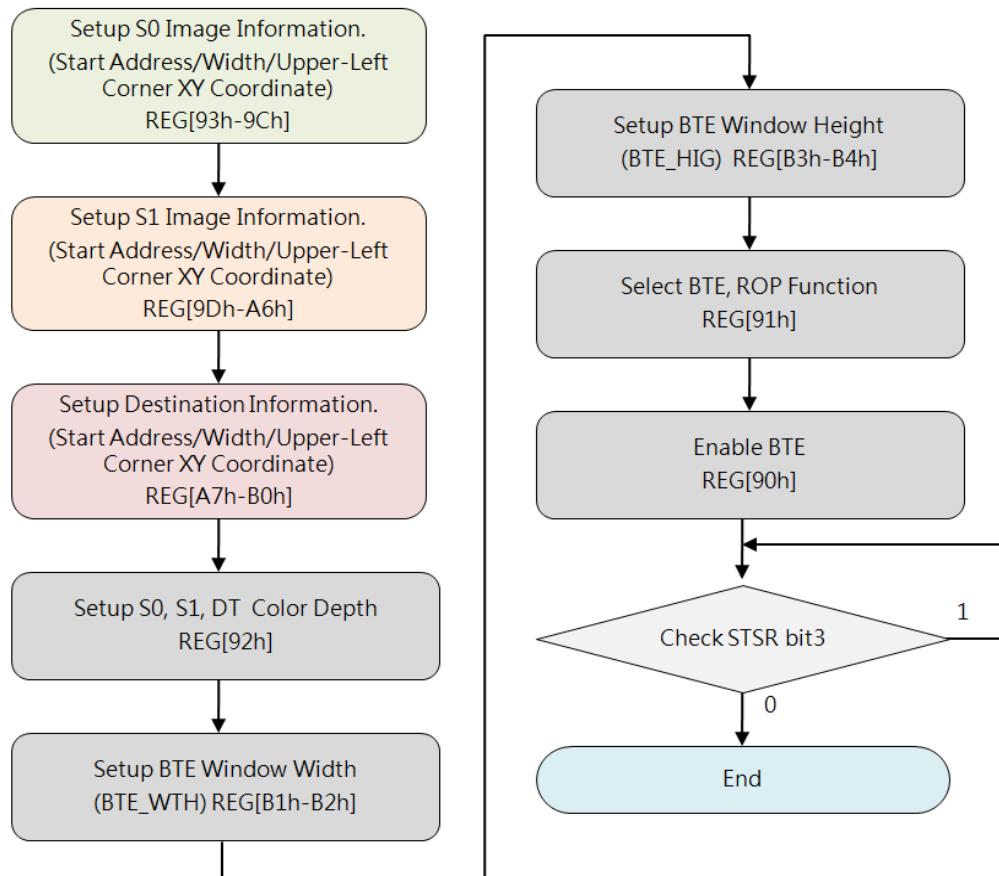


Figure 12-27: Flow Chart of Memory Copy with Opacity - Pixel Mode

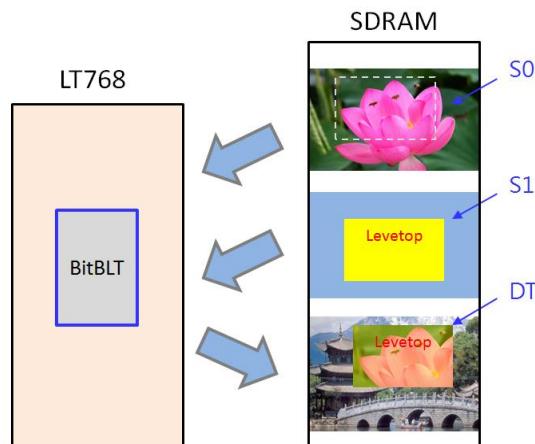


Figure 12-28: Example of Memory Copy with Opacity - Picture Mode

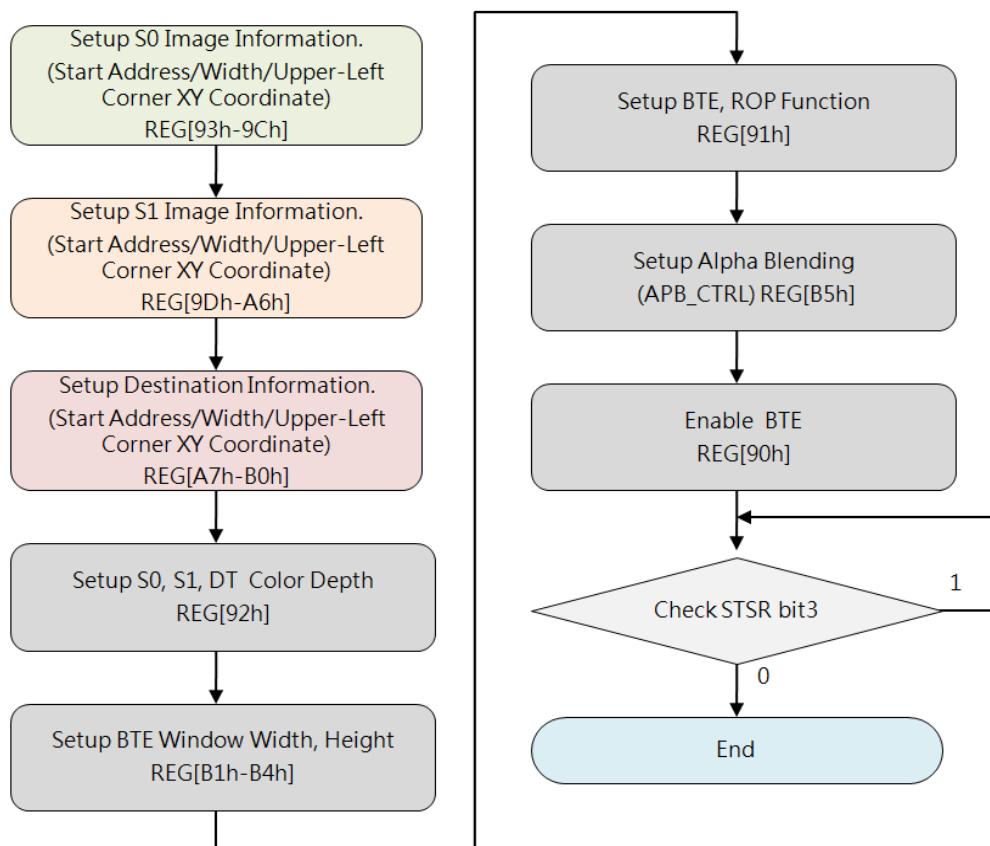
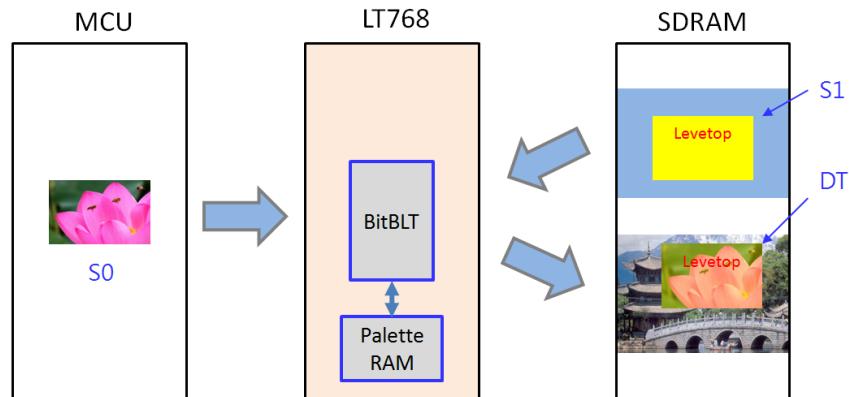


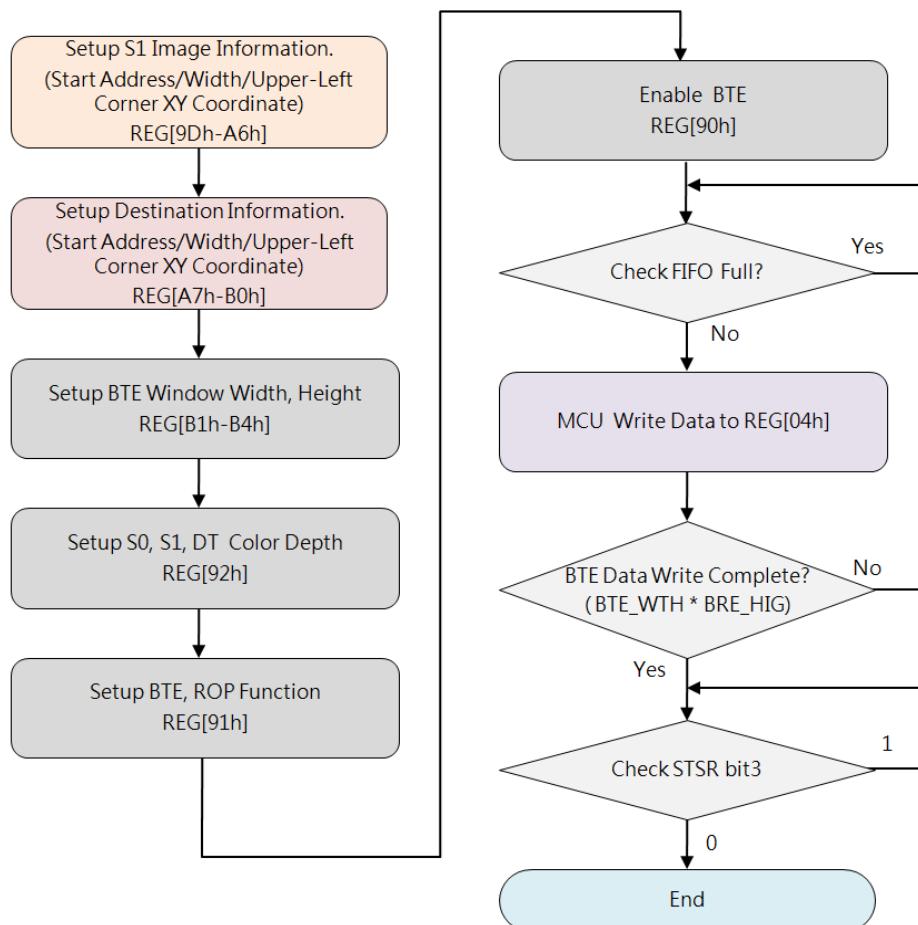
Figure 12-29: Flow Chart of Memory Copy with Opacity - Picture Mode

### 12.6.10 MCU Write with Opacity

This operation is similar to Memory Copy with Opacity, but the difference is that S0 is from MCU Write, and it also has Picture Mode and Pixel Mode, for more descriptions please refer to Section 12.6.9.



**Figure 12-30: Example of MCU Write with Opacity**



**Figure 12-31: Flow Chart of MCU Write with Opacity**

### 12.6.11 Memory Copy with Color Expansion

This operation performs Color Expansion for the S0 data (from Memory), it is useful to translate bit-wise monochrome data to byte-wise color data. In this operation, S0 data Color Depth (Word Width) is defined in REG[92h] bit[6:5]. Users should set needed Start Bit to REG[91h] bit[7:4] based on the corresponding color depth, bit[7] ~ bit[0] are available for 8-bits depth and bit[15] ~ bit[0] are available for 16-bits depth. BTE disassembles Word by Word to bit sequences (from MSB to LSB) based on the ROW Line of source image (from left to right), and sequentially expands bit by bit until the BTE Width reaches ending as well. The result to DT is that data\_1b is expanded to Foreground Color and data\_0b is expanded to Background Color. Any bit before the Start Bit and other Bits uncovered by the BTE Window will be discarded.

Below example shows expanding data\_1b to RED (Foreground Color) and data\_0b to BLUE (Background Color), so the result to DT is RED “TOP” together with BLUE background:

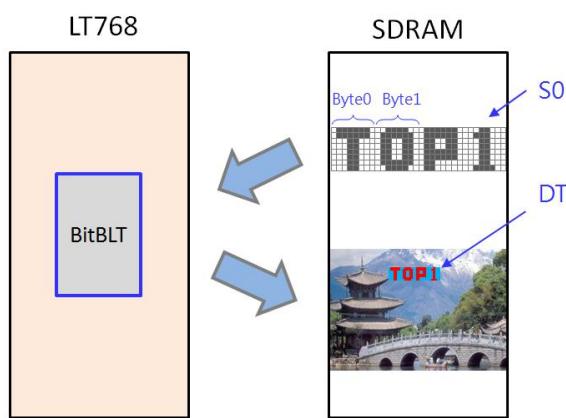


Figure 12-32: Example of Memory Copy with Color Expansion

While: S0 Color Depth = 8, Foreground Color = RED, Background Color = KHAKI, BTE window Width = 23, please refer to Figure 12-33 (ROP = 7) and Figure 12-34 (ROP = 3) as examples.

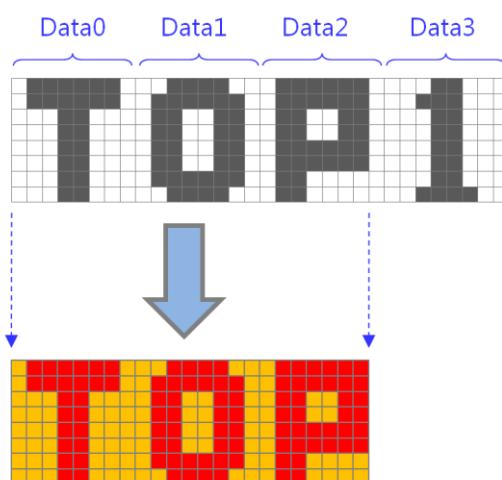


Figure 12-33: Example 1 of Memory Copy with Color Expansion (ROP=7)

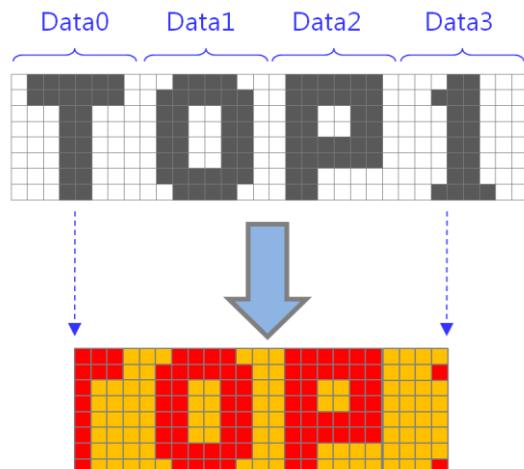


Figure 12-34: Example 2 of Memory Copy with Color Expansion (ROP=3)

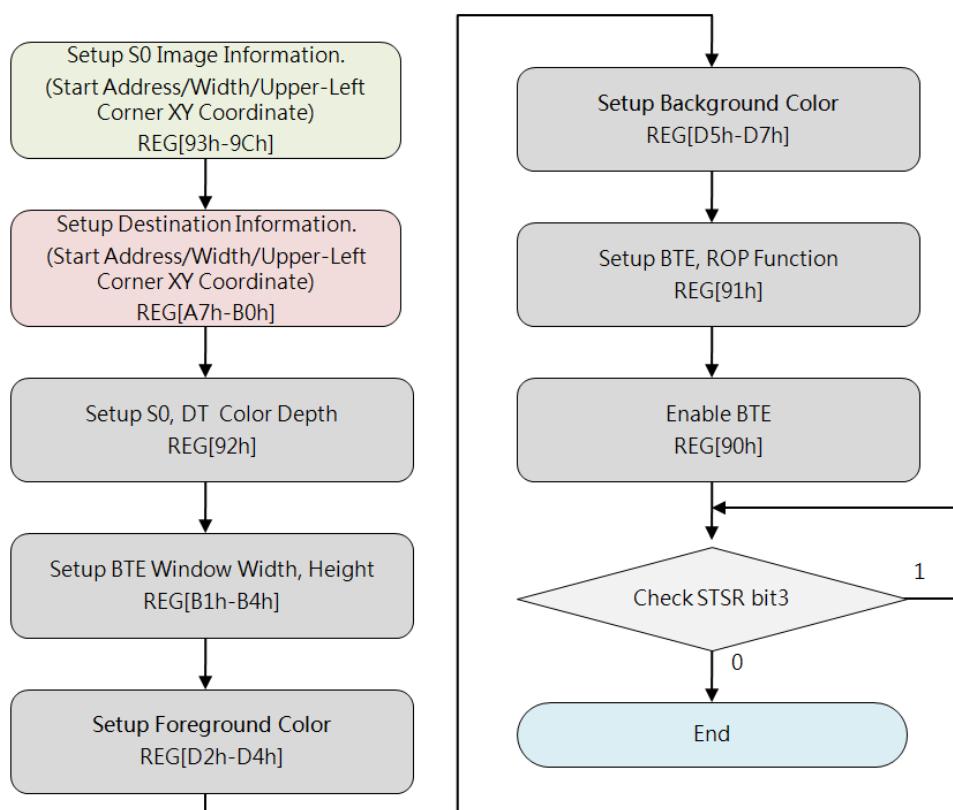
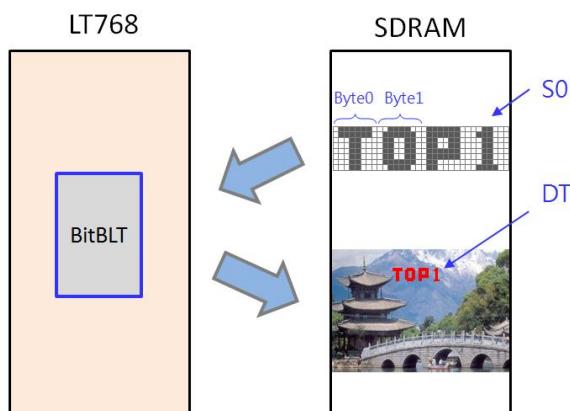


Figure 12-35: Flow Chart of Memory Copy with Color Expansion

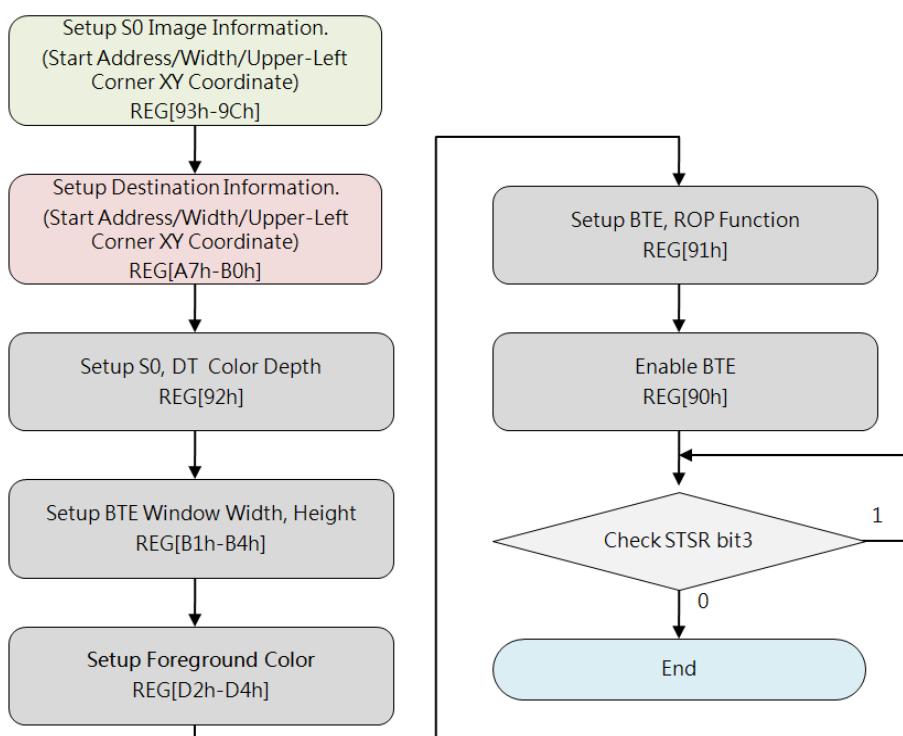
### 12.6.12 Memory Copy with Color Expansion and Chroma Key

This operation is similar to Memory Copy with Color Expansion, but the difference is that data\_0b will be discarded, BTE expands only data\_1b to Foreground Color.

Below example shows that data\_1b is expanded to RED (Foreground Color), and data\_0b is discarded, so the result to DT is RED "TOP1" with TRANSPARENT background:



**Figure 12-36: Example of Memory Copy with Color Expansion and Chroma Key**



**Figure 12-37: Flow Chart of Memory Copy with Color Expansion and Chroma Key**

### 12.6.13 Solid Fill

This operation is to fill a specified Rectangle Area (BTE Window) of DT with a Solid Color whose data is defined in the Foreground Color Register.

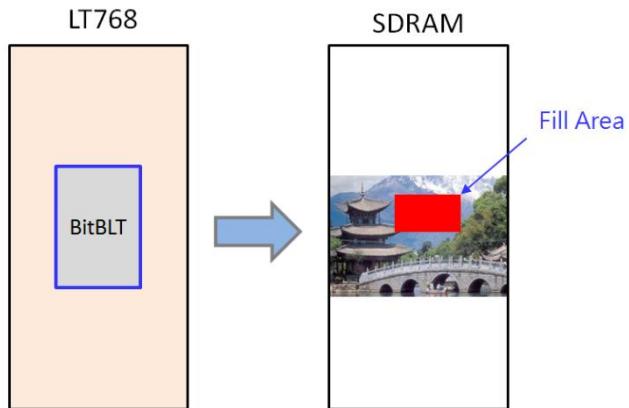


Figure 12-38: Example of Solid Fill

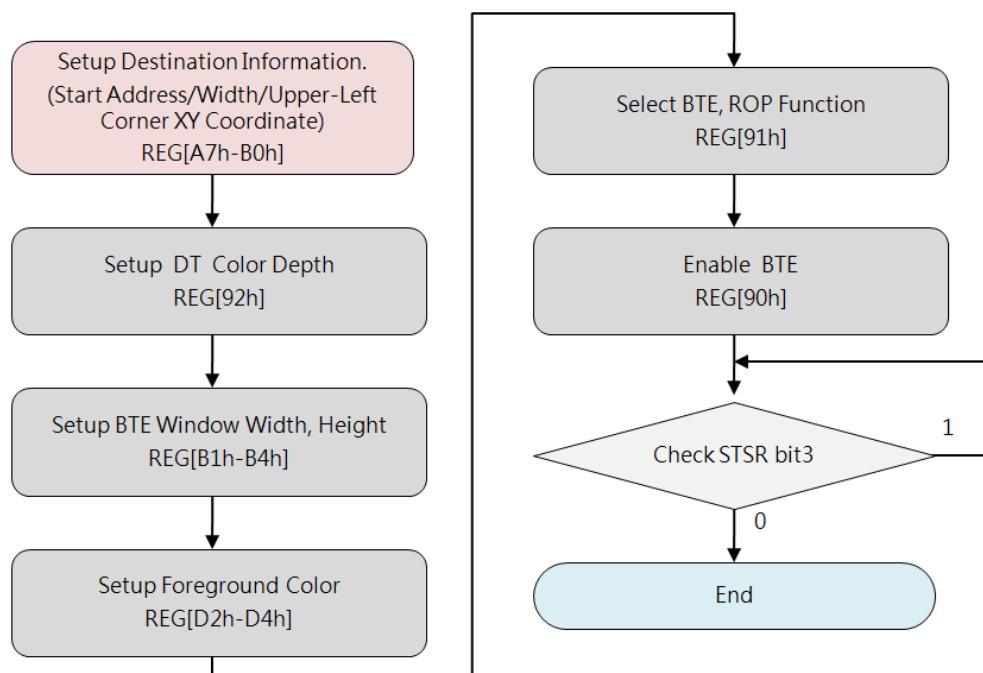


Figure 12-39: Flow Chart of Solid Fill

## 13. Display Text

LT768x supports 2 sources of Text (Character and Symbols):

- CGROM(Character Graphic ROM): Embedded ISO/IEC 8859 Character Sets
- CGRAM(Character Graphic RAM): User-defined Character Sets (UCG sets)

LT768x internal CGROM supports four sets of embedded Characters and Symbols of ASCII code, and internal CGRAM supports users to create their own Characters and Symbols sets when needed. The registers REG[CCh] ~REG[DEh] are for the purpose of User-defined Characters. The Foreground Color registers REG[D2h] ~REG[D4h] and the Background Color registers REG[D5h] ~ REG[D7h] are also employable to define Color for characters from any source.

### 13.1 Internal CGROM

LT768x has three built-in ASCII font types with different resolutions (character sizes): 8 x 16, 12 x 24, 16 x 32. The MCU can simply writes the font code to display the ASCII code on the LCD panel. The size of the displayed ASCII characters is set by REG[CCh] [5:4]. The ASCII code is corresponding to the standard ISO/IEC 8859-1/2/4/5 coding (Table 13-1 to Table 13-4 below). The type of ISO/IEC 8859 font displayed on the TFT panel is set by REG [CCh] [1:0]. In addition, users can select the color of the text by setting the front-view register REG (D2h to D4h) and the background color register (REG (D5h~D7h)). Please refer to the following program flowchart:

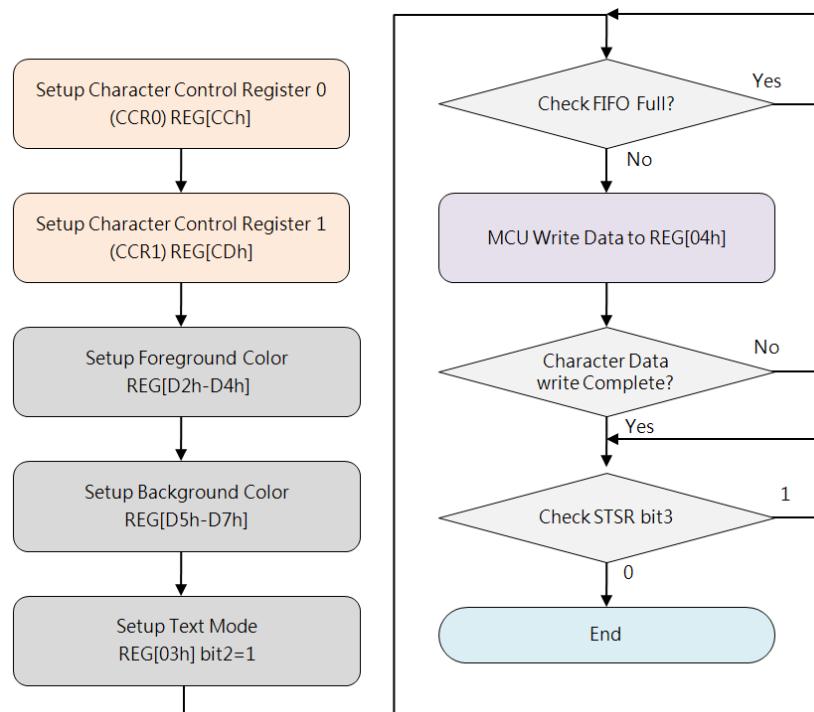


Figure 13-1: Flow Chart of CGROM Basic Programming

Table 13-1 shows the standard character encoding of ISO/IEC 8859-1. ISO stands for International Standardization Organization. The ISO/IEC 8859-1, generally known as "Latin-1". It is the first sets of 8-bit coded character encoding developed by the ISO, and referred to ASCII that consists of 192 characters from the Latin script in the range of 0xA0-0xFF. This character coding is used throughout Western Europe, including Albanian, Afrikaans, Breton, Danish, Faroese, Frisian, Galician, German, Greenlandic, Icelandic, Irish, Italian, Latin, Luxembourgish, Norwegian, Portuguese, Rhaeto-Romanic, Scottish Gaelic, Spanish, Swedish. English letters

without accent marks are also available in ISO/IEC 8859-1. In addition, it is also commonly used in many languages outside Europe, such as Swahili, Indonesian, Malaysian and Tagalog.

In Table 13-1, character codes 0x80~0x9F are defined by Microsoft Windows, also called CP1252 (WinLatin1).

**Table 13-1: ISO/IEC 8859-1**

L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
H	☺	☻	♥	♦	♣	♠	•	+	○	○	♂	♀	♪	♫	◊	
0	▶	◀	↑	↓	!'	§	■	↑	↓	→	←	↶	↷	▲	▼	
1	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
2	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	
3	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
4	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	^	_	
5	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
6	p	q	r	s	t	u	v	w	x	y	z	{	}	~	□	
7	€	,	f	„	…	†	‡	‰	š	č	đ	ž	ÿ			
8	‘	’	“	”	•	-	-	™	š	č	đ	ž	ÿ			
9	‘	’	“	”	•	-	-	™	š	č	đ	ž	ÿ			
A	í	¢	£	¤	¥	₩	₪	₪	₪	₪	₪	₪	₪	₪	₪	
B	º	±	²	³	’	µ	¶	·	¹	º	»	¼	½	¾	¸	
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Ï	
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	ï	
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	

**Table 13-2: ISO/IEC 8859-2**

L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
H	☺	☻	♥	♦	♣	♠	•	+	○	○	♂	♀	♪	♫	◊	
0	▶	◀	↑	↓	!'	§	■	↑	↓	→	←	↶	↷	▲	▼	
1	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
2	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	
3	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
4	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	^	_	
5	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
6	p	q	r	s	t	u	v	w	x	y	z	{	}	~	□	
7	€	,	f	„	…	†	‡	‰	š	č	đ	ž	ÿ			
8																
9																
A	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Ï	
B	Ã	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Ï	
C	Ŕ	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Ï	
D	Đ	Ń	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	
E	ŕ	á	â	ã	ä	å	æ	ç	è	é	ê	ë	í	í	ï	
F	đ	ń	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	

Table 13-2 shows the standard characters of ISO/IEC 8859-2, also known as Latin-2, it is the second sets of the 8-bit character encoding developed by ISO. This code sets can be used in almost any data interchange system to communicate in the following European languages: Croatian, Czech, Hungarian, Polish, Slovak, Slovenian, and Upper Sorbian. The Serbian, English, German, Latin can use ISO/IEC 8859-2 as well. Furthermore it is suitable to represent some western European languages like Finnish (with the exception of å used in Swedish and Finnish)

Table 13-3 shows the standard characters of ISO/IEC 8859-4, also known as Latin-4 or “North European”, it is the fourth sets of the ISO/IEC 8859 8-bit character encoding. It was designed originally to cover Estonian, Greenlandic, Latvian, Lithuanian, and Sami. This character set also supports Danish, English, Finnish, German, Latin, Norwegian, Slovenian, and Swedish.

Table 13-3: ISO/IEC 8859-4

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	*
1	►	◀	↑	↓	!&	§	■	↑	↓	→	←	Γ	⊖	▲	▼	
2	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	^	_	
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	}	~	ø	
8																
9																
A	À	Á	Â	Ã	҆	҇	҈	҉	Ҋ	ҋ	Ҍ	ҍ	Ҏ	ҏ	Ґ	ґ
B	à	á	â	ã	҆	҇	҈	҉	Ҋ	ҋ	Ҍ	ҍ	Ҏ	ҏ	ґ	Ғ
C	Ā	Ā	Ā	Ā	Ā	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
D	Đ	Ń	Ó	Ķ	Ô	Ö	×	Ø	Ӯ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ
E	ä	å	å	ä	ä	æ	í	ć	é	é	é	í	í	í	í	í
F	đ	ń	ó	ķ	ô	ö	÷	ø	ӱ	ӱ	ӱ	ӱ	ӱ	ӱ	ӱ	ӱ

Table 13-4: ISO/IEC 8859-5

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	*
1	►	◀	↑	↓	!&	§	■	↑	↓	→	←	Γ	⊖	▲	▼	
2	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	^	_	
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	}	~	ø	
8																
9																
A	Ё	Ђ	Ѓ	Ќ	І	І	І	І	І	І	І	І	І	І	І	І
B	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ
C	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ
D	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ
E	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ
F	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ	Ѡ

Table 13-4 shows the standard characters of ISO/IEC 8859-5, also known as the fifth sets of the ISO/IEC 8859 8-bit character encoding. It was designed originally to cover Bulgarian ,Belarusian, Russian, Serbian and Macedonian.

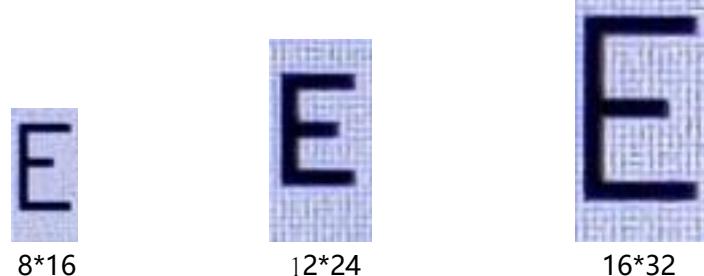


Figure 13-2: Internal ASCII Font for 8\*16, 12\*24 and 16\*32

### 13.2 User-defined Character Graphic (UCG)

Users can create and utilize UCG when needed. LT768x supports Half Width size (8x16, 12x24, 16x32 dot-matrix graphic) and Full Width size (16x16, 24x24, 32x32 dot-matrix graphic), and supports up to 32,768 UCGs with half width by encoding from 0000h up to 7FFFh, and up to 32,768 UCGs with full width by encoding from 8000h up to FFFFh.

To display a specific UCG, MCU just needs to write corresponding CODE of the UCG to LT768x, LT768x can resolve the CODE (relative ADDRESS of CGRAM) to correspond the absolute ADDRESS of Memory where the UCG data is really saved, then transfer the graphic data to display Memory Buffer. Users can define Foreground Color by setting the REG[D2h] ~REG[D4h] and Background Color by setting the REG[D5h]~REG[D7h] in advance.

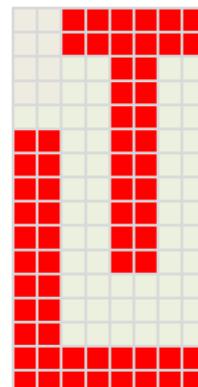
#### 13.2.1 8\*16 UCG Data Format

UCG with 8\*16 size needs 16 bytes data. For example, CGRAM start address is 1000h and the first UCG encoding is 0000h, the data will be saved in 1000h~100Fh, then the second UCG encoding will be 0001h and its data will be saved in 1010h~101Fh. Below formula shows how the 8\*16 UCG address in Memory is calculated, and Table 13-5 explains the data format and data byte sequence:

$$\text{UCG\_ADD} = \text{CGRAM\_Start\_ADD} + (\text{UCG\_Code} * 16)$$

Table 13-5: Data Format and Byte Sequence of 8\*16 UCG

UCG Code: 0000h		UCG Code: 0001h	
Address	Data	Address	Data
1000h	Byte0: 3Fh	1010h	Byte0
1001h	Byte1: 3Fh	1011h	Byte1
1002h	Byte2: 0Ch	1012h	Byte2
1003h	Byte3: 0Ch	1013h	Byte3
:	:	:	:
:	:	:	:
:	:	:	:
100Dh	Byte14: C0h	101Dh	Byte14
100Eh	Byte14: FFh	101Eh	Byte14
100Fh	Byte15: FFh	101Fh	Byte15



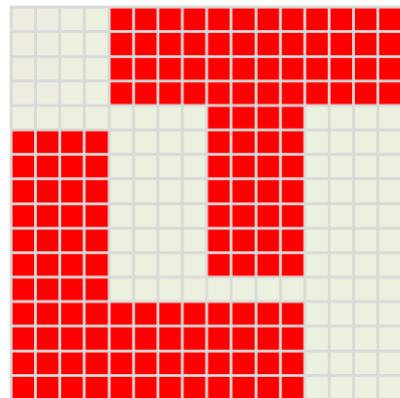
### 13.2.2 16\*16 UCG Data Format

UCG with 16\*16 size needs 32 bytes data. For example, CGRAM start address is 1000h and the first UCG encoding is 0000h, the data will be saved in 1000h~101Fh, then the second UCG encoding will be 0001h and its data will be saved in 1020h~103Fh. Below formula shows how the 16\*16 UCG address in Memory is calculated, and Table 13-6 explains the data format and data byte sequence:

$$\text{UCG\_ADD} = \text{CGRAM\_Start\_ADD} + (\text{UCG\_Code} * 32)$$

**Table 13-6: Data Format and Byte Sequenc of 16\*16 UCG**

UCG Code: 0000h			
Address	Data	Address	Data
1000h	Byte0: 0Fh	1001h	Byte1: FFh
1002h	Byte2: 0Fh	1003h	Byte3: FFh
1004h	Byte4: 0Fh	1005h	Byte5: FFh
1006h	Byte6: 0Fh	1007h	Byte7: FFh
:	:	:	:
:	:	:	:
:	:	:	:
101Ah	Byte26: FFh	101Bh	Byte27: F0h
101Ch	Byte28: FFh	101Dh	Byte29: F0h
101Eh	Byte30: FFh	101Fh	Byte31: F0h



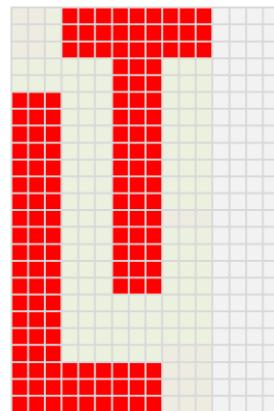
### 13.2.3 12\*24 UCG Data Format

UCG with 12\*24 size needs 48 bytes data, note that bit[3:0] of the byte with Odd Sequence Number is ignored. For example, CGRAM start address is 1000h and the first UCG encoding is 0000h, the data will be saved in 1000h~102Fh, then the second UCG encoding will be 0001h and its data will be saved in 1030h~105Fh. Below formula shows how the 12\*24 UCG address in Memory is calculated, and Table 13-7 explains the data format and data byte sequence:

$$\text{UCG\_ADD} = \text{CGRAM\_Start\_ADD} + (\text{UCG\_Code} * 48)$$

**Table 13-7: Data Format and Byte Sequence of 12\*24 UCG**

UCG Code: 0000h			
Address	Data	Address	Data
1000h	Byte0: 1Fh	1001h	Byte1: F0h
1002h	Byte2: 1Fh	1003h	Byte3: F0h
1004h	Byte4: 1Fh	1005h	Byte5: F0h
1006h	Byte6: 03h	1007h	Byte7: 80h
:	:	:	:
:	:	:	:
:	:	:	:
102Ah	Byte42: FFh	102Bh	Byte43: 80h
102Ch	Byte44: FFh	102Dh	Byte45: 80h
102Eh	Byte46: FFh	102Fh	Byte47: 80h



### 13.2.4 24\*24 UCG Data Format

UCG with 24\*24 size needs 72 bytes data. For example, CGRAM start address is 1000h and the first UCG encoding is 0000h, the data will be saved in 1000h~1047h, then the second UCG encoding will be 0001h and its data will be saved in 1048h~108Fh. Below formula shows how the 24\*24 UCG address in Memory is calculated, and Table 13-8 explains the data format and data byte sequence:

$$\text{UCG\_ADD} = \text{CGRAM\_Start\_ADD} + (\text{UCG\_Code} * 72)$$

Table 13-8: Data Format and Byte Sequence of 24\*24 UCG

UCG Code: 0000h					
Address	Data	Address	Data	Address	Data
1000h	Byte0	1001h	Byte1	1002h	Byte2
1003h	Byte3	1004h	Byte4	1005h	Byte5
1006h	Byte6	1007h	Byte7	1008h	Byte8
1009h	Byte9	100Ah	Byte10	100Bh	Byte11
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
103Fh	Byte63	1040h	Byte64	1041h	Byte65
1042h	Byte66	1043h	Byte67	1044h	Byte68
1045h	Byte69	1046h	Byte70	1047h	Byte71

### 13.2.5 16\*32 UCG Data Format

UCG with 16\*32 size needs 64 bytes data. For example, CGRAM start address is 1000h and the first UCG encoding is 0000h, the data will be saved in 1000h~103Fh, then the second UCG encoding will be 0001h and its data will be saved in 1040h~107Fh. Below formula shows how the 16\*32 UCG address in Memory is calculated, and Table 13-9 explains the data format and data byte sequence:

$$\text{UCG\_ADD} = \text{CGRAM\_Start\_ADD} + (\text{UCG\_Code} * 64)$$

Table 13-9: Data Format and Byte Sequence of UGC

UCG Code: 0000h			
Address	Data	Address	Data
1000h	Byte0	1001h	Byte1
1002h	Byte2	1003h	Byte3
1004h	Byte4	1005h	Byte5
1006h	Byte6	1007h	Byte7
:	:	:	:
:	:	:	:
103Ah	Byte58	103Bh	Byte59
103Ch	Byte60	103Dh	Byte61
103Eh	Byte62	103Fh	Byte63

### 13.2.6 32\*32 UCG Data Format

UCG with 32\*32 size needs 128 bytes data. For example, CGRAM start address is 1000h and the first UCG encoding is 0000h, the data will be saved in 1000h~107Fh, then the second UCG encoding will be 0001h and its data will be saved in 1080h~10FFh. Below formula shows how the 32\*32 UCG address in Memory is calculated, and Table 13-10 explains the data format and data byte sequence:

$$\text{UCG\_ADD} = \text{CGRAM\_Start\_ADD} + (\text{UCG\_Code} * 128)$$

**Table 13-10: Data Format and Byte Sequence of 32\*32 UCG**

UCG Code: 0000h							
Address	Data	Address	Data	Address	Data	Address	Data
1000h	Byte0	1001h	Byte1	1002h	Byte2	1003h	Byte3
1004h	Byte4	1005h	Byte5	1006h	Byte6	1007h	Byte7
1008h	Byte8	1009h	Byte9	100Ah	Byte10	100Bh	Byte11
100Ch	Byte12	100Dh	Byte13	100Eh	Byte14	100Fh	Byte15
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
1074h	Byte116	1075h	Byte117	1076h	Byte118	1077h	Byte119
1078h	Byte120	1079h	Byte121	107Ah	Byte122	107Bh	Byte123
107Ch	Byte124	107Dh	Byte125	107Eh	Byte126	107Fh	Byte127

### 13.2.7 Initialize CGRAM from MCU

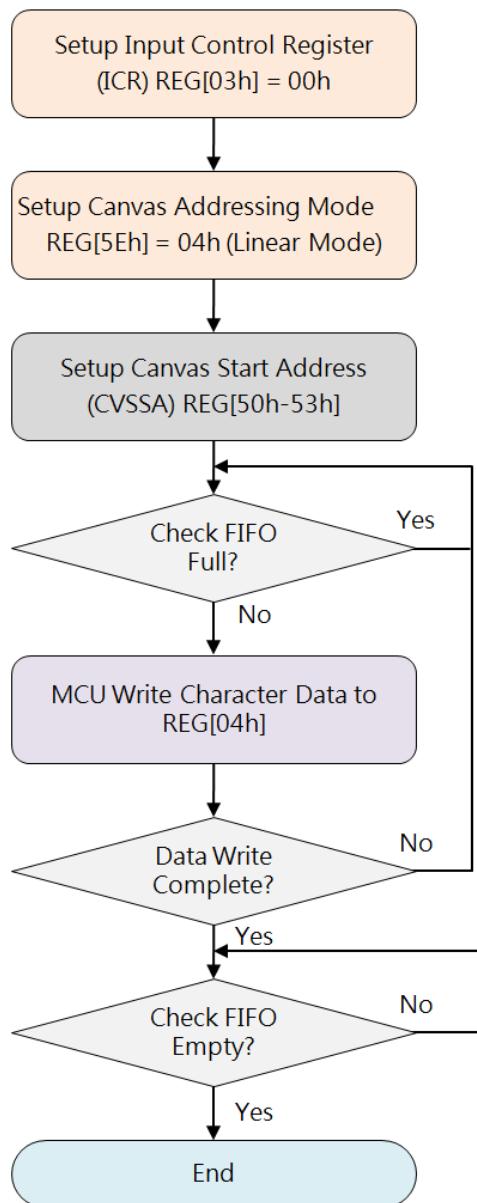


Figure 13-3: Flow Chart of CGRAM Initialization from MCU

### 13.2.8 Initialize CGRAM from Serial Flash

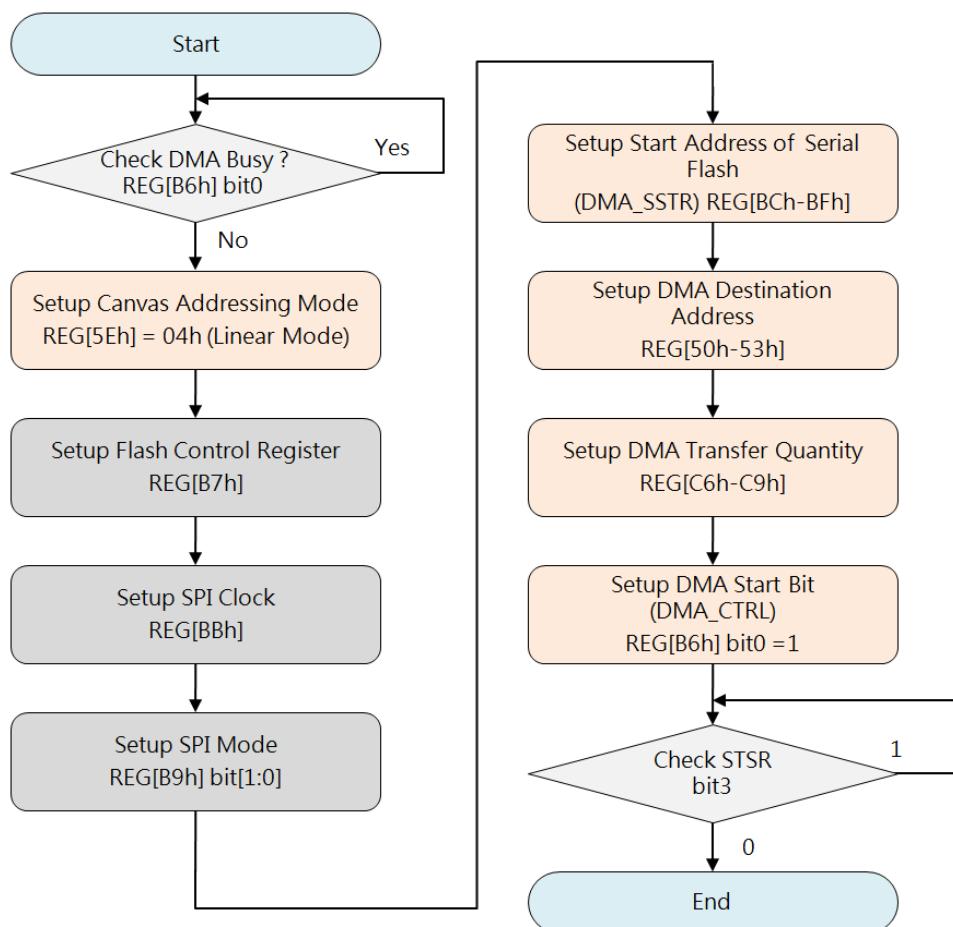


Figure 13-4: Flow Chart of CGRAM Initialization from Serial Flash

### 13.3 Character Rotation by 90 Degree

LT768x supports character rotation by counterclockwise 90 degree. Normal (REG[CDh] bit4=0) text direction is from left to right then from top to bottom. If set REG[CDh] bit4=1, the character will be counterclockwise rotated by 90 degree, and flipped in vertical. Also, the text direction will be changed to from top to bottom then from left to right. However, to get the correct display result, the Display Scan Direction must be changed by setting VDIR Reg[12h] bit3=1 (Please note that Text Cursor, Graphic Cursor, and PIP will be disabled automatically under this setting). Below example shows a set of characters rotated by 90 degree:

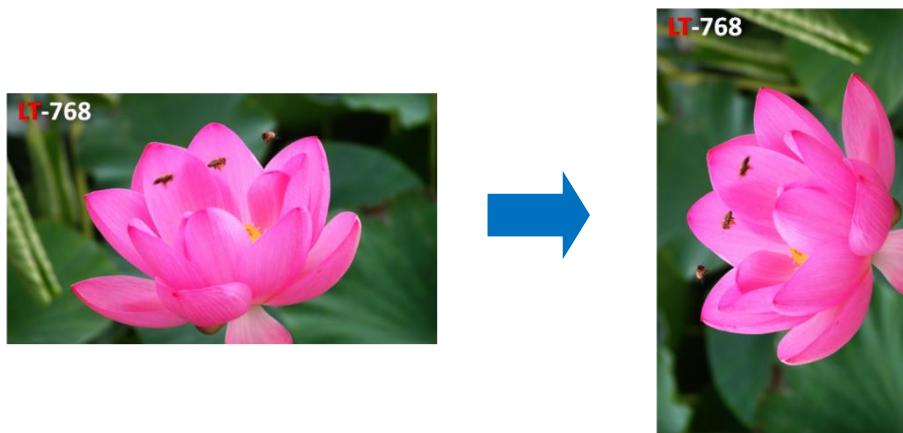


Figure 13-5: Example of Character Rotation

### 13.4 Size Enlargement

LT768x supports linear \*1, \*2, \*3, \*4 character size enlargement for Height and/or Width, controlled by REG[CDh] bit[3:0].

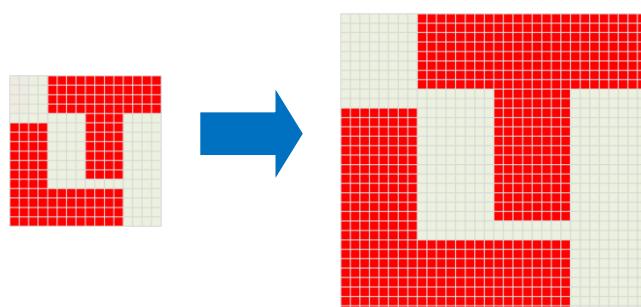


Figure 13-6: Example of Size Enlargement

### 13.5 Background Transparency

LT768x supports character background transparent, controlled by REG[CDh] bit6.

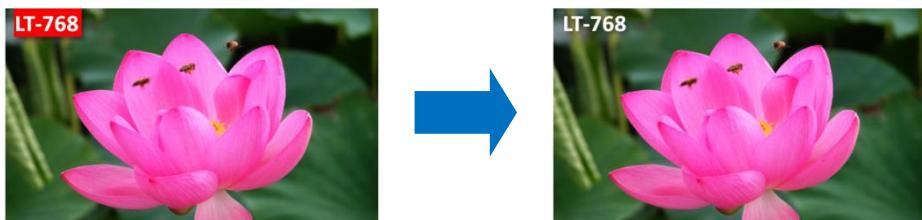


Figure 13-7: Example of Background Transparency

### 13.6 Character Full-Alignment

LT768x supports character full-alignment that makes the character to align each other. When inputting and displaying Half or Full size characters, set REG[CDh] bit7=1.

這是一款高效能TFT LCD图形加速显示芯片。其主要的功能就是协助MCU将所要显示到TFT屏的内容传递给TFT驱动器，并且提供PIP、图形加速、几何图形绘图等功能，除了提升显示效率外，还降低MCU处理图形显示所花费的时间。



這是一款高效能TFT LCD 图形加速显示芯片。其主要的功能就是协助MCU 将所要显示到TFT 屏的内容传递给TFT 驱动器，并且提供PIP 、图形加速、几何图形绘图等功能，除了提升显示效率外，还降低MCU 处理图形显示所花费的时间。

Figure 13-8: Example of Character Full-Alignment

### 13.7 Automatic Line Feed

LT768x supports sequent text input and display, and can perform automatically Line Feed at active window boundary.

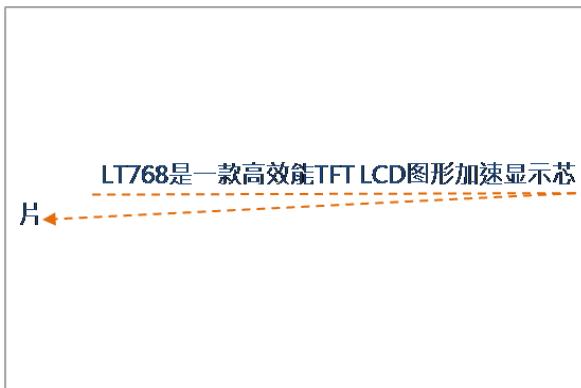


Figure 13-9: Example of Automatic Line Feed

### 13.8 Cursor

LT768x supports 2 types of Cursor, Graphic Cursor and Text Cursor. The Graphic Cursor is 32\*32 pixel graphic with 2-bits color index which can be displayed at an user-defined position. The Text Cursor is bit-wise graphic with 32\*32 as the maximum size, and is used to point text input position. Note that when Vertical Scan Direction is set to "From bottom to Top" (VDIR REG[12h] bit=1), Text Cursor, Graphic Cursor, as well as PIP will be disabled automatically.

#### 13.8.1 Text Cursor

Text Cursor has Auto-move, Blinking, and Enlargement functions. Once enabled, the Text Cursor appears at the input position, and can automatically move to the next input position after the current input is completed. Auto-move also supports Auto Line Feed, but is dominated by Active Window. Therefore, Text Cursor must be positioned in active window and be used in Text Mode. Settings for moving distance and direction are the same as Character input settings.

Table 13-11: Registers Related with Text Cursor

Register Address	Register Name	Description
REG[03h]	ICR	bit2: Graphic/Text Mode Selection (Text Mode Enable)
REG[3Ch]	GTCCR	bit1: Text Cursor Enable
		bit0: Text Cursor Blinking Enable)
REG[64h:63h]	F_CURX	X_Position: Text Input X coordinate
REG[66h:65h]	F_CURY	Y_Position: Text Input Y coordinate
REG[D0h]	FLDR	Text Line Gap Setting

### ■ Text Cursor Blinking

Text Cursor Blinking can be enabled by setting GTCCR ( REG[3Ch] ) bit[0]=1, and disabled by setting bit[0]=0. Blinking interval can be obtained through below formula:

$$\text{Blink\_Time (sec)} = \text{BTCR[3Dh]} * (1/\text{Frame\_Rate})$$



Figure 13-10: Example of Text Cursor Blinking

### ■ Text Cursor Height and Width

Text Cursor Height and Width are controlled by CURHS ( REG[3Eh] ) and CURVS ( REG[3Fh] ).

If Character enlargement is enabled, Text Cursor enlargement is also enabled automatically according to the same enlargement settings.

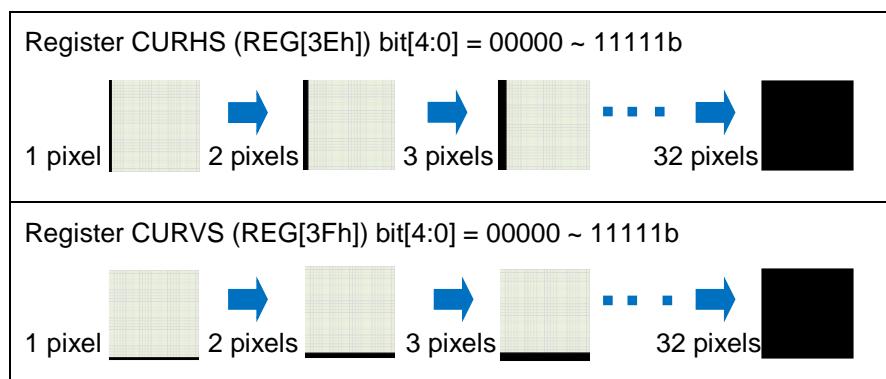


Figure 13-11: Example of Text Cursor Height and Width Settings

### 13.8.2 Graphic Cursor

LT768x Graphic Cursor is 32\*32 pixel graphic, and the color of each pixel is represented by 2 bits data. The color data are defined as shown in Table 13-12:

Table 13-12: Graphic Color Definition

<b>2'b00</b>	Color-0, defined by GCC0 (REG[44h])
<b>2'b01</b>	Color-1, defined by GCC1 (REG[45h])
<b>2'b10</b>	Background Color
<b>2'b11</b>	Inversed Background Color

To create a Graphic Cursor, it takes 256 bytes (32x32x2/8), Figure 13-12 shows the data format and byte sequence. LT768x supports 4 sets of graphic cursor that users can choose from by setting GTCCR REG[3Ch] bit[3:2]. Display position of graphic cursor is controlled by register GCHP0 (REG[40h]), GCHP1(REG[41h]), GCVP0(REG[42h]) and GCVP1(REG[43h]).

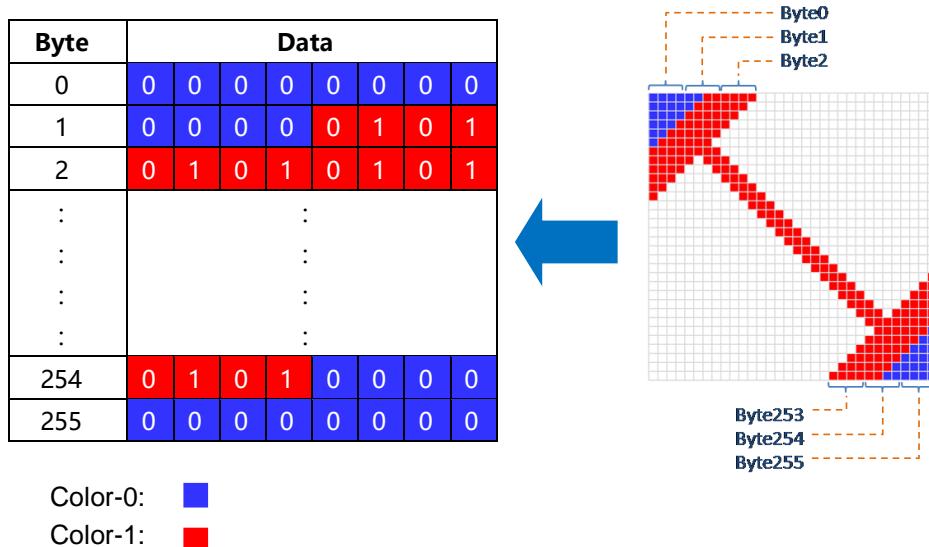


Figure 13-12: Graphic Cursor Data Format and Example

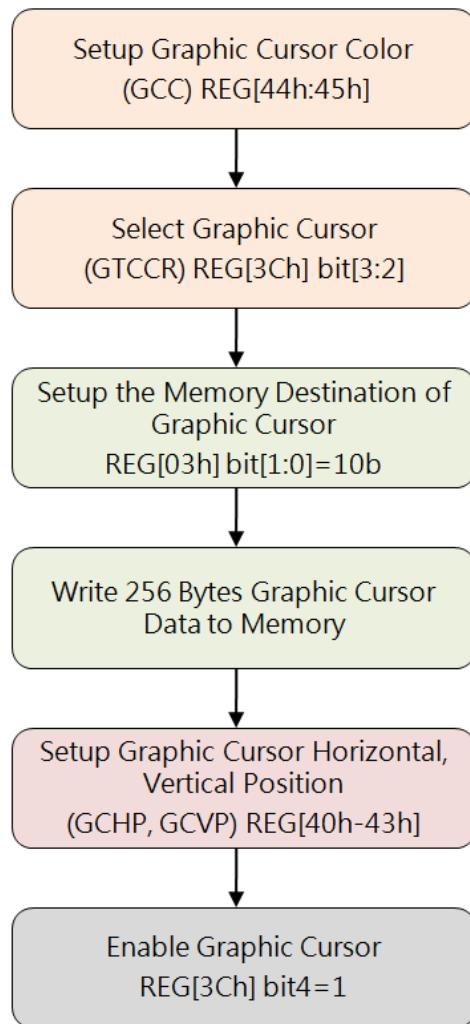
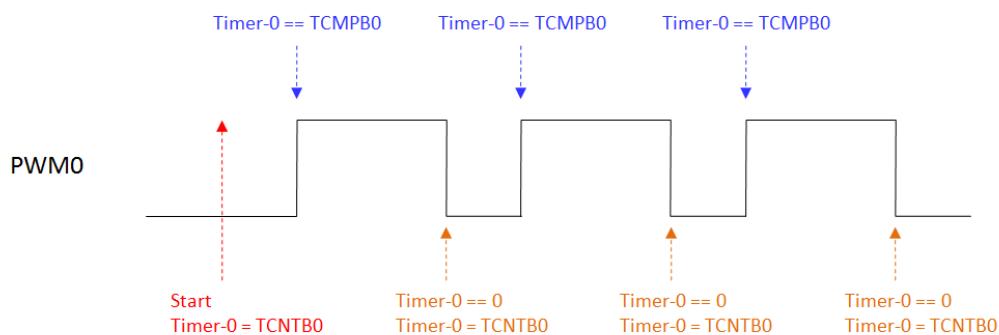


Figure 13-13: Flow Chart of Graphic Cursor Creation

## 14. Pulse Width Modulation (PWM)

LT768x has built-in PWM functions, and provides two PWM signal outputs: PWM0 and PWM1. LT768x also has embedded two 16bits counters, Timer-0 and Timer-1, and its action is related to the output state of the PWM signals. For example, before using the PWM0, the Host must set Timer-0 count registers (TCNTB0, REG[8Ah-8Bh],) and Timer-0 count comparison registers (TCMPB0, REG[88h-89h]). After the PWM function is enabled, the Timer-0 counter will first load the TCNTB0 value and start counting down according to the PWM clock frequency. When the value of the Timer-0 counter equals the value of the TCMPB0 register, PWM will be active. This means if the original state of PWM0 is 0, it will change to 1. The Timer-0 counter will continue to count down, and when Timer-0 equals 0, then an interrupt will be generated. The state of PWM0 will be back to the original 0, and also automatically reload TCNTB0 value. The above procedure represents a complete PWM cycle.



**Figure 14-1: PWM0 Output Wave Form**

Based on the above waveform and description, the duty cycle of PWM0 is determined by comparison registers (TCMPB0, REG[88h-89h]). For example, to generate a voltage of approximate DC potential by PWM0, when the PWM0 is initially set to 0, the larger value of the TCMPB0 is set, the higher equivalent voltage on PWM0 will be. On the other hand, when the PWM0 is initially set to 1, the smaller value of the TCMPB0 is set, the higher equivalent voltage on PWM0 will be.

**Note:** The automatic reload feature of PWM0 (REG[86h] bit1) must be activated, and when Timer-0 equals 0, it will reload the value of the TCNTB0 automatically. Therefore, if the MCU changes TCNTB0 or TCMPB0 value before Timer-0 equals 0, PWM can produce different duty-cycle waveform.

### 14.1 PWM Clock Source

The PWM's clock source comes from CCLK(System Clock), while the Timer-0 and Timer-1 base frequencies are determined by register PSCLR (REG[84h]):

$$\text{PWM\_CLK} = \text{CCLK} / (\text{Prescaler} + 1)$$

The clock source of Timer is determined by the respective frequency registers (REG[85h]). The Timer Divisor provides four options: 1, 1/2, 1/4, 1/8 for the Timer's clock. For example the REG[85h] Bit[5:4] = 10b, then the Timer-0 count Clock = System\_clock/4. Please refer to the Register Description of the Chapter 19 for REG[84H] and REG[85h].

## 14.2 PWM Output

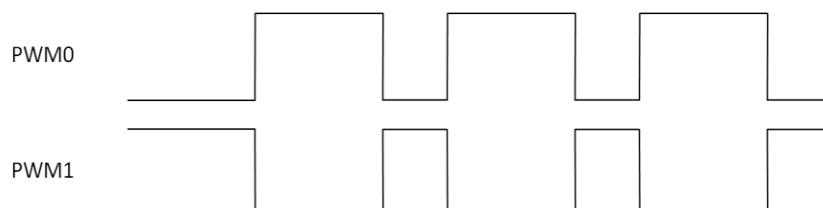
The output of PWM can also be set to a fixed high or low level. For PWM0, first to turn off the automatic reload feature (REG[86h] bit1 = 0), and stop counting down Timer-0 (REG[86h] bit0 = 0), if Timer-0 < TCMP0, then PWM output high. If Timer-0 > TCMP0, then PWM output low (assuming the reverse phase is closed, REG[86h] bit2=0). The output state of the PWM0 can be set to the inverse phase by REG[86h] bit2.

In addition, PWM0 and PWM1 are multiplex output pins that can be used for other purposes, please refer to the following description of Register REG[85h] bit[3:0].

**Table 14-1: REG[85h] Description**

Bit	Description
3-2	<b>PWM[1] Function Control</b> 0xb: PWM[1] output system error flag (Scan FIFO POP error or Memory access out of range) 10b: PWM[1] output PWM timer 1 event or invert of PWM timer 0 11b: PWM[1] output Oscillator Clock
1-0	<b>PWM[0] Function Control</b> 0xb: PWM[0] becomes GPIOC[7] 10b: PWM[0] output PWM Timer 0 11b: PWM[0] output Core Clock (CCLK)

PWM0 and PWM1 can also be set to complementary outputs. In this case, the output state of PWM1 follows the setting and control of PWM0, except that it is a PWM0 reverse output state:



**Figure 14-2: The Complementary Outputs of PWM0 and PWM1**

In some applications, when using Complementary Outputs of PWM0 and PWM1, changing state of PWM0 and PWM1 at the same time may cause excessive current. LT768x provides a dead-zone timing control to stagger the output state of PWM0 and PWM1. The dead-zone length of PWM is set by register REG[87h] (DZ\_LENGTH):

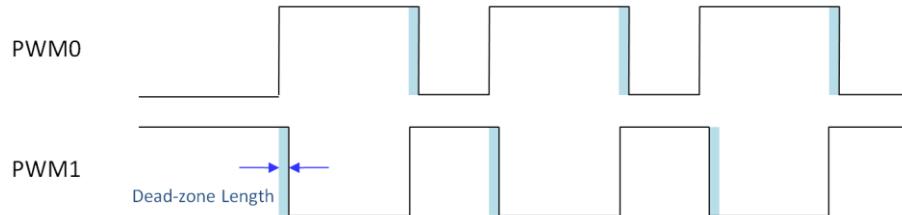
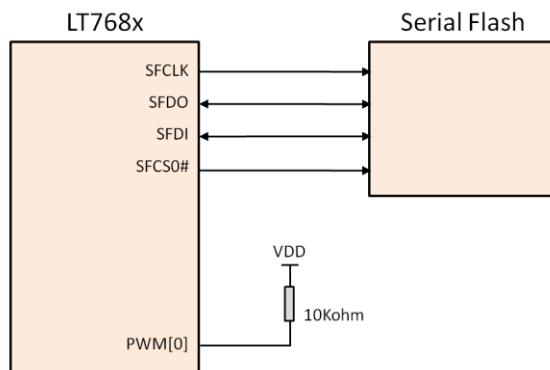


Figure 14-3: dead-zone of PWM0 and PWM1

## 15. Serial Bus Master

### 15.1 Power-on Display

LT768x's Power-on Display circuit has embedded a small microprocessor unit. The main function is to quickly display the screen at boot time by executing the program code stored in flash memory in the absence of Host, or when the Host is still in its start-up phase. To use this function, PWM[0] pin must connect a 10K pull-up resistor, then the "Power-On Display" function will be enabled (Refer to Figure15-1). In this case where the function is enabled, LT768x will automatically execute the program until the program code in the Flash memory is fully executed. After that, Host will retrieve the control of the system.



**Figure 15-1: Enable the "Power-On Display" Function**

The power-on display feature restricts that program code and display data must be stored in the same Flash memory. The Power-on Display unit supports 12 instructions as follows:

**Table 15-1: 12 Power-on Display Instructions**

No.	Instruction	Description	instruction Length (Byte)
1	EXIT	Exit instruction (00h/FFh)	1
2	NOP	NOP instruction (AAh)	1
3	EN4B	Enter 4-Byte mode instruction (B7h)	1
4	EX4B	Exit 4-Byte mode instruction (E9h)	1
5	STSR	Status read instruction (10h)	2
6	CMDW	Command write instruction (11h)	2
7	DATR	Data read instruction (12h)	2
8	DATW	Data write instruction (13h)	2
9	REPT	Load repeat counter instruction (20h)	2
10	ATTR	Fetch Attribute instruction (30h)	2
11	JUMP	Jump instruction (80h)	5
12	DJNZ	Decrement & Jump instruction (81h)	5

**One Byte Instruction:****■ Exit instruction (EXIT) – 00h | FFh | Undefined instructions**

EXIT instruction is executed to exit the Power-on Display function, and return control right to Host.

**■ NOP instruction (NOP) – AAh**

It will do nothing and then fetch next instruction.

**■ Enter 4-Byte mode instruction (EN4B) – B7h**

This instruction enables accessing the address length of 32-bit for the memory area of higher density (larger than 128Mb). The control unit default is in 24-bit address mode; after sending out the EN4B instruction, the address length becomes 32-bits instead of the default 24-bits. There are three methods to exit the 4-bytes mode: writing Exit 4-bytes Mode (EX4B) instruction, Hardware Reset or Power-off.

**■ Exit 4-Byte mode instruction (EX4B) – E9h**

The EX4B instruction is executed to exit the 4-bytes address mode and return to the default 3-bytes address mode. Once exiting the 4-bytes address mode, the address length will return to 24-bits.

**Two Bytes Instruction:****■ Load repeat counter instruction (REPT) – 20h + param[0]**

This instruction contains one byte parameter. This parameter is mainly for repeating counter value and used with DJNZ instruction.

**■ Fetch attribute instruction (ATTR) – 30h + param[0]**

This instruction contains one byte parameter. This instruction is used to configure controller to access serial Flash data. In Param[0]:

\_bit [3:0] is clock divisor for SPI clock. This is used to set proper SPI clock based on the system clock. Default is 0.

$$Fsck = Fcore / [ (Divisor + 1) * 2 ]$$

\_bit [4] is device mode selection for [CPOL, CPHA]. Value '0' is mode 0, value '1' is mode 3. Default is 1 for mode 3.

\_bit [5] is to define SFCS[1:0] deselect time which is also called chip select high time (tCSH). Value '0' is 4 core clocks; value '1' is 8 clocks. Default is 8 core clocks.

\_bit [7:6] is dummy cycle number. These two bits set 4 kinds of dummy cycle. The Value 0, 1, 2 or 3 stands for 0, 8, 16 or 24 dummy cycles. Default is 0. If dummy cycle number is 0 then serial flash read command code is 03h, otherwise serial flash read command code is 0Bh.

**■ Status read instruction (STSR) – 10h + param[0]**

The parameter represents the expected value in Status read **instruction**. If returned data dose not match with the expected value, this read instruction will be repeated.

**■ Command write instruction (CMDW) – 11h + param[0]**

The parameter represents the value that is to be written through CMDW instruction.

**■ Data read instruction (DATR) – 12h + param[0]**

The parameter represents an expected value. After issuing DATR, if the returned data does not match with the expected value, the instruction will be exected repeatedly.

**■ Data write instruction (DATW) – 13h + param[0]**

The parameter represents the written value by Data write instruction.

**Five Bytes Instruction:****■ Jump instruction (JUMP) – 80h + param[3] + param[2] + param[1] + param[0]**

It contains 4-bytes parameters. Parameter 3~0 are serial flash's 28-bits address information. i.e. param[3] is address[27:24], param[2] is address[23:16], param[1] is address[15:8] and param[0] is address[7:0]. After execution, next instruction will be fetched from this specified address.

■ **Decrement & Jump while not equal to zero (DJNZ) instruction – 81h + param[3] + param[2] + param[1] + param[0]**

Parameter 3~0 are serial flash's 28-bits address information. i.e. param[3] is address[27:24], param[2] is address[23:16], param[1] is address[15:8] and param[0] is address[7:0]. If the repeat counter equals zero, then the next instruction will be fetched from "current instruction address + 5", otherwise the repeat counter will decrement one and jump to specified address.

After the Power-on Reset, the Power-on Display function will search the two SPI interfaces provided by LT768x. The first 8 bytes must be "61h, 72h, 77h, 63h, 77h, 62h, 78h, 67h". If the flash memory is identified then the subsequent processing address will be (0008h), otherwise the master control will be transferred to the Host. LT768x internal microprocessor will start to execute instructions from the address 0008h of the flash memory. During the execution of Power-on Display, if an "Exit" or undefined instruction is executed, the processor will hand over the control to the host. Here is an example of using an external serial Flash and show a 1024\*768 picture on an 1024\*768 resolution TFT panel:

```
// Addr: 'h0000
61 72 77 63 77 62 78 67 // ID

//Initial PLL
11 05 13 8A // REG_WR('h05, 'h8A), Write 0x8A to REG[05]
11 06 13 41 // REG_WR('h06, 'h41)
11 07 13 8A // REG_WR('h07, 'h8A)
11 08 13 64 // REG_WR('h08 'h64)
11 09 13 8A // REG_WR('h09,'h8A)
11 0A 13 64 // REG_WR('h0A, 'h64)
11 00 13 80 // REG_WR('h00, 'h80)
11 01 13 82 // REG_WR('h01, 'h82)
11 01 12 82 // REG_WR('h01, 'h82)
11 02 13 40 // REG_WR('h02, 'h40)
AA AA AA AA // NOP

//Initial Display RAM
11 E0 13 29 // REG_WR('hE0, 'h29)
11 E1 13 03 // REG_WR('hE1, 'h03)
11 E2 13 0B // REG_WR('hE2, 'h0B)
11 E3 13 03 // REG_WR('hE3, 'h03)
11 E4 13 01 // REG_WR('hE4, 'h01)
AA AA AA AA // NOP

//Setup LCD Panel
11 10 13 04 // REG_WR('h10, 'h04)
11 12 13 85 // REG_WR('h12, 'h85)
11 13 13 03 // REG_WR('h13, 'h03)
11 14 13 7F // REG_WR('h14, 'h7F)
11 15 13 00 // REG_WR('h15, 'h00)
11 1A 13 FF // REG_WR('h1A, 'hFF)
11 1B 13 02 // REG_WR('h1B, 'h02)

// Setup Main Window
11 20 13 00 // REG_WR('h20, 'h00)
11 21 13 00 // REG_WR('h21, 'h00)
11 22 13 00 // REG_WR('h22, 'h00)
11 23 13 00 // REG_WR('h23, 'h00)
```

```

11 24 13 00      // REG_WR('h24, 'h00)
11 25 13 04      // REG_WR('h25, 'h04)
11 26 13 00      // REG_WR('h26, 'h00)
11 27 13 00      // REG_WR('h27, 'h00)
11 28 13 00      // REG_WR('h28, 'h00)
11 29 13 00      // REG_WR('h29, 'h00)
AA AA AA AA      // NOP

//Setup Canvas Window
11 50 13 00      // REG_WR('h50, 'h00)
11 51 13 00      // REG_WR('h51, 'h00)
11 52 13 00      // REG_WR('h52, 'h00)
11 53 13 00      // REG_WR('h53, 'h00)
11 54 13 00      // REG_WR('h54, 'h00)
11 55 13 04      // REG_WR('h55, 'h04)

//Setup Active Window
11 56 13 00      // REG_WR('h56, 'h00)
11 57 13 00      // REG_WR('h57, 'h00)
11 58 13 00      // REG_WR('h58, 'h00)
11 59 13 00      // REG_WR('h59, 'h00)
11 5A 13 00      // REG_WR('h5A, 'h00)
11 5B 13 04      // REG_WR('h5B, 'h04)
11 5C 13 00      // REG_WR('h5C, 'h00)
11 5D 13 03      // REG_WR('h5D, 'h03)
11 5E 13 02      // REG_WR('h5E, 'h02)

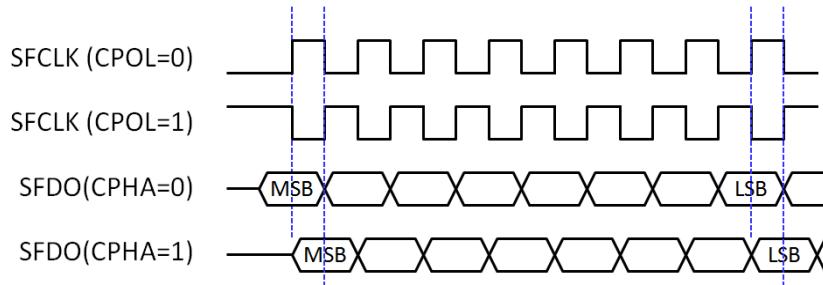
//Setup DMA Transfer Data from Flash to Display RAM
11 BC 13 00      // REG_WR('hBC, 'h00)
11 BD 13 02      // REG_WR('hBD, 'h02)
11 BE 13 00      // REG_WR('hBE, 'h00)
11 BF 13 00      // REG_WR('hBF, 'h00)
11 C0 13 00      // REG_WR('hC0, 'h00)
11 C1 13 00      // REG_WR('hC1, 'h00)
11 C2 13 00      // REG_WR('hC2, 'h00)
11 C3 13 00      // REG_WR('hC3, 'h00)
11 C6 13 00      // REG_WR('hC6, 'h00)
11 C7 13 04      // REG_WR('hC7, 'h04)
11 C8 13 00      // REG_WR('hC8, 'h00)
11 C9 13 03      // REG_WR('hC9, 'h03)
11 CA 13 00      // REG_WR('hCA, 'h00)
11 CB 13 04      // REG_WR('hCB, 'h04)
11 B7 13 C0      // REG_WR('hB7, 'hC0)
11 B6 13 01      // REG_WR('hB6, 'h01)
AA AA AA AA      // NOP
11 B6 12 00      // REG_WR('hB6, 'h00)
11 12 13 40      // REG_WR('h12, 'h40)
00                  // Exit

```

**Note:** Power-on display unit requests program codes & display data, fonts, and required contents for the program must be stored in the same serial Flash. If Host needs to switch to another serial Flash then all the codes & display data etc. will need to be stored to that serial Flash too.

## 15.2 SPI Master

When transferring data through SPI master of LT768x, the two serial data lines will be sampled and synchronized with the serial clock signal SFCLK. Master will place the relevant information on the SFDO signal line for the slave device to catch data in the first half of the clock edge. There are four possible protocol modes of CPOL and CPHA that can be chosen by setting the bit[1:0] of Serial Master Control Register [SPIMCR2]. The master and slave device must operate under the same frequency.



**Figure 15-2: Master SPI Data Transfer**

### Transmitting Data Bytes

The SPI transmission can be initialized after the control register is setup. The method to initialize data is to write data to the Register SPIDR (REG[B8h]). The data written to SPDIFR is actually written to a FIFO which has 16 bytes depths, also known as "Write-FIFO". Each write access adds a data byte to the Write-FIFO. When the SS\_Active is set to 1 and the FIFO is not empty, LT768x will start to transfer the data that is first written into "Write-FIFO" to the Slave.

### Receiving Data Bytes

Receiving data is done simultaneously with transmitting data; whenever a data byte is transmitted, a data byte is received. For each byte that needs to be read from a device, a dummy byte has to be written to the Write FIFO. This means using a dummy data write cycle to activate the SPI while receiving the data. Whenever the transfer is done, the received data is written into the "Read-FIFO". The "Read-FIFO" is the counterpart of the Write-FIFO. It is an independent 16 bytes deep FIFO. The FIFO contents can be read by reading from the Register [SPIDR] (REG[B8h]).

### FIFO Overrun

When FIFO is full, writing a new data into "Write-FIFO" will overwrite the oldest data. When writing data to "Write-FIFO" via [SPIDR] registers causes overflow, it will result in data errors. Then the data transmitted using the SPI interface will not be the first data entered, but the data that finally enter the FIFO. As shown in the example of Figure 15-3, WP is the write pointer, when "Write-FIFO" is full, and then next data written in FIFO will overwrite the first one. RP is the read pointer. If "Read-FIFO" is not performed before "Write-FIFO" is overflow, RP will stay at the first place. Therefore, once the overflow occurs, "Read-FIFO" will get wrong data.

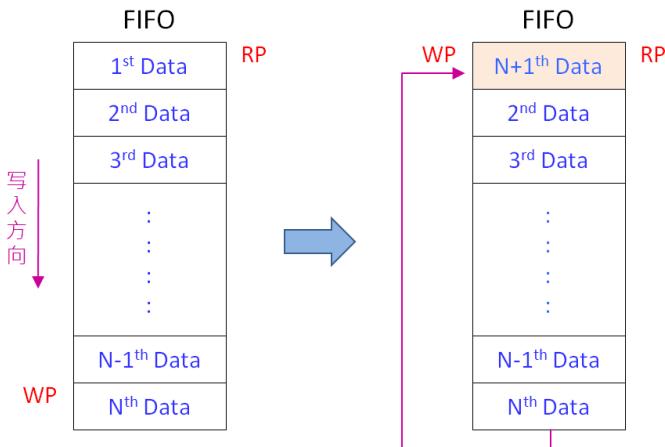


Figure 15-3: FIFO Overrun Example

The only way to recover from this situation is to reset the Write Buffer. Both the Read FIFO and the Write FIFO are reset when the Slave Select signal active [SS\_ACTIVE] bit is cleared ('0'). Read FIFO overruns might be less destructive. Especially when the SPI bus is used to transmit data only, and the received data are simply ignored. So the Read FIFO overruns are irrelevant. But, if the SPI bus is used to transmit and receive data, it is important to keep the Read FIFO aligned. The easiest way to do this is to perform a number of dummy reads. The number of dummy reads should be equal to the amount of bytes transmitted modulo 16.

$$\text{N}_{\text{dummy\_reads}} = \text{N}_{\text{transmitted\_bytes}} \bmod 16$$

**Note:** If the "Read-FIFO" is not empty, storing 16 of data is bound to cause overwritten. Therefore, before receiving every 16 bytes of data, the host must confirm if "Read-FIFO" is empty or not.

#### Reference Code for SPI Master Loop Test (Connect SFDO to SFDI)

```

REG_WR ('hBB, 8'h1f) ;           //Divisor, configure SPI clock frequency
REG_WR ('hB9, 8'b0001_1111) ;   // {1'b0, mask, SS#_sel, ss_active, ovfirqen, emtirqen,
                                // cpol, cpha}, SS# low
REG_WR ('hB8, 8'h55) ;          // TX
REG_WR ('hB8, 8'haa) ;          // TX
REG_WR ('hB8, 8'h87) ;          // TX
REG_WR ('hB8, 8'h78) ;          // TX
wait (INT#) ;
REG_RD ('hBA, acc) ;
while (acc != 8'h84) begin
    $display ("wait for FIFO empty ...") ;
    REG_RD ('hBA, acc) ;
end
REG_WR ('hBA, 8'h04) ;           // clear interrupt flag
REG_RD ('hB8, 8'h55) ;          // RX
REG_RD ('hB8, 8'haa) ;          // RX
REG_RD ('hB8, 8'h87) ;          // RX
REG_RD ('hB8, 8'h78) ;          // RX
REG_WR ('hB9, 8'b0000_1111) ;   // {1'b0, mask, SS#_sel, ss_active, ovfirqen, emtirqen,
                                // cpol, cpha}, SS# high.

```

### 15.3 Serial Flash Controller

LT768x builds in a SPI Master interface for Serial Flash/ROM, supporting for protocol of 4-BUS (Normal Read), 5-BUS (FAST Read), Dual mode 0, Dual mode 1 and Mode 0/Mode 3. Serial Flash/ROM function can be used for FONT mode and DMA mode. FONT mode means that the external serial Flash/ROM is treated as a source of character bitmap. DMA mode means that the external Flash/ROM is treated as the data source of DMA (Direct Memory Access). Host can speed up the data transfer to display memory and does not need to intervene in this mode.

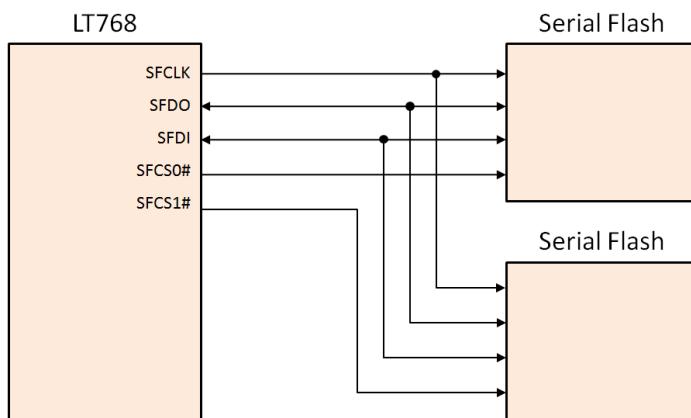


Figure 15-4: Application Circuit of Serial Flash/ROM

About Serial Flash/ROM read command protocol setting, please refer to Table 15-2 as below:

Table 15-2: Read Command Protocol of SPI Flash

REG [B7h] Bit[3:0]	Read Command Code
000xb	<b>1x Read Command Code – 03h</b> Normal read speed. Flash to LT768x data input is SFDI pin. Without dummy cycle between address and data.
010xb	<b>1x Read Command Code – 0Bh</b> To some Serial Flash provide Faster read speed. Flash to LT768x data input is SFDI pin. Eight dummy cycles inserted between address and data.
1x0xb	<b>1x Read Command Code – 1Bh</b> To some Serial Flash provide High read speed. Flash to LT768x data input is SFDI pin. Sixteen dummy cycles inserted between address and data.
xx10b	<b>2x Read Command Code – 3Bh</b> Interleaved data input on SFDI and SFDO pins. Eight dummy cycles inserted between address and data phase. (Mode 0)
xx11b	<b>2x Read Command Code – BBh</b> Address output & data input interleaved on SFDI and SFDO pins. Four dummy cycles inserted between address and data phase. (Mode 1)

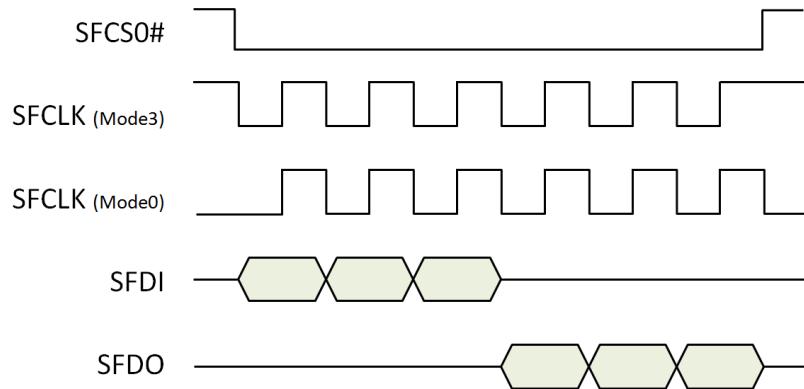


Figure 15-5: Mode 0 and Mode 3 Protocol

Figure 15-6 is a Timing diagram of Read Commands for SPI Flash Memory. If REG[B7h] bit5=0, the address line is 24bits and requires 24 clock. If REG[B7h] bit5=0, the address line is 32bits and requires 32 clock.

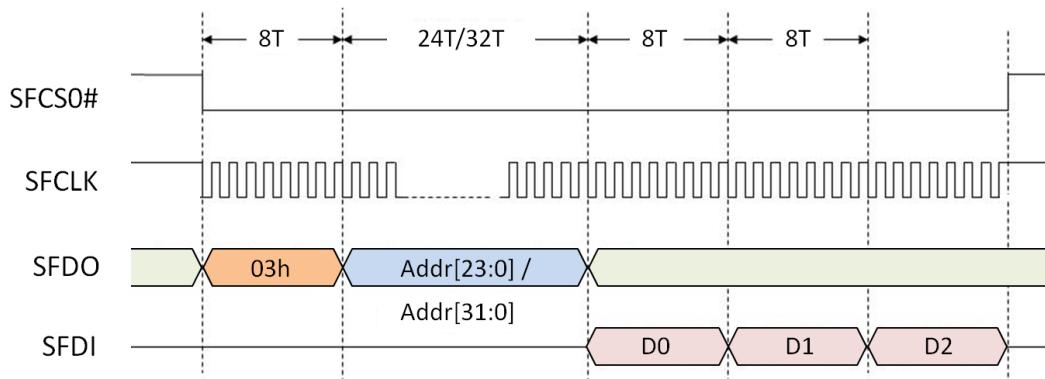


Figure 15-6: Normal Read Command of SPI Flash

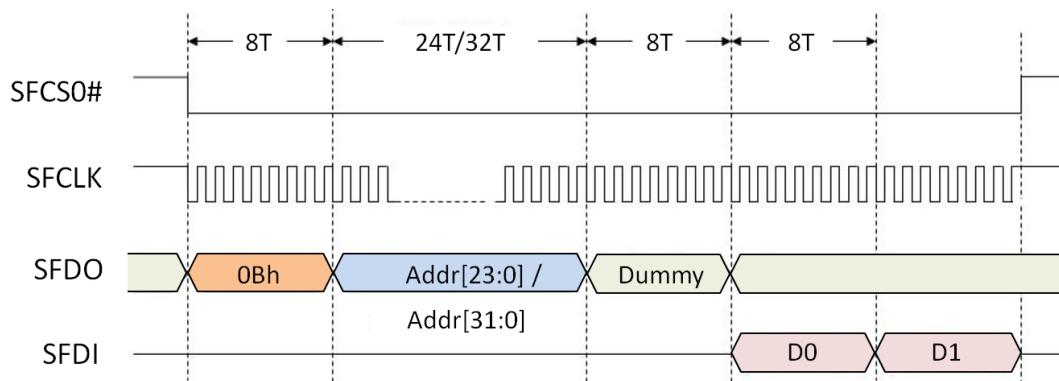
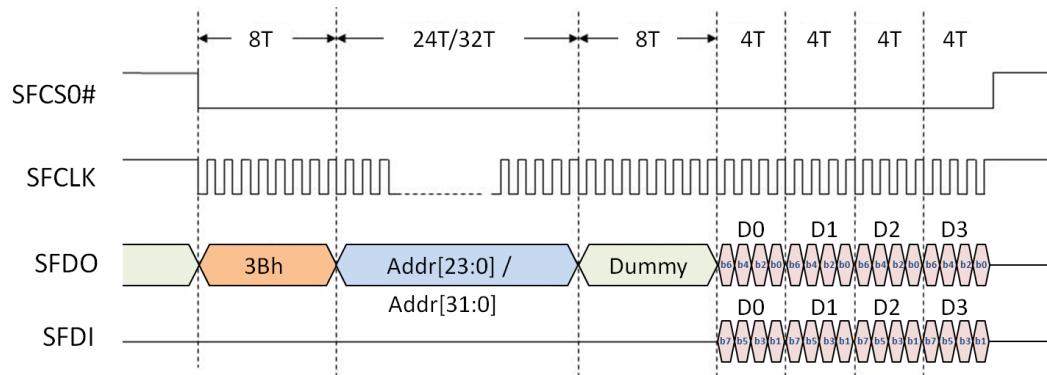


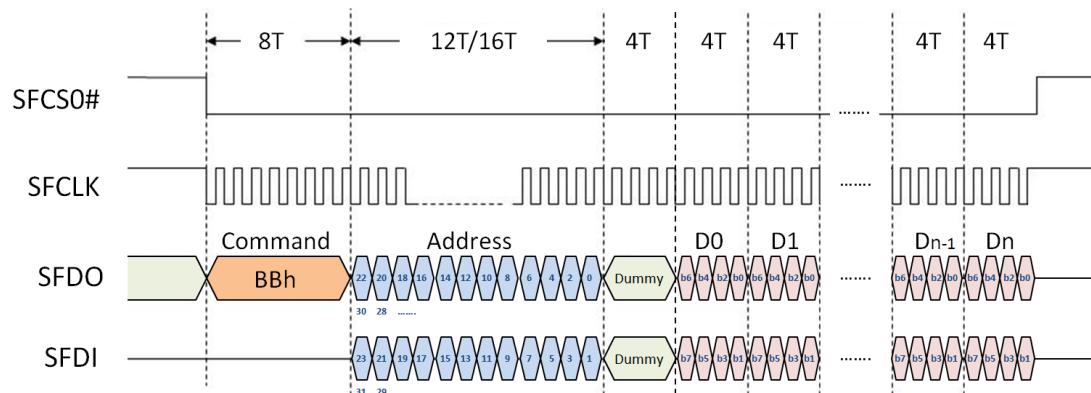
Figure 15-7: SPI Fast Read Command of SPI Flash

Figure 15-8 shows flash to LT768x data input as interleaved input, and data input appears on SFDI and SFDO pins. If REG[B7h] bit5=0 represents an address line of 24bits, it requires 24 clocks. If REG[B7h] bit5=1 represents an address line of 32bits, it requires 32 clocks.



**Figure 15-8: SPI Flash Read Command (Data and Address are Interleaved)**

Figure 15-9 shows LT768x address output and data input are interleaved, data and address are interleaved on SFDI and SFDO pins. If REG[B7h] bit5=0 on behalf of the address line is 24bits, because it is interleaved input so only 12 of clock. If REG[B7h] bit5=1 represents an address line of 32bits, only 16 clock are required.



**Figure 15-9: SPI Flash Read Command (Data and Address are Interleaved)**

### 15.3.1 External Serial Flash

External Flash Memory can be used as a source of the image data. In Graphics Mode, Host can use DMA (Direct Memory access) mode to access data of External Flash Memory. It means the Flash Memory can be used as the source of DMA, and developer can store a large amount of display data in advance. Therefore, Host does not need to take a lot of time to transfer large and commonly used graphics data to LT768x's Display RAM.

The data format of external Serial Flash Memory must be consistent with the format of the Display RAM. The graphic data format for Flash Memory are as follows:

**Table 15-3: 8bpp Image Data Format of Serial Flash**

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0001h	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	0000h	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>
0003h	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>	0002h	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>
0005h	R <sub>5</sub> <sup>7</sup>	R <sub>5</sub> <sup>6</sup>	R <sub>5</sub> <sup>5</sup>	G <sub>5</sub> <sup>7</sup>	G <sub>5</sub> <sup>6</sup>	G <sub>5</sub> <sup>5</sup>	B <sub>5</sub> <sup>7</sup>	B <sub>5</sub> <sup>6</sup>	0004h	R <sub>4</sub> <sup>7</sup>	R <sub>4</sub> <sup>6</sup>	R <sub>4</sub> <sup>5</sup>	G <sub>4</sub> <sup>7</sup>	G <sub>4</sub> <sup>6</sup>	G <sub>4</sub> <sup>5</sup>	B <sub>4</sub> <sup>7</sup>	B <sub>4</sub> <sup>6</sup>
0007h	R <sub>7</sub> <sup>7</sup>	R <sub>7</sub> <sup>6</sup>	R <sub>7</sub> <sup>5</sup>	G <sub>7</sub> <sup>7</sup>	G <sub>7</sub> <sup>6</sup>	G <sub>7</sub> <sup>5</sup>	B <sub>7</sub> <sup>7</sup>	B <sub>7</sub> <sup>6</sup>	0006h	R <sub>6</sub> <sup>7</sup>	R <sub>6</sub> <sup>6</sup>	R <sub>6</sub> <sup>5</sup>	G <sub>6</sub> <sup>7</sup>	G <sub>6</sub> <sup>6</sup>	G <sub>6</sub> <sup>5</sup>	B <sub>6</sub> <sup>7</sup>	B <sub>6</sub> <sup>6</sup>
0009h	R <sub>9</sub> <sup>7</sup>	R <sub>9</sub> <sup>6</sup>	R <sub>9</sub> <sup>5</sup>	G <sub>9</sub> <sup>7</sup>	G <sub>9</sub> <sup>6</sup>	G <sub>9</sub> <sup>5</sup>	B <sub>9</sub> <sup>7</sup>	B <sub>9</sub> <sup>6</sup>	0008h	R <sub>8</sub> <sup>7</sup>	R <sub>8</sub> <sup>6</sup>	R <sub>8</sub> <sup>5</sup>	G <sub>8</sub> <sup>7</sup>	G <sub>8</sub> <sup>6</sup>	G <sub>8</sub> <sup>5</sup>	B <sub>8</sub> <sup>7</sup>	B <sub>8</sub> <sup>6</sup>
000Bh	R <sub>11</sub> <sup>7</sup>	R <sub>11</sub> <sup>6</sup>	R <sub>11</sub> <sup>5</sup>	G <sub>11</sub> <sup>7</sup>	G <sub>11</sub> <sup>6</sup>	G <sub>11</sub> <sup>5</sup>	B <sub>11</sub> <sup>7</sup>	B <sub>11</sub> <sup>6</sup>	000Ah	R <sub>10</sub> <sup>7</sup>	R <sub>10</sub> <sup>6</sup>	R <sub>10</sub> <sup>5</sup>	G <sub>10</sub> <sup>7</sup>	G <sub>10</sub> <sup>6</sup>	G <sub>10</sub> <sup>5</sup>	B <sub>10</sub> <sup>7</sup>	B <sub>10</sub> <sup>6</sup>

**Table 15-4: 16bpp Image Data Format of Serial Flash**

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	R <sub>0</sub> <sup>4</sup>	R <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	0000h	G <sub>0</sub> <sup>4</sup>	G <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>2</sup>	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>	B <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>4</sup>	B <sub>0</sub> <sup>3</sup>
2	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	R <sub>1</sub> <sup>4</sup>	R <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	0002h	G <sub>1</sub> <sup>4</sup>	G <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>2</sup>	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	B <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>4</sup>	B <sub>1</sub> <sup>3</sup>
3	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	R <sub>2</sub> <sup>4</sup>	R <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	0004h	G <sub>2</sub> <sup>4</sup>	G <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>2</sup>	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>	B <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>4</sup>	B <sub>2</sub> <sup>3</sup>
4	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	R <sub>3</sub> <sup>4</sup>	R <sub>3</sub> <sup>3</sup>	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	0006h	G <sub>3</sub> <sup>4</sup>	G <sub>3</sub> <sup>3</sup>	G <sub>3</sub> <sup>2</sup>	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>	B <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>4</sup>	B <sub>3</sub> <sup>3</sup>
5	R <sub>4</sub> <sup>7</sup>	R <sub>4</sub> <sup>6</sup>	R <sub>4</sub> <sup>5</sup>	R <sub>4</sub> <sup>4</sup>	R <sub>4</sub> <sup>3</sup>	G <sub>4</sub> <sup>7</sup>	G <sub>4</sub> <sup>6</sup>	G <sub>4</sub> <sup>5</sup>	0008h	G <sub>4</sub> <sup>4</sup>	G <sub>4</sub> <sup>3</sup>	G <sub>4</sub> <sup>2</sup>	B <sub>4</sub> <sup>7</sup>	B <sub>4</sub> <sup>6</sup>	B <sub>4</sub> <sup>5</sup>	B <sub>4</sub> <sup>4</sup>	B <sub>4</sub> <sup>3</sup>
6	R <sub>5</sub> <sup>7</sup>	R <sub>5</sub> <sup>6</sup>	R <sub>5</sub> <sup>5</sup>	R <sub>5</sub> <sup>4</sup>	R <sub>5</sub> <sup>3</sup>	G <sub>5</sub> <sup>7</sup>	G <sub>5</sub> <sup>6</sup>	G <sub>5</sub> <sup>5</sup>	000Ah	G <sub>5</sub> <sup>4</sup>	G <sub>5</sub> <sup>3</sup>	G <sub>5</sub> <sup>2</sup>	B <sub>5</sub> <sup>7</sup>	B <sub>5</sub> <sup>6</sup>	B <sub>5</sub> <sup>5</sup>	B <sub>5</sub> <sup>4</sup>	B <sub>5</sub> <sup>3</sup>

**Table 15-5: 24bpp Image Data Format of Serial Flash**

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	G <sub>0</sub> <sup>7</sup>	G <sub>0</sub> <sup>6</sup>	G <sub>0</sub> <sup>5</sup>	G <sub>0</sub> <sup>4</sup>	G <sub>0</sub> <sup>3</sup>	G <sub>0</sub> <sup>2</sup>	G <sub>0</sub> <sup>1</sup>	G <sub>0</sub> <sup>0</sup>	0000h	B <sub>0</sub> <sup>7</sup>	B <sub>0</sub> <sup>6</sup>	B <sub>0</sub> <sup>5</sup>	B <sub>0</sub> <sup>4</sup>	B <sub>0</sub> <sup>3</sup>	B <sub>0</sub> <sup>2</sup>	B <sub>0</sub> <sup>1</sup>	B <sub>0</sub> <sup>0</sup>
2	B <sub>1</sub> <sup>7</sup>	B <sub>1</sub> <sup>6</sup>	B <sub>1</sub> <sup>5</sup>	B <sub>1</sub> <sup>4</sup>	B <sub>1</sub> <sup>3</sup>	B <sub>1</sub> <sup>2</sup>	B <sub>1</sub> <sup>1</sup>	B <sub>1</sub> <sup>0</sup>	0002h	R <sub>0</sub> <sup>7</sup>	R <sub>0</sub> <sup>6</sup>	R <sub>0</sub> <sup>5</sup>	R <sub>0</sub> <sup>4</sup>	R <sub>0</sub> <sup>3</sup>	R <sub>0</sub> <sup>2</sup>	R <sub>0</sub> <sup>1</sup>	R <sub>0</sub> <sup>0</sup>
3	R <sub>1</sub> <sup>7</sup>	R <sub>1</sub> <sup>6</sup>	R <sub>1</sub> <sup>5</sup>	R <sub>1</sub> <sup>4</sup>	R <sub>1</sub> <sup>3</sup>	R <sub>1</sub> <sup>2</sup>	R <sub>1</sub> <sup>1</sup>	R <sub>1</sub> <sup>0</sup>	0004h	G <sub>1</sub> <sup>7</sup>	G <sub>1</sub> <sup>6</sup>	G <sub>1</sub> <sup>5</sup>	G <sub>1</sub> <sup>4</sup>	G <sub>1</sub> <sup>3</sup>	G <sub>1</sub> <sup>2</sup>	G <sub>1</sub> <sup>1</sup>	G <sub>1</sub> <sup>0</sup>
4	G <sub>2</sub> <sup>7</sup>	G <sub>2</sub> <sup>6</sup>	G <sub>2</sub> <sup>5</sup>	G <sub>2</sub> <sup>4</sup>	G <sub>2</sub> <sup>3</sup>	G <sub>2</sub> <sup>2</sup>	G <sub>2</sub> <sup>1</sup>	G <sub>2</sub> <sup>0</sup>	0006h	B <sub>2</sub> <sup>7</sup>	B <sub>2</sub> <sup>6</sup>	B <sub>2</sub> <sup>5</sup>	B <sub>2</sub> <sup>4</sup>	B <sub>2</sub> <sup>3</sup>	B <sub>2</sub> <sup>2</sup>	B <sub>2</sub> <sup>1</sup>	B <sub>2</sub> <sup>0</sup>
5	B <sub>3</sub> <sup>7</sup>	B <sub>3</sub> <sup>6</sup>	B <sub>3</sub> <sup>5</sup>	B <sub>3</sub> <sup>4</sup>	B <sub>3</sub> <sup>3</sup>	B <sub>3</sub> <sup>2</sup>	B <sub>3</sub> <sup>1</sup>	B <sub>3</sub> <sup>0</sup>	0008h	R <sub>2</sub> <sup>7</sup>	R <sub>2</sub> <sup>6</sup>	R <sub>2</sub> <sup>5</sup>	R <sub>2</sub> <sup>4</sup>	R <sub>2</sub> <sup>3</sup>	R <sub>2</sub> <sup>2</sup>	R <sub>2</sub> <sup>1</sup>	R <sub>2</sub> <sup>0</sup>
6	R <sub>3</sub> <sup>7</sup>	R <sub>3</sub> <sup>6</sup>	R <sub>3</sub> <sup>5</sup>	R <sub>3</sub> <sup>4</sup>	R <sub>3</sub> <sup>3</sup>	R <sub>3</sub> <sup>2</sup>	R <sub>3</sub> <sup>1</sup>	R <sub>3</sub> <sup>0</sup>	000Ah	G <sub>3</sub> <sup>7</sup>	G <sub>3</sub> <sup>6</sup>	G <sub>3</sub> <sup>5</sup>	G <sub>3</sub> <sup>4</sup>	G <sub>3</sub> <sup>3</sup>	G <sub>3</sub> <sup>2</sup>	G <sub>3</sub> <sup>1</sup>	G <sub>3</sub> <sup>0</sup>

DMA function provides a faster method for users to update/transfer mass data to display memory. The only source of DMA function in LT768x is external Serial Flash/ROM. There are two kinds of data type defined for the DMA. One is Linear Mode and the other is Block Mode. It provides a flexible selection for user applications. The destination of DMA function is dominated by active window in display memory. When DMA function is active, the specific data from Serial Flash/ROM will be transferred one by one to Display Memory by LT768x automatically. After the DMA function is completed, LT768x will generate an interrupt to notify the Host. Please refer to the following sections for more information.

### 15.3.2 DMA in Linear Mode for External Serial Flash

The DMA Linear Mode is used to send CGRAM data for Display RAM. The color depth of the Active window must be set as 8bpp.

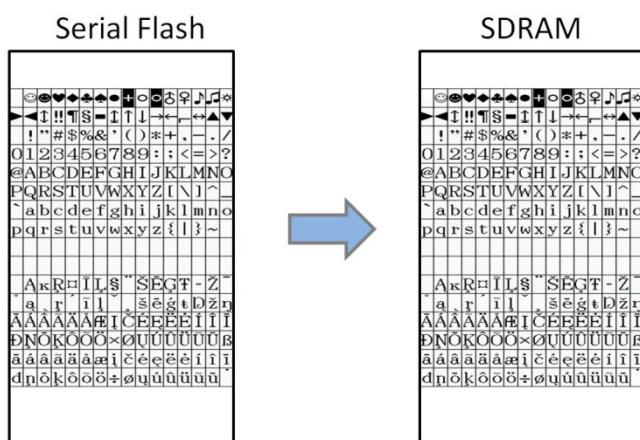


Figure 15-10: DMA in Linear Mode for External Serial Flash

### 15.3.3 DMA in Block Mode for External Serial Flash

DMA access in this block mode is primarily used to transfer graphic data. The process unit is pixel. Please refer to following diagram and flowchart.

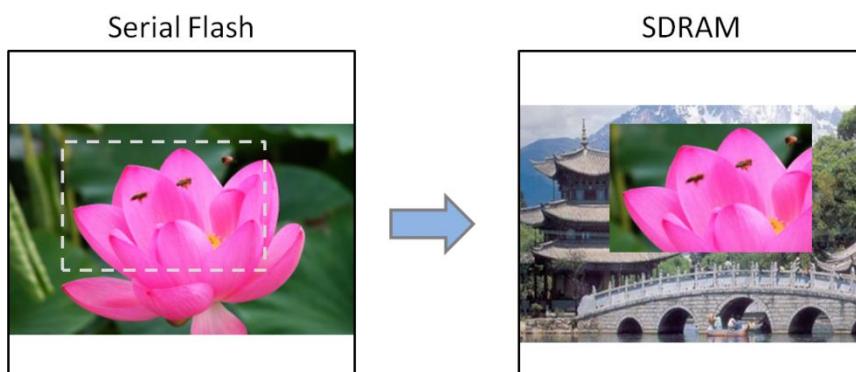


Figure 15-11: DMA in Block Mode for External Serial Flash

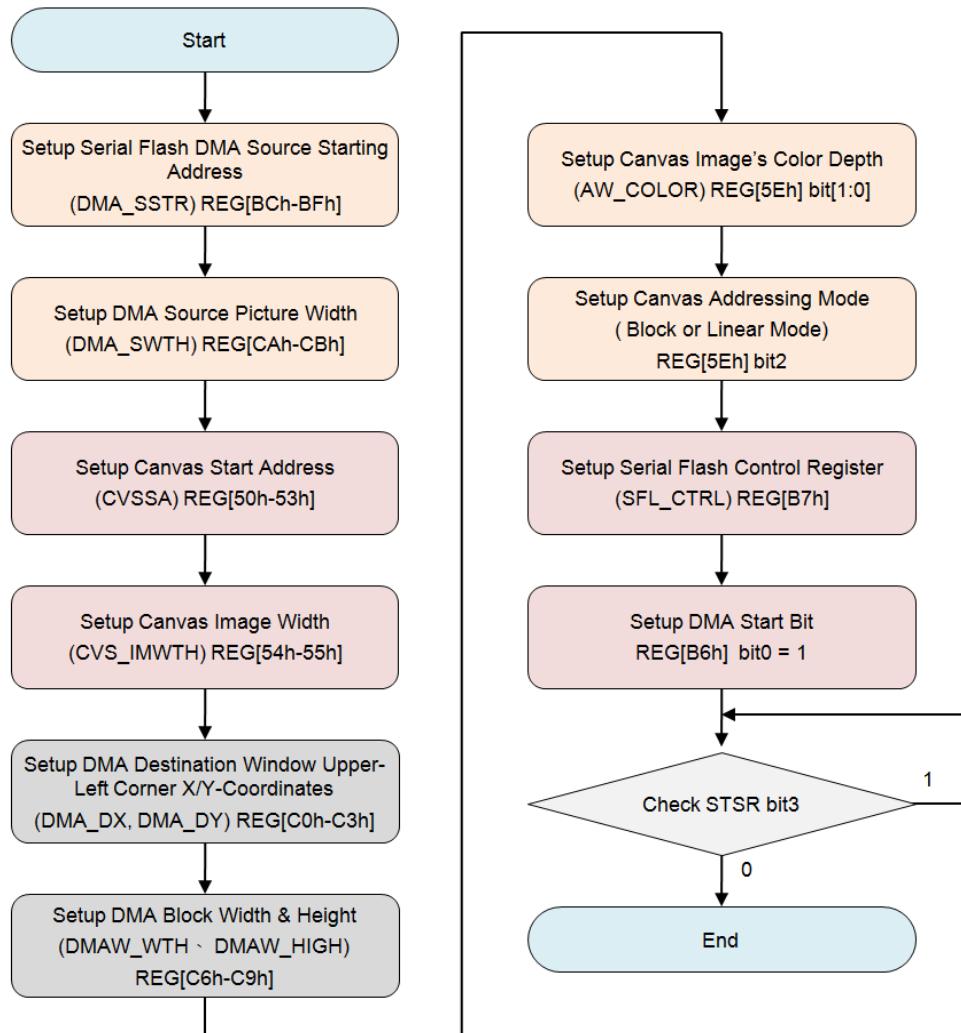


Figure 15-12: DMA Data Transfer Flowchart (Polling Mode)

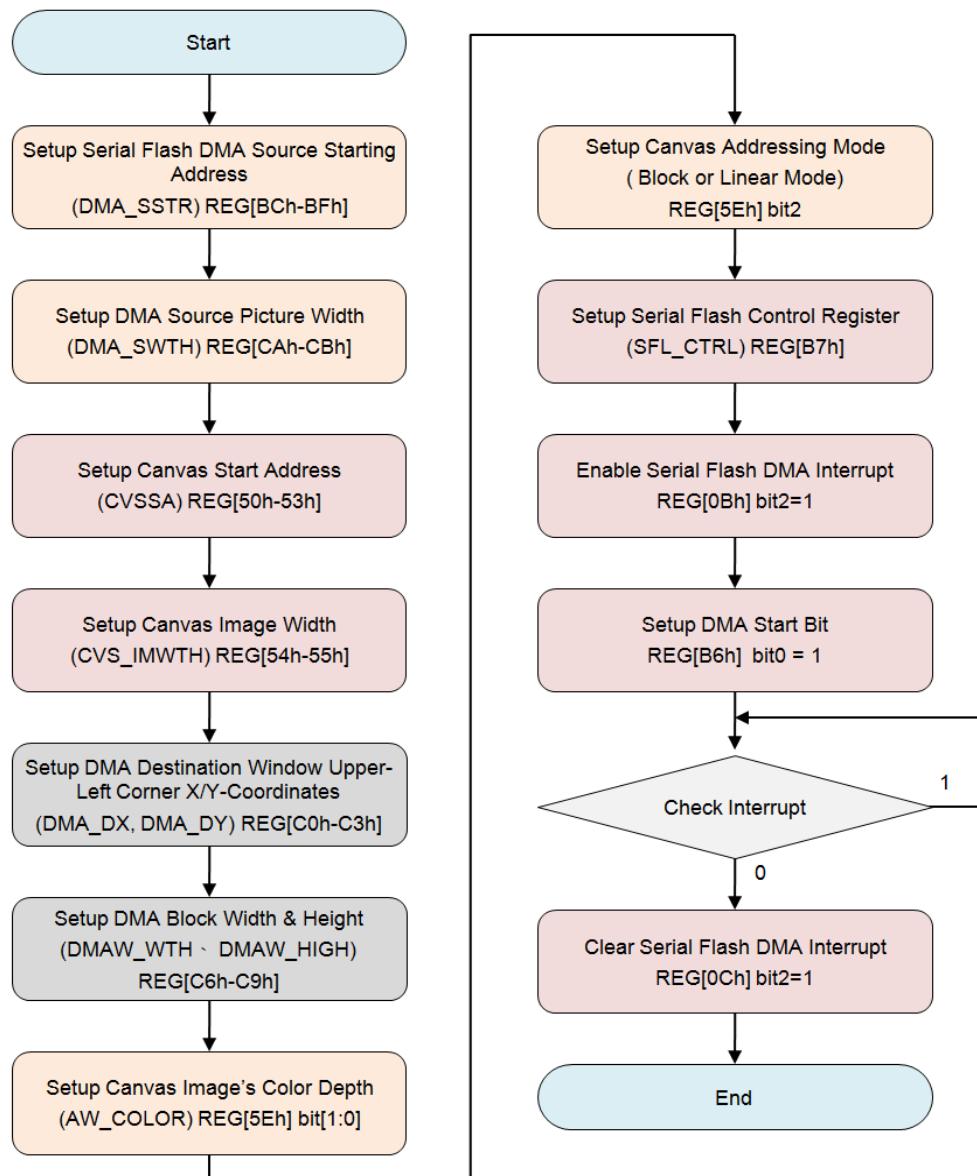


Figure 15-13: DMA Data Transfer Flowchart (Interrupt Mode)

## 15.4 I2C Master

I2C Master is a two-wires, bi-directional serial bus that provides a simple and efficient method of data exchange between devices.

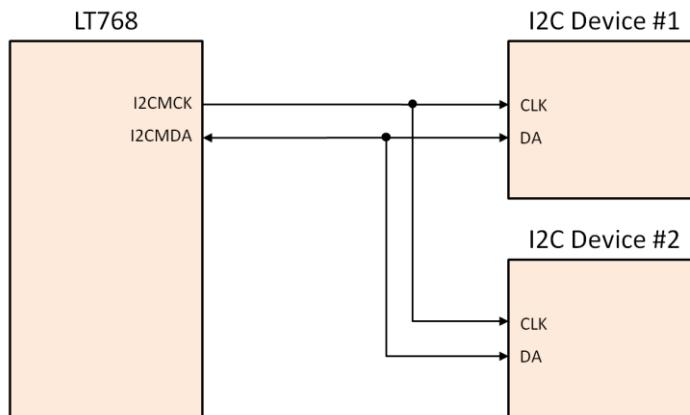


Figure 15-14: Application Circuit of LT768x I2C Master

LT768x supports 100Kbps and 400Kbps transfer rate for I2C bus. The formula of I2C Master clock(I2CMCK) is as the following:

$$\text{I2CMCK} = \text{CCLK} / [5^* (\text{Prescaler} + 2)]$$

For example, if I2CMCK is 100 KHz and CCLK is 50 MHz, pre-scalar must be set to 98. Data transfer between I2C Master and Slave is synchronized by I2CMCK on the basis of byte. Each data byte is 8bit. There is one I2CMCK pulse for each I2CMDA bit and the MSB will be transmitted first. And then an acknowledge bit will be transmitted for each transferred byte. Each bit is processed during the high period of I2CMCK so that the I2CMDA could be changed only during the low period of I2CMCK and must be held stable during the high period of I2CMCK.

The standard Communication Protocol of I2C consists of four parts:

- Start Signal
- Slave Address Transfer
- Data Transfer
- End Signal

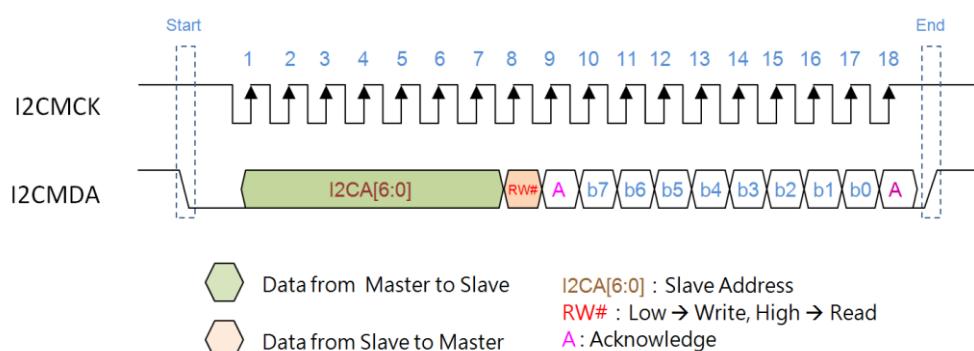
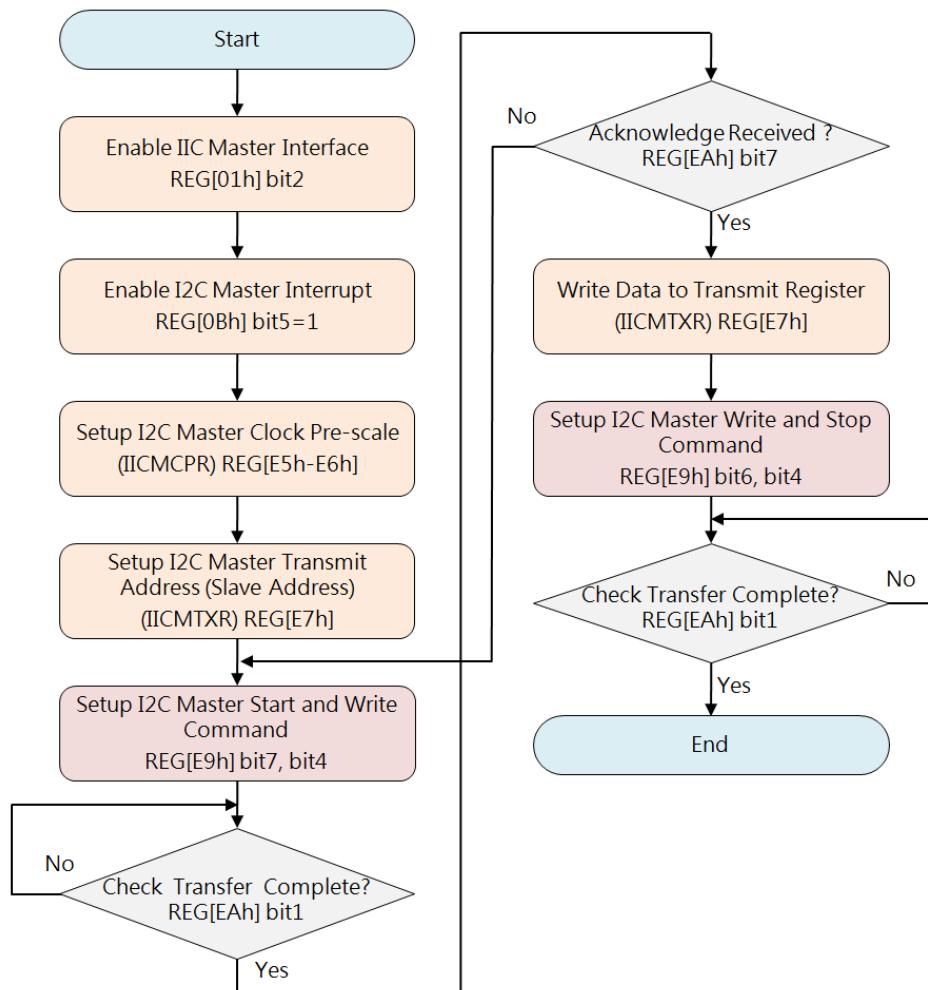
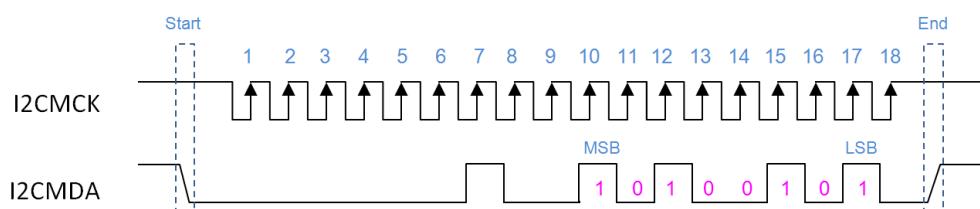
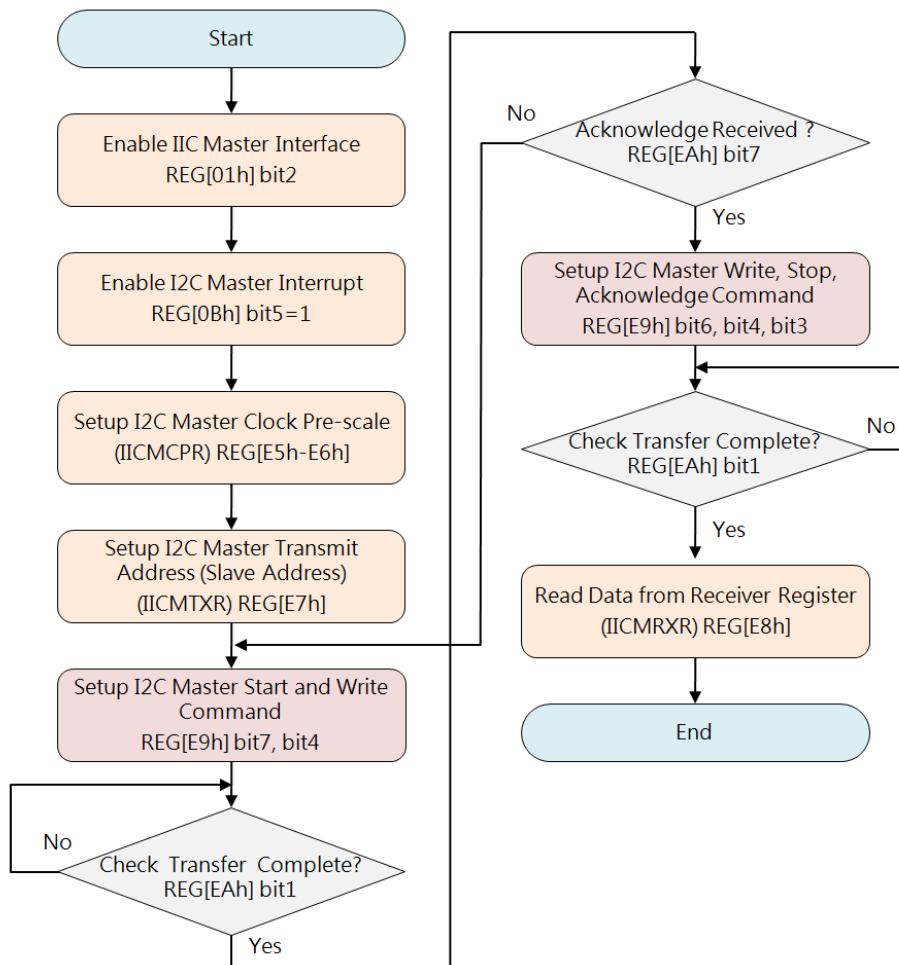
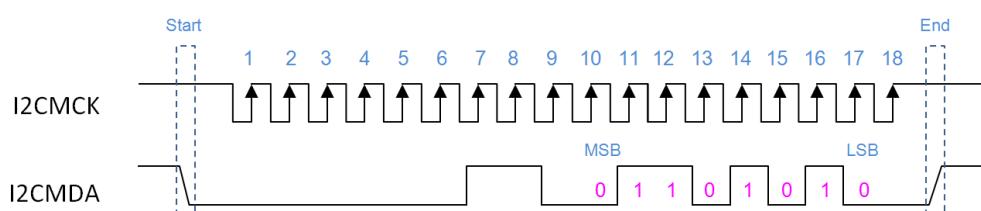


Figure 15-15: I2C Communication Protocol

**Example 1. Write 1 Byte data to Slave Device:****Figure 15-16: Write Data to Slave****Figure 15-17: Write Data "A5"to Slave (Address=0x01)**

**Example 2. Read 1 Byte Data from Slave Device:****Figure 15-18: Read Data from Slave****Figure 15-19: Read Data "6A" from Slave (Address=0x01)**

**Note:** The I2CMCK and I2CMDA are multiplex pins that share with KI[0] and KO[0]. And only LQFP-128pin package (LT7681, LT7683, LT7686) supports I2C Master function.

## 16. Keypad-Scan

LT768x provides a set of 5x5 keyboard scanning circuits. In addition to saving MCU IO interface, it can also reduce the hardware and software loading of MCU. By setting severral registers, Host can easily get the keyboard data. The following diagram is the basic keyboard application circuit.

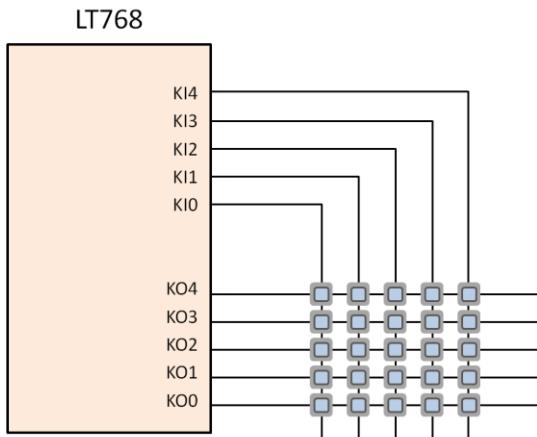


Figure 16-1: Application Circuit of Keypad-scan

### 16.1 Keypad-scan Operation

The embedded Keypad-scan features of LT768x are as below:

- Support up to 5\*5 Keypad matrix
- Programmable setting of sampling times and scan frequency of Keypad-scan
- Adjustable long Key-press timing
- Support Multi Key-press
- Support Key-stroke wakeup function

The Keypad State Register KSCR is used to control the function of Keypad-scan, such as sampling time, sampling frequency, and enabling long keystrokes. If the key is pressed, The Host can also get the interrupt to know whether any keypad is pressed. The Register KSCR2 bit[1:0] records the number of keystrokes pressed. Host can get the corresponding key-code of pressed keys by reading register KSDR.

Table 16-1: Key-code of Normal-press Key

	KI0	KI1	KI2	KI3	KI4
KO0	00h	01h	02h	03h	04h
KO1	10h	11h	12h	13h	14h
KO2	20h	21h	22h	23h	24h
KO3	30h	31h	32h	33h	34h
KO4	40h	41h	42h	43h	44h

Table 16-1 and Table 16-2 show the corresponding Key-code of Normal-press key and Long-press key. The Key-code will be stored in the Registers KSDR0 ~ KSDR2 when there is any key pressed. If users press the Keypad button for a long time, then LT768x will correspond different key-code saved in Registers (KSDR0 ~ KSDR2).

**Table 16-2: Key-code of Long-press Key**

	KI0	KI1	KI2	KI3	KI4
<b>KO0</b>	80h	81h	82h	83h	84h
<b>KO1</b>	90h	91h	92h	93h	94h
<b>KO2</b>	A0h	A1h	A2h	A3h	A4h
<b>KO3</b>	B0h	B1h	B2h	B3h	B4h
<b>KO4</b>	C0h	C1h	C2h	C3h	C4h

When multiple keystrokes are pressed, up to three key-code will be presented in Register KSDR0, KSDR1 and KSDR2.

**Note:** The key-codes stored in Register (KSDR0 ~ KSDR2) is related to the location of the keys, not to the sequence of keys. For example, if the key code 0x34, 0x00, 0x22 are pressed at the same time, then the KSDR0 ~ KSDR2 may store the key-code as follows:

$$\text{KSDR0} = 0x00$$

$$\text{KSDR1} = 0x22$$

$$\text{KSDR2} = 0x34$$

The following is the related Register of Keypad-scan functions:

**Table 16-3: The Related Register of Keypad-scan**

Register Address	Register Name	Description
REG[FBh]	KSCR1	bit6: Long-press Key Enable
		bit[5:4]: Short Key De-bounce Times
		bit[2:0]: Keypad Row Scan Time
REG[FCh]	KSCR2	bit7: Keypad-Scan Wakeup Enable
		bit[4:2]: Long-press Key Recognition Factor
		bit[1:0]: Numbers of Key Pressed
REG[FDh ~ FFh]	KSDR0 ~ 2	Key Strobe Data1 ~ Data3
REG[01h]	CCR	bit5: Keypad-Scan Enable/Disable
REG[0Bh]	INTEN	bit3: Keypad-Scan Interrupt Enable
REG[0Ch]	INTF	bit3: Keypad-Scan Interrupt Flag

The Host can use two ways to check if there is any keypad pressed:

- Host checks the Keypad-scan Status Register
- Host checks the Interrupt Signal

If the Keypad Interrupt Enable (REG[0Bh] bit3=1), then the interrupt signal will be active when keypad is pressed. When interrupts occur, the Keypad-scan interrupt state flag (REG[0Ch] bit3) will be 1. So users must clear this Interrupt status flag after reading the key-code, otherwise there will be no next interruptions.

In addition, LT768x supports "key-press wakeup" in power-saving mode. After the register are setup completely, any keystroke triggers can awaken LT768x from sleep mode. In order to recognize the Wake-up event, Host can use software to poll LT768x interrupt status bit. As shown in Figure 16-2.

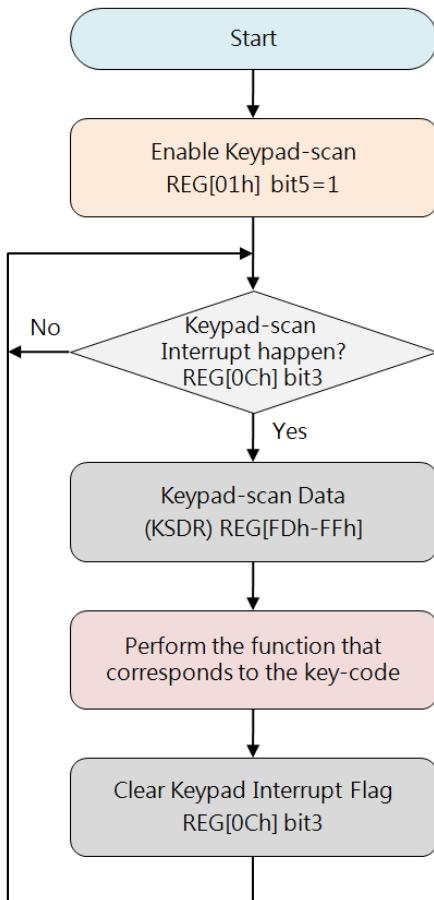


Figure 16-2: Flowchart to check Key-pad Interrupt ( S/W Polling Mode)

The Host can also be notified by hardware interrupt signal INT#, as shown in the Figure 16-3 flowchart.

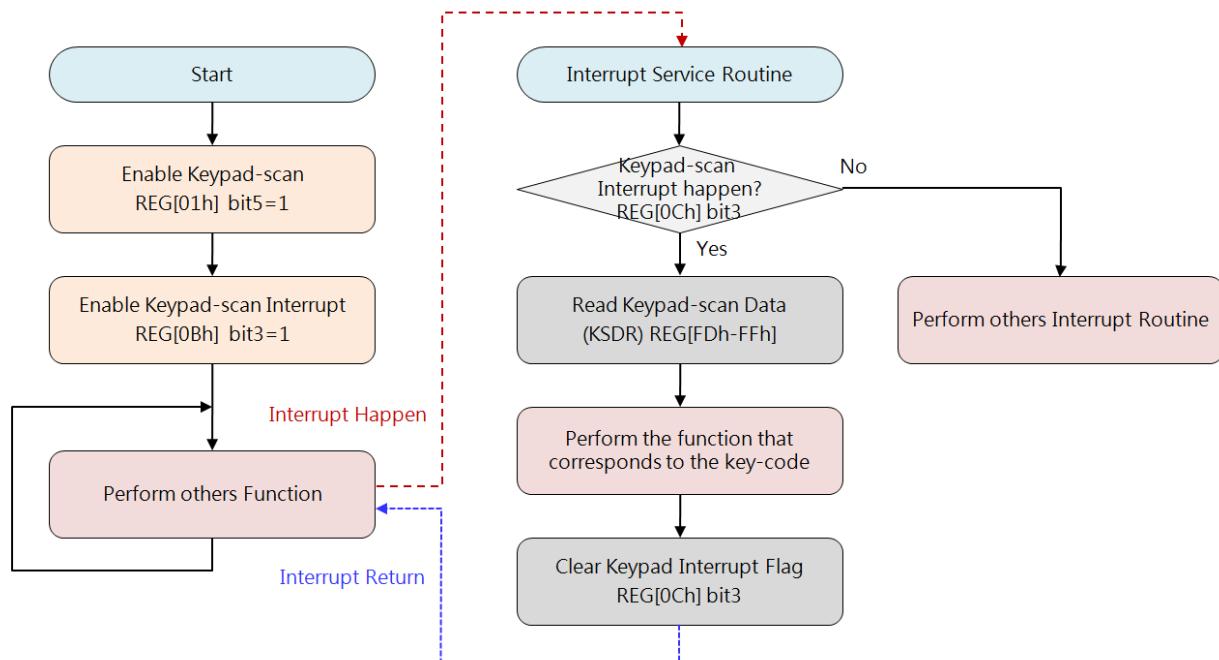


Figure 16-3: Flowchart to check Key-pad Interrupt (H/W Interrupt Mode)

## 17. GPIO Interface

LT768x provides many GPIO signals that can be extended as MCU I/O interfaces. Usually these GPIO signals are shared with other control signals. Please refer to Table 17-1 as below, and note these GPIO signals are available only when their mapping control signals are disabled.

**Table 17-1: GPIO Port vs. Shared Signals**

GPIO Port	Shared Signals
GPIOA[7:0]	DB[15:8]
GPIB[4],	KI[0]
GPOB[4]	KO[0]
GPIB[3:0]	{A0, WR#, RD#, CS#}
GPIOC[7]	PWM[0]
GPIOC[4:0],	{ SFCS1#, SFCS0#, SFDI, SFDO, SFCLK }
IOD[7:0]	PD[18, 2, 17, 16, 9, 8, 1, 0]

Table 17-2 is a supported GPIO signal list for different models of LT768x. These GPIO signals can be set to output or input by REG[F0h-F6h]. Please refer to Chapter 19 - Register Description.

**Table 17-2: The Supported GPIO Port for LT768x**

Model	GPIO Number	GPIO Port
LT7681 LT7683 LT7686	28	GPIOA[7:0] GPIB[4], GPOB[4], GPIB[3:0] GPIOC[7], GPIOC[4:0], GPIOD[7:0]
LT7680A LT7680B	7	GPIOA[7] GPIOC[7], GPIOC[4:0]

## 18. Power Management

LT768x has four operation modes, distinguished by the amount of power consumption, from high to Low: Normal mode, Standby mode, Suspend mode, and Sleep mode. The four modes of operation are set by the register REG[DFh]. Following is the comparison table for the related clock action of the four operation modes:

**Table 18-1: Power Management vs. Clock Active**

Item	Normal Mode	Standby Mode		Suspend Mode		Sleep Mode	
	PLL Enable	Parallel MCU	Serial MCU	Parallel MCU	Serial MCU	Parallel MCU	Serial MCU
MCLK	MPLL Clock	MPLL Clock	MPLL Clock	OSC	OSC	Stop	Stop
CCLK	CPLL Clock	OSC	OSC	Stop	OSC	Stop	OSC
PCLK	PPLL Clock	Stop	Stop	Stop	Stop	Stop	Stop
CPLL	ON	ON	ON	OFF	OFF	OFF	OFF
MPLL	ON	ON	ON	OFF	OFF	OFF	OFF
PPLL	ON	ON	ON	OFF	OFF	OFF	OFF

**Note:**

- When LT768x enters the power saving mode, the LCD interface will not output the signal. Therefore, before entering the power-saving mode, Host needs to issue “display off” or “power off” to the LCD module to avoid LCD polarization damage.
- The “OSC” means external X’tal oscillator.

### 18.1 Normal Mode

In this mode, three of the internal PLL functions are working normally. That is, Host setup Registers to control three PLL to generate CCLK (Core Clock), MCLK (Display RAM Clock), and PCLK (LCD scan Clock) so that the whole system can work normally. Because it will take some time for the PLL to work steadily, Host must first check if the PLL frequency is in a stable state through register 01h bit7.

### 18.2 Standby Mode

If setup REG[DFh] bit[1:0] to 01b, LT768x will enter Standby mode. The System Clock (CCLK) and LCD-Scan Clock (PCLK) will stop. The Display RAM Clock will remain active and still be provided by MCLK.

**Enter Standby Mode:**

- ◆ Select Standby Mode (REG[DFh] bit[1:0] = 01b)
- ◆ Setup REG[DFh] bit7 to 1 (Enter Standby Mode)
- ◆ Host may check the bit1 of Status Register (STSR) , and wait for this bit to become 1 to make sure LT768x enters Standby mode.

**Back to Normal Mode:**

- ◆ Setup REG[DFh] bit7 to 0 (Quit from Standby Mode)
- ◆ Host may check the bit1 of Status Register (STSR), and wait for this bit to become 0 to make sure LT768x is back to Normal Mode.

### 18.3 Suspend Mode

If REG[DFh] bit[1:0] is set to 10b, LT768x will enter Suspend mode. The System Clock (CCLK), Display RAM Clock(MCLK) and LCD-Scan Clock (PCLK) will stop. The clock of Display RAM will be provided by OSC Clock.

#### Enter Suspend Mode:

- ◆ Setup appropriate Display RAM(SDRAM) Refresh Clock based on the OSC frequency.
- ◆ Select Suspend Mode( REG[DFh] bit[1:0] = 10b)
- ◆ Setup REG[DFh] bit7 to 1 (Enter Suspend Mode)
- ◆ Host may check the bit1 of Status Register (STSR), and wait for this bit to become 1 to make sure LT768x enters Suspend Mode.

#### Back to Normal Mode:

- ◆ Setup REG[DFh] bit7 to 0 (Quit from Suspend Mode)
- ◆ Host may check the bit1 of Status Register (STSR), and wait for this bit to become 0 to make sure LT768x is back to Normal Mode.

### 18.4 Sleep Mode

If REG[DFh] bit[1:0] is set to 11b, LT768x will enter Sleep Mode, and all of Clocks and PLL will stop operating.

#### Enter Sleep Mode:

- ◆ Select Sleep Mode (REG[DFh] bit[1:0] = 11b)
- ◆ Setup REG[DFh] bit7 to 1 (Enter Sleep Mode)
- ◆ If REG[E0h] bit7 is 0, the Display RAM will enter Power Down mode before LT768x enters Sleep mode. If REG[E0h] bit7 is 1, the Display RAM will enter Refresh mode.
- ◆ If Host interface is Parallel Mode, then LT768x will stop OSC. If Host interface is Serial Mode, then OSC will not be stopped.
- ◆ Host may check the bit1 of Status Register (STSR), and wait for this bit to become 1 to make sure LT768x enters Sleep Mode.

#### Back to Normal Mode:

- ◆ Setup REG[DFh] bit7 to 0 (Quit from Sleep Mode)
- ◆ If OSC is stopped in Sleep mode, the Host has to enable OSC.
- ◆ The Host may check the bit1 of Status Register (STSR), and wait for this bit to become 0 to make sure LT768x is back to Normal Mode.

## 19. Register Description

LT768x provides a compatible 4 forms of Host interface, and the internal registers are read and written through these Host interface cycles. LT768x contains a Status Register and many Instruction Registers. Status Registers can read data through the state reading period, and it can only be read and cannot be written. Instruction Registers can control most of the functions through the "Command Write" cycle and the "Data Write" cycle. During the "Command Write" cycle, the address of the register is specified, and then during the "Data Write" cycle, data can be written to the specified register. When a specified register data is to be read, the master will need to send the "Command write" cycle first. Then use the "Data Read" cycle to read the data. In other words, "Command Write" is to set the register address, and "Data Read" is to read the register data.

**Table 19-1: Host Interface Cycle**

Host Interface Cycle	CS#	A0	8080 Type MCU		6800 Type MCU		Action Description
			RD# EN	WR# RW#	RD# EN	WR# RW#	
Command Write	0	0	1	0	1	0	Write Address of Register
Status Read	0	0	0	1	1	1	Read Status Register Data
Data Write	0	1	1	0	1	0	Write Data to Register or Memory
Data Read	0	1	0	1	1	1	Read Data from Register or Memory

All of the Registers of LT768x are described as below. Each register contains 8bits data. The register table includes the detail description, default value and access attribute (RO: Read Only, WO: Write Only, RW: Readable and Writeable).

### 19.1 Status Register

**Table 19-2: Status Register (STSR)**

Bit	Description	Default	Access
7	<b>Memory Write FIFO Full Indicate</b> 0: The FIFO of Memory Write not Full 1: The FIFO of Memory Write is Full  Host can write data to memory only if the Memory Write FIFO is not full.	0	RO
6	<b>Memory Write FIFO Empty Indicate</b> 0: The FIFO of Memory Write not Empty 1: The FIFO of Memory Write is Empty  When the Memory Write FIFO is empty, Host can continuously write 8bpp data with 64 pixels, 16bpp data with 32 pixels or 24bpp data with 16 pixels.	1	RO

Bit	Description	Default	Access
5	<b>Memory Read FIFO Full Indicate</b> 0: The FIFO of Memory Read not Full 1: The FIFO of Memory Read is Full When the Memory Read FIFO is Full, Host can continuously read 8bpp data with 64 pixels, 16bpp data with 32 pixels or 24bpp data with 16 pixels.	0	RO
4	<b>Memory Read FIFO Empty Indicate</b> 0: The FIFO of Memory Read not Empty 1: The FIFO of Memory Read is Empty When the Memory Read FIFO is not Empty, Host can continuously read pixel data from Memory.	1	RO
3	<b>Core Task is Busy, Fontwr_Busy</b> This bit represents whether the on-going action is completed (BTE, Geometry engine, DMA, text writing, or graphic writing). 0: The action is completed or idle. 1: The action is not completed, or in a busy state. When the Host program switches text and graphic mode, or changes the base diagram related settings, the program must first verify if the LT768x is idle. <b>Note:</b> In text mode, before program changes text rotation, line spacing, character spacing, foreground color, background color, and text/graphics settings, Host must confirm this bit is 0.	0	RO
2	<b>Display RAM Ready for Access</b> 0: Display RAM is not ready for access 1: Display RAM is ready for access Before checking this bit, Host has to setup REG[E4h] bit0 "SDR_INIT" as 1.	0	RO
1	<b>Operation Mode Status</b> 0: Normal operation state 1: Inhibit operation state Inhibit operation state means internal reset event is running, or initial display is running, or the chip enters power saving state. In power saving state, this bit becomes 1 until PLL clock stops.	0	RO
0	<b>Interrupt Pin State</b> 0: No interrupt activated 1: Interrupt activated	0	RO

## 19.2 Configuration Registers

REG[00h] Software Reset Register (SRR)

Bit	Description	Default	Access
7	<b>Reconfigure PLL frequency</b> Write “1” to this bit will reconfigure PLL frequency. <b>Note:</b> 1. When users change PLL relative parameters, PLL clock will not be changed immediately, users must set this bit as “1” again. 2. Users may read (check) this bit to find out whether PLL clock is changed or not.	0	RW
6-1	Reserved	0	RO
0	<b>Software Reset (Reconfigure PLL Frequency)</b> 0: Normal Operation. 1: Software Reset. The bit will be auto cleared after reset. Software Reset only resets internal state machine. Configuration Registers value will not be reset. Therefore, all read-only flag in the register will return to its initial value. Users should actively make sure the flag is in desired state.	0	WO
0	<b>Warning Condition Flag</b> 0: No warning operation occurred 1: Warning condition occurred. Please refer to REG[E4h] bit3 description.	0	RO

REG[01h] Chip Configuration Register (CCR)

Bit	Description	Default	Access
7	<b>Check PLL Ready</b> Host may read (check) this bit to find out whether PLL clock is ready or not. Read “1” means PLL clock is ready and switched successfully.	0	RW
6	<b>Mask WAIT# on CS# De-assert</b> 0: No Mask, WAIT# keep assert if internal state keep busy and cannot accept next R/W cycle, no matter CS# assert/de-assert. If Host cycle cannot be extended while WAIT# keeps low, Host program should poll WAIT# and wait for it to become high, then start next access. 1: Mask, WAIT# de-assert when CS# de-assert. Used to automatically extend Host cycle by Wait#.	1	RW
5	<b>Keypad-scan Enable/Disable</b> 0: Disable 1: Enable	0	RW

Bit	Description	Default	Access
4-3	<b>TFT Panel I/F Setting</b> 00b: 24bits TFT Output 01b: 18bits TFT Output 10b: 16bits TFT Output 11b: Without TFT Output Other unused TFT output pins are set as GPIO or Key-Scan function.	01b	RW
2	<b>I2C Master Interface Enable/Disable</b> 0: Disable (GPIO Function) 1: Enable (I2C Master Function) This bit has higher priority than Keypad-scan Enable bit. i.e. if I2C master and Keypad-scan are enabled simultaneously, then KI[0] and KO[0] will become I2C function & other KI/KO pins still keep Keypad-scan function.	0	RW
1	<b>Serial Flash or SPI Interface Enable/Disable</b> 0: Disable (GPIO Function) 1: Enable (SPI Master Function)	0	RW
0	<b>Host Data Bus Width Selection</b> 0: 8bits Data Bus 1: 16bits Data Bus If Serial Host I/F is selected or if the system is in initial display operation period, LT768x will force this bit as 0 and only allow 8bits access.	0	RW

## REG[02h] Memory Access Control Register (MACR)

Bit	Description	Default	Access
7-6	<b>Host Read/Write Image Data Format</b> 0xb: Direct Write, for below I/F format: 1. 8bits MCU I/F. 2. 16bits MCU I/F with 8bpp data mode 1 & 2. 3. 16bits MCU I/F with 16/24bpp data mode 1. 4. Serial SPI/I2C I/F. 10b: Mask high byte of each data (ex. 16 bit MPU I/F with 8-bpp data mode 1) 11b: Mask high byte of even data (ex. 16 bit MPU I/F with 24-bpp data mode 2)	0	RW
5-4	<b>Host Read Memory Direction (Only for Graphic Mode)</b> 00b: Left→Right then Top→Bottom. 01b: Right→Left then Top→Bottom. 10b: Top→Bottom then Left→Right. 11b: Bottom→Top then Left→Right. Ignored if canvas in linear addressing mode.	0	RW
3	NA	NA	NA
2-1	<b>Host Write Memory Direction (Only for Graphic Mode)</b> 00b: Left→Right then Top→Bottom (Original) 01b: Right → Left then Top → Bottom (Horizontal Flip) 10b: Top → Bottom then Left → Right (Rotate right 90° & Horizontal flip) 11b: Bottom → Top then Left → Right (Rotate left 90°) Ignored if canvas in linear addressing mode.	0	RW
0	NA (must keep it as 0)	0	RO

## REG[03h] Input Control Register (ICR)

Bit	Description	Default	Access
7	<b>Interrupt Pin Active Level</b> 0: Low Active 1: High Active	0	RW
6	<b>External Interrupt Signal - PSM[0] Pin De-bounce</b> 0: Without De-bounce 1: Enable De-bounce (1,024 OSC Clock)	0	RW
5-4	<b>External Interrupt Signal - PSM[0] Pin Trigger Type</b> 00b: low level trigger 01b: falling edge trigger 10b: high level trigger 11b: rising edge trigger	00b	RW
3	NA	NA	NA
2	<b>Text Mode Enable</b> 0: Graphic mode 1: Text mode  Before toggling this bit, users must make sure core task busy bit in status register is busy or idle. This bit is always 0 (in graphic mode) if canvas' address mode is linear mode.	0	RW
1-0	<b>Memory Port Read/Write Destination Selection</b> 00b: Image buffer (Display RAM) for image data, pattern, user-characters. Support Read-modify-Write. 01b: Gamma table for Color Red/Green/Blue. Each color's gamma table has 256 bytes. Users need to specify desired gamma table and continuously write 256 bytes. 10b: Graphic Cursor RAM (only accept low 8-bits MPU data, similar to normal register data r/w.), "Graphic Cursor RAM read" is not supported. It contains 4 graphic cursor sets. Each set has 128x16 bits. Users need to specify target graphic cursor set and continuously write 256 bytes. 11b: Color palette RAM. It is 64x12 bits SRAM. Since MCU writes 8 bits data at a time, when it writes every even byte, only the lower 4bits will be stored to the RAM. "Color palette RAM read" is not supported. Users need to continuously write 128 bytes.	0	RW

## REG[04h] Memory Data Read/Write Port (MRWDP)

Bit	Description	Default	Access
7-0	<p><b>Write Function: Memory Write Data</b>  Data to write in memory corresponding to the setting of REG[03h][1:0]. Continuous data write cycle can be accepted in case of writing great amount of data.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>A. Image data in Display RAM: Set to 8/16-bits based on the MCU I/F bit width. Host R/W image data format can be set. Canvas color depth and related settings can also be set in block mode.</li> <li>B. Pattern data for BTE operation in Display RAM: Set to 8/16-bits based on the MCU I/F bit width. Host R/W image data format can be set. Canvas color depth and related settings can also be set in block mode. Active window's width and height should be set as 8x8 or 16x16 depending on users needs.</li> <li>C. User-characters in Display RAM: Set to 8/16-bits based on the MCU I/F bit width. Host R/W image data format can be set, and canvas is set to linear mode.</li> <li>D. Character code: only low 8-bits MCU data are accepted, similar to normal register R/W. For two bytes character code, input high byte first. For user defined Character, code &lt; 8000h is half size, code &gt;= 8000h is full size.</li> <li>E. Gamma table data: only low 8-bits MPU data are accepted. Users must set "Select Gamma table sets([3Ch] Bit6-5)" to clear internal Gamma table's address counter then start to write data. Users should program 256 bytes data to memory data port.</li> <li>F. Graphic Cursor RAM data: only low 8-bits MPU data are accepted. Users must set "Select Graphic Cursor sets" bits to clear internal Graphic Cursor RAM address counter then start to write data.</li> <li>G. Color palette RAM data: only low 8-bits MPU data are accepted. Users must program full Color palette RAM in a continuous 128 byte data, and write to memory data port, and cannot change register address during the writing process.</li> </ul> <p><b>Read Function: Memory Read Data</b></p> <p>To read data from memory, REG[03h][1:0] has to be set. Continuous data read cycle can be accepted in case of reading great amount of data.</p> <p><b>Note1:</b> when reading different port address, a dummy read must be issued. The first data read cycle is dummy read and data should be ignored. Graphic Cursor RAM &amp; Color palette RAM data do not support data read function.</p> <p><b>Note2:</b> No matter what the color depth setting is, data reading is based on 4 bytes.</p> <p><b>Note3:</b> If users want to change the address of the written registers, but there are data already written to Display RAM, users must make sure  (1) write FIFO is empty before changing register number; (2) Core task busy status bit becomes idle.</p>	--	RW

### 19.3 PLL Setting Register

REG[05h] PCLK PLL Control Register 1 (PPLLC1)

Bit	Description	Default	Access
7-6	<b>PCLK Output Divider Ratio, OD[1:0]</b> 00b: Divided by 1. 01b: Divided by 2. 10b: Divided by 3. 11b: Divided by 4.	0	RW
5-1	<b>PCLK Input Divider Ratio, R[4:0]</b> The value should be 2~31.	10	RW
0	<b>PCLK Feedback Divider Ratio of Loop, N[8]</b> Total 9 bits, the value should be 2~511..	0	RW

REG[06h] PCLK PLL Control Register 2 (PPLLC2)

Bit	Description	Default	Access
7-0	<b>PCLK PLLDIVN[7:0]</b> Total 9 bits, the value should be 2~511.	60	RW

**Note:** PCLK is used by TFT panel's scan clock and derived from CCLK.

REG[07h] MCLK PLL Control Register 1 (MPLLC1)

Bit	Description	Default	Access
7-6	<b>MCLK output divider Ratio, OD[1:0]</b> 00b: Divided by 1. 01b: Divided by 2. 10b: Divided by 3. 11b: Divided by 4.	0	RW
5-1	MCLK Input Divider Ratio, R[4:0] The value should be 2~31.	10	RW
0	MCLK Feedback Divider Ratio of Loop, N[8] Total 9 bits, the value should be 2~511.	0	RW

REG[08h] MCLK PLL Control Register 2 (MPLLC2)

Bit	Description	Default	Access
7-0	<b>MCLK PLLDIVN[7:0]</b> Total 9 bits, the value should be 2~511.	133	RW

**Note:** MCLK is used by internal Display RAM's clock.

## REG[09h] CCLK PLL Control Register 1 (CPLLC1)

Bit	Description	Default	Access
7-6	<b>CCLK Output Divider Ratio, OD[1:0]</b> 00b: Divided by 1. 01b: Divided by 2. 10b: Divided by 3. 11b: Divided by 4.	0	RW
5-1	<b>CCLK Input Divider Ratio, R[4:0]</b> The value should be 2~31.	10	RW
0	<b>CCLK Feedback Divider Ratio of Loop, N[8]</b> Total 9 bits, the value should be 2~511.	0	RW

## REG[0Ah] CCLK PLL Control Register 2 (CPLLC2)

Bit	Description	Default	Access
7-0	<b>CCLK PLLDIVN[7:0]</b> Total 9 bits, the value should be 2~511.	133	RW

**Note1:** CCLK is used by Core's Clock.

**Note2:** PLL output frequency is calculated from the following the equation:

$$F_{OUT} = X_I * (N/R) \div OD$$

The input frequency  $X_I/R$  is no less than 1MHz. For example,  $X_I = 10MHz$ ,  $R[4:0] = 01010$ ,  $N[8:0] = 100000000$ ,  $OD[1:0] = 11$ , then

$$F_{OUT} = 10MHz * (256 / 10) \div 4 = 64 MHz$$

## 19.4 Interrupt Control Register

**REG[0Bh] Interrupt Enable Register (INTEN)**

Bit	Description	Default	Access
7	<b>Wakeup/Resume Interrupt Enable</b> 0: Disable 1: Enable	0	RW
6	<b>External Interrupt Input - PSM[0] Enable)</b> 0: Disable 1: Enable	0	RW
5	<b>I2C Master Interrupt Enable</b> 0: Disable 1: Enable	0	RW
4	<b>VSYNC Time Base Interrupt Enable</b> 0: Disable Interrupt 1: Enable Interrupt This interrupt is to inform MCU that the VSYNC of the LCD has tearing effect.	0	RW
3	<b>Keypad-scan Interrupt Enable</b> 0: Disable Interrupt 1: Enable Interrupt	0	RW
2	<b>Serial Flash DMA Complete / Draw Task Finished / BTE Process Complete etc. Interrupt Enable</b> 0: Disable Interrupt 1: Enable Interrupt	0	RW
1	<b>PWM Timer-1 Interrupt Enable</b> 0: Disable Interrupt 1: Enable Interrupt	0	RW
0	<b>PWM Timer-0 Interrupt Enable</b> 0: Disable Interrupt 1: Enable Interrupt	0	RW

## REG[0Ch] Interrupt Event Flag Register (INTF)

Bit	Description	Default	Access
7	<b>Wakeup/Resume Interrupt Flag</b> <b>Write:</b> Clear Interrupt Flag 0: No Operation 1: Clear Wakeup/Resume Interrupt Flag <b>Read:</b> Read Interrupt Flag 0: No Wakeup/Resume Interrupt Happens 1: Wakeup/Resume Interrupt Happens	0	RW
6	<b>External Interrupt Input - PSM[0] Flag</b> <b>Write:</b> Clear PSM[0] Pin Interrupt Flag 0: No Operation 1: Clear PSM[0] Interrupt Flag <b>Read:</b> Read PSM[0] Pin Interrupt Flag 0: No PSM[0] Interrupt Happens 1: PSM[0] Interrupt Happens	0	RW
5	<b>I2C Master Interrupt Flag</b> <b>Write:</b> Clear I2C Master Interrupt Flag 0: No Operation 1: Clear I2C Master Interrupt Flag <b>Read:</b> Read I2C Master Interrupt Flag 0: No I2C Master Interrupt Happens 1: I2C Master Interrupt Happens	0	RW
4	<b>VSYNC Time Base Interrupt Flag</b> <b>Write:</b> Clear VSYNC Interrupt Flag 0: No Operation 1: Clear VSYNC Interrupt Flag <b>Read:</b> Read VSYNC Interrupt Flag 0: No VSYNC Interrupt Happens 1: VSYNC Interrupt Happens	0	RW
3	<b>Keypad-scan Interrupt Flag</b> <b>Write:</b> Clear Keypad-scan Interrupt Flag 0: No Operation 1: Clear Keypad-scan Interrupt Flag <b>Read:</b> Read Keypad-scan Interrupt Flag 0: No Keypad-scan Interrupt Happens 1: Keypad-scan Interrupt Happens	0	RW

Bit	Description	Default	Access
2	<b>Serial Flash DMA Complete   Draw Task Finished   BTE Process Complete etc. Interrupt Flag</b> <b>Write:</b> Clear Process Complete Interrupt Flag 0: No Operation 1: Clear Interrupt Flag <b>Read:</b> Read Process Complete Interrupt Flag 0: No Interrupt Happens 1: Interrupt Happens	0	RW
1	<b>PWM1 Timer Interrupt Flag</b> <b>Write:</b> Clear PWM1 Interrupt Flag 0: No Operation 1: Clear PWM1 Interrupt Flag <b>Read:</b> Read PWM1 Interrupt Flag 0: No PWM1 Interrupt Happens 1: PWM1 Interrupt Happens	0	RW
0	<b>PPWM0 Timer Interrupt Flag</b> <b>Write:</b> Clear PWM0 Interrupt Flag 0: No Operation 1: Clear PWM0 Interrupt Flag <b>Read:</b> Read PWM0 Interrupt Flag 0: No PWM0 Interrupt Happens 1: PWM0 Interrupt Happens	0	RW

**Note:** If the Host receives interruption, but the flag of this register shows no interruption happened, then Host should confirm SPI Master State Register Interrupt Flag of REG[BAh].

#### REG[0Dh] Mask Interrupt Flag Register (MINTFR)

Bit	Description	Default	Access
7	<b>Mask Wakeup/Resume Interrupt Flag</b> 0: Unmask 1: Mask	0	RW
6	<b>External Interrupt Input - PSM[0] Flag</b> 0: Unmask 1: Mask	0	RW
5	<b>I2C Master Interrupt Flag</b> 0: Unmask 1: Mask	0	RW
4	<b>VSYNC Time Base Interrupt Flag</b> 0: Unmask 1: Mask	0	RW
3	<b>Keypad-scan Interrupt Flag</b> 0: Unmask 1: Mask	0	RW

Bit	Description	Default	Access
2	<b>Serial Flash DMA Complete   Draw Task Finished   BTE Process Complete etc. Interrupt Flag</b> 0: Unmask 1: Mask	0	RW
1	<b>PWM1 Timer Interrupt Flag</b> 0: Unmask 1: Mask	0	RW
0	<b>PWM0 Timer Interrupt Flag</b> 0: Unmask 1: Mask	0	RW

**Note:** If the Host masks the interrupt flag, then LT768x will not send interruption to the Host. If only some unmasked interrupt flags are used, Host can check interrupt flag to find out whether there is an interrupt.

#### REG[0Eh] Pull-High Control Register (PUENR)

Bit	Description	Default	Access
7-6	NA.	NA	NA
5	<b>GPIO-F[7:0] Pull-High Enable</b> 0: without Pull-up Resistor 1: with Pull-up	0	RW
4	<b>GPIO-E[7:0] Pull-High Enable</b> 0: without Pull-up Resistor 1: with Pull-up	0	RW
3	<b>GPIO-D[7:0] Pull-High Enable</b> 0: without Pull-up Resistor 1: with Pull-up	0	RW
2	<b>GPIO-C[4:0] Pull-High Enable</b> 0: without Pull-up Resistor 1: with Pull-up	0	RW
1	<b>DB[15:8] Pull- High Enable</b> 0: without Pull-up Resistor 1: with Pull-up	0	RW
0	<b>DB[7:0] Pull- High Enable</b> 0: without Pull-up Resistor 1: with Pull-up	0	RW

**Note:** These bits are available only when GPIO function is enabled.

REG[0Fh] PD for GPIO/Key Function Select Register (PSFSR)

Bit	Description	Default	Access
7	<b>PD[18] – Function Select</b> 0: GPIO-D7 1: KO[4]	0	RW
6	<b>PD[17] – Function Select</b> 0: GPIO-D5 1: KO[2]	0	RW
5	<b>PD[16] – Function Select</b> 0: GPIO-D4 1: KO[1]	0	RW
4	<b>PD[9] – Function Select</b> 0: GPIO-D3 1: KO[3]	0	RW
3	<b>PD[8] – Function Select</b> 0: GPIO-D2 1: KI[3]	0	RW
2	<b>PD[2] – Function Select</b> 0: GPIO-D6 1: KI[4]	0	RW
1	<b>PD[1] – Function Select</b> 0: GPIO-D1 1: KI[2]	0	RW
0	<b>PD[0] – Function Select</b> 0: GPIO-D0 1: KI[1]	0	RW

Some of the LCD Data pins are shared with GPIO and Keypad-scan pins. If LCD I/F is not 24bpp mode, then PD[18:16 / 8:9 / 2:0] can be defined as GPIO or Keypad-scan pins.

## 19.5 LCD Display Control Registers

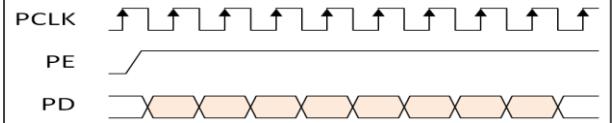
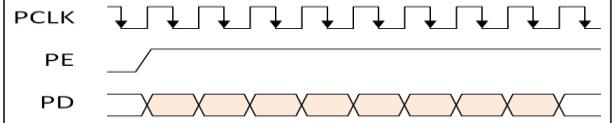
**REG[10h] Main/PIP Window Control Register (MPWCTR)**

Bit	Description	Default	Access
7	<b>PIP-1 Window Enable/Disable</b> 0: PIP-1 Window Disable 1: PIP-1 Window Enable  PIP-1 window always on top of PIP-2 window.	0	RW
6	<b>PIP-2 Window Enable/Disable</b> 0: PIP-2 Window Disable 1: PIP-2 Window Enable  PIP-1 window always on top of PIP-2 window	0	RW
5	NA	NA	NA
4	<b>Select Configure PIP 1 or 2 Window's parameters</b> PIP window's parameter including Color Depth, Starting Address, Image Width, Display Coordinates, Window Coordinates, Window Width and Window Height. 0: To configure PIP 1's parameters. 1: To configure PIP 2's parameters.	0	RW
3-2	<b>Main Image Color Depth Setting</b> 00b: 8bpp Generic TFT (256 color) 01b: 16bpp Generic TFT (65K color) 1xb: 24bpp Generic TFT (16.7M color)	1	RW
1	NA	0	RW
0	<b>To Control Panel's Synchronous Signals</b> 0: Sync Mode → Enable VSYNC, HSYNC, DE 1: DE Mode → Only DE enable, VSYNC & HSYNC in idle state.	0	RW

**REG[11h] PIP Window Color Depth Setting (PIPCDEP)**

Bit	Description	Default	Access
7-4	NA	0	RO
3-2	<b>PIP-1 Window Color Depth Setting</b> 00b: 8bpp Generic TFT (256 color) 01b: 16bpp Generic TFT (65K color) 1xb: 24bpp Generic TFT (16.7M color)	1	RW
1-0	<b>PIP-2 Window Color Depth Setting</b> 00b: 8bpp Generic TFT (256 color) 01b: 16bpp Generic TFT (65K color) 1xb: 24bpp Generic TFT (16.7M color)	1	RW

## REG[12h] Display Configuration Register (DPCR)

Bit	Description	Default	Access
7	<b>PCLK Inversion</b> 0: TFT Panel fetches PD at PCLK rising edge.  1: TFT Panel fetches PD at PCLK falling edge. 	0	RW
6	<b>Display ON/OFF</b> 0b: Display Off 1b: Display On	0	RW
5	<b>Display Test Color Bar</b> 0b: Disable 1b: Enable	0	RW
4	<b>This bit must be 0.</b>	0	RW
3	<b>VDIR: Vertical Scan Direction</b> 0: From Top to Bottom 1: From bottom to Top	0	RW
2-0	<b>Parallel PD[23:0] Output Sequence</b> 000b: RGB 001b: RBG 010b: GRB 011b: GBR 100b: BRG 101b: BGR 110b: Gray 111b: Send out Idle State. All data are 0 (Black) or 1(White). This option has to setup with REG[13h].	0	RW

**Note:** When VDIR = 1, PIP window, Graphic Cursor and Text Cursor function will be disabled automatically.

## REG[13h] Panel Scan Clock and Data Setting Register (PCSR)

Bit	Description	Default	Access
7	<b>Hsync Polarity</b> 0: Low Active 1: High Active	0	RW
6	<b>Vsync Polarity</b> 0: Low Active 1: High Active	0	RW
5	<b>PDE Polarity</b> 0: High Active 1: Low Active	0	RW
4	<b>PDE Idle State</b> 0: Pin "PDE" output Low 1: Pin "PDE" output High This is used to setup the PDE output status in Power Saving mode or Display Off.	0	RW
3	<b>PCLK Idle State</b> 0: Pin "PCLK" output Low 1: Pin "PCLK" output High This is used to setup the PCLK output status in Power Saving mode or Display Off.	0	RW
2	<b>PD Idle State</b> 0: Pins "PD[23:0]" output Low 1: Pins "PD[23:0]" output High This is used to setup the PD output status in Vertical/Horizontal Non-Display Period, Power Saving mode or Display Off.	0	RW
1	<b>Hsync Idle State</b> 0: Pin "Hsync" output Low 1: Pin "Hsync" output High This is used to setup the Hsync output status in Power Saving mode or Display Off.	1	RW
0	<b>Vsync Idle State</b> 0: Pin "Vsync" output Low 1: Pin "Vsync" output High This is used to setup the Vsync output status in Power Saving mode or Display Off.	1	RW

**Note:** The sum of (HST + HPW + HND) had better be larger than 64Pixels to prevent scanning FIFO empty. If both PIP1 and PIP2 are enabled and are very close to the left side of the window, there is only a small change in PIP1 and PIP2's UL-X. Please refer to Figure 9-1 TFT-LCD interface Timing diagram.

## REG[14h] Horizontal Display Width Register (HDWR)

Bit	Description	Default	Access
7-0	<p><b>Horizontal Display Width Setting</b>  This register is used to set a horizontal display width. The specified LCD screen resolution is 8 pixels in one unit resolution.</p> <p style="color: blue;"><b>Horizontal Display Width (pixels)  = (HDWR + 1)* 8 + HDWFTR</b></p> <p>HDWFTR (REG[15h]) is the fine-tuning value for Horizontal Display Width. Each fine-tuning resolution is 1 pixel, and the maximum horizontal width is 2,048 pixels.</p>	4Fh	RW

## REG[15h] Horizontal Display Width Fine Tune Register (HDWFTR)

Bit	Description	Default	Access
7-4	NA	NA	NA
3-0	<p><b>Horizontal Display Width Fine Tuning</b>  This Register is the fine-tuning value for Horizontal Display Width, and each fine-tuning resolution is 1 pixel.</p>	0	RW

## REG[16h] Horizontal Non-Display Period Register (HNDR)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	<p><b>Horizontal Non-Display Period</b>  This register assigns the Period of Horizontal Non-Display. It is also called "Back Porch".</p> <p style="color: blue;"><b>Horizontal Non-Display Period (Pixels)  = (HNDR + 1)* 8 + HNDFTR</b></p> <p>HNDFTR (REG[17h]) is the fine-tuning value for Horizontal Non-display Period. Each fine-tuning resolution is 1 pixel, and the maximum horizontal width is 2,048 pixels.</p>	03h	RW

## REG[17h] Horizontal Non-Display Period Fine Tune Register (HNDFTR)

Bit	Description	Default	Access
7-4	NA	0	RO
3-0	<p><b>Horizontal Non-Display Period Fine Tuning</b>  This Register is the fine-tuning value for Horizontal Non-display Period, and each fine-tuning resolution is 1 pixel. it is used to support the SYNC mode panel.</p>	06h	RW

## REG[18h] HSYNC Start Position Register (HSTR)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	<b>HSYNC Start Position</b> This register specifies the starting address of the HSYNC. The starting point for the calculation is the point at which the end of the display area is to start producing HSYNC. The basic unit of each adjustment is 8 pixels. It's also called "Front Porch". $\text{HSYNC Start Position} = (\text{HSTR} + 1) * 8$	1Fh	RW

## REG[19h] HSYNC Pulse Width Register (HPWR)

Bit	Description	Default	Access
7-5	NA	NA	NA
4-0	<b>HSYNC Pulse Width</b> $\text{HSYNC Pulse Width (Pixels)} = (\text{HPW} + 1) * 8$	0	RW

## REG[1Ah-1Bh] Vertical Display Height Register (VDHR)

Bit	Description	Default	Access
7-0	<b>Vertical Display Height</b> REG[1Ah] mapping to VDHR [7:0] REG[1Bh] bit[2:0] mapping to VDHR [10:8], bit[7:3] are not used. The height of the vertical display is in the unit of line, and the formula is as following: $\text{Vertical Display Height (Line)} = \text{VDHR} + 1$	DFh	RW

## REG[1Ch-1Dh] Vertical Non-Display Period Register (VNDR)

Bit	Description	Default	Access
7-0	<b>Vertical Non-Display Period</b> REG[1Ch] mapping to VNDR [7:0] REG[1Dh] bit[1:0] mapping to VNDR [9:8], REG[1Dh] bit[7:2] are not used. The formula is as following: $\text{Vertical Non-Display Period (Line)} = (\text{VNDR} + 1)$	15h	RW

## REG[1Eh] VSYNC Start Position Register (VSTR)

Bit	Description	Default	Access
7-0	<b>VSYNC Start Position</b> The starting position is from the end of display area to the beginning of VSYNC.  <b>VSYNC Start Position (Line)= (VSTR + 1)</b>	0Bh	RW

## REG[1Fh] VSYNC Pulse Width Register (VPWR)

Bit	Description	Default	Access
7-6	NA	0	RO
5-0	<b>VSYNC Pulse Width</b>  <b>VSYNC Pulse Width (Line)= (VPWR + 1)</b>	0	RW

## REG[23h-20h] Main Image Start Address (MISA)

Bit	Description	Default	Access
7-0	<b>Main Image Start Address</b> REG[20h] mapping to MISA[7:0], bit[1:0] must be 0. REG[21h] mapping to MISA[15:8] REG[22h] mapping to MISA[23:16] REG[23h] mapping to MISA[31:24]	0	RW

## REG[25h-24h] Main Image Width (MIW)

Bit	Description	Default	Access
7-0	<b>Main Image Width</b> REG[24h] mapping to MIW[7:0], bit[1:0] must be 0. REG[25h] bit[4:0] mapping to MIW[12:8], bit[7:5] are not used..  The unit is pixel, which is the pixel that represents the horizontal width of the actual LCD. The maximum setting is 8,192 pixels.	0	RW

## REG[27h-26h] Main Window Upper-Left Corner X-Coordinates (MWULX)

Bit	Description	Default	Access
7-0	<b>Main Window Upper-Left Corner X-Coordinates</b> REG[26h] mapping to MWULX[7:0], bit[1:0] must be 0. REG[27h] bit[4:0] mapping to MWULX [12:8], bit[7:5] are not used..  The unit is pixel. The sum of X-Coordinates plus Horizontal Display Width cannot be greater than 8,191.	0	RW

**REG[29h-28h] Main Window Upper-Left corner Y-Coordinates (MWULY)**

Bit	Description	Default	Access
7-0	<b>Main Window Upper-Left Corner Y-Coordinates</b> REG[28h] mapping to MWULY[7:0] REG[29h] bit[4:0] mapping to MWULY [12:8], bit[7:5] are not used. Unit: Pixel. The range is between 0 and 8,191.	0	RW

**REG[2Bh-2Ah] PIP Window 1 or 2 Display Upper-Left Corner X-Coordinates (PWDULX)**

Bit	Description	Default	Access
7-0	<b>PIP Window Display Upper-Left Corner X-Coordinates</b> REG[2Ah] mapping to PWDULX[7:0], bit[1:0] must be 0. REG[2Bh] bit[4:0] mapping to PWDULX[12:8], bit[7:5] are not used. Unit: Pixel. X-axis coordinates should be less than the horizontal display width. This setting is relative to the PIP settings in the register of REG[10h].	0	RW

**REG[2Dh-2Ch] PIP Window 1 or 2 Display Upper-Left corner Y-Coordinates (PWDULY)**

Bit	Description	Default	Access
7-0	<b>PIP Window Display Upper-Left Corner Y-Coordinates</b> REG[2Ch] mapping to PWDULX[7:0] REG[2Dh] bit[4:0] mapping to PWDULX[12:8], bit[7:5] are not used. Y-axis coordinates should be less than the vertical display height. This setting is relative to the PIP settings in the register of REG[10h].	0	RW

**REG[31h-2Eh] PIP Image 1 or 2 Start Address (PISA)**

Bit	Description	Default	Access
7-0	<b>PIP Image Start Address</b> REG[2Eh] mapping to PISA[7:0], bit[1:0] must be 0. REG[2Fh] mapping to PISA [15:8] REG[30h] mapping to PISA [23:16] REG[31h] mapping to PISA [31:24] This setting is relative to the PIP settings in the register of REG[10h].	0	RW

## REG[33h-32h] PIP Image 1 or 2 Width (PIW)

Bit	Description	Default	Access
7-0	<p><b>PIP Image Width</b></p> <p>REG[32h] mapping to PIW[7:0], bit[1:0] must be 0.</p> <p>REG[33h] bit[5:0] mapping to PIW[13:8], bit[7:6] are not used.</p> <p>Unit: Pixel. The value is physical width, and maximum value is 8192 pixels.</p> <p>This width should be less than horizontal display width. This setting is relative to the PIP settings in the register of REG[10h].</p>	0	RW

## REG[35h-34h] PIP Window Image 1 or 2 Upper-Left Corner X-Coordinates (PWIULX)

Bit	Description	Default	Access
7-0	<p><b>PIP Window 1 or 2 Image Upper-Left Corner X-Coordinates</b></p> <p>REG[34h] mapping to PWIULX[7:0], bit[1:0] must be 0.</p> <p>REG[35h] bit[4:0] mapping to PWIULX[12:8], bit[7:5] are not used.</p> <p>Unit: Pixel. The sum of X-axis coordinates and PIP image width must be less than or equal to 8,191.</p> <p>This setting is relative to the PIP settings in the register of REG[10h].</p>	0	RW

## REG[37h-36h] PIP Window Image 1 or 2 Upper-Left Corner Y-Coordinates (PWIULY)

Bit	Description	Default	Access
7-0	<p><b>PIP Windows Display Upper-Left Corner Y-Coordinates</b></p> <p>REG[36h] mapping to PWIULY[7:0], bit[1:0] must be 0.</p> <p>REG[37h] bit[4:0] mapping to PWIULY[12:8], bit[7:5] are not used.</p> <p>Unit: Pixel. The sum of Y-axis coordinates and PIP window height should be less than or equal to 8,191.</p> <p>This setting is relative to the PIP settings in the register of REG[10h].</p>	0	RW

## REG[39h-38h] PIP Window 1 or 2 Width (PWW)

Bit	Description	Default	Access
7-0	<p><b>PIP Window Width</b></p> <p>REG[38h] mapping to PWW[7:0], bit[1:0] must be 0.</p> <p>REG[39h] bit[5:0] mapping to PWW[13:8], bit[7:6] are not used.</p> <p>Unit: Pixel. Maximum value is 8,192 pixels.</p> <p>This setting is relative to the PIP settings in the register of REG[10h].</p>	0	RW

**REG[3Bh-3Ah] PIP Window 1 or 2 Height (PWH)**

Bit	Description	Default	Access
7-0	<b>PIP Window Height</b> REG[3Ah] mapping to PWH[7:0], bit[1:0] must be 0. REG[3Bh] bit[5:0] mapping to PWH[13:8], bit[7:6] are not used. Unit: Pixel. The value is physical pixel number and maximum value is 8,192 pixels. This setting is relative to the PIP settings in the register of REG[10h].	0	RW

**Note 1:** The resolution of Window Size and Starting Position in the horizontal direction is 8 pixels, the vertical resolution is 1 line.

**Note 2:** The above registers REG[20h] ~ REG[3Bh] need to be written from LSB to MSB in turn. Suppose we need to set the Main Image Start Address, this relative register are REG[20h] to REG[23h]. Then Host must write Address data from LSB (REG[20h]) to MSB (REG[23h]) in turn. When REG[23h] is written, LT768x will then write the complete Main Image Start Address to the internal register.

**REG[3Ch] Graphic / Text Cursor Control Register (GTCCR)**

Bit	Description	Default	Access
7	<b>Gamma Correction Enable</b> 0: Disable 1: Enable Gamma correction is the last output stage.	0	RW
6-5	<b>Gamma Table Select for Host Write Gamma Data</b> 00b: Gamma table for Blue 01b: Gamma table for Green 10b: Gamma table for Red 11b: NA	0	RW
4	<b>Graphic Cursor Enable</b> 0: Graphic Cursor Disable 1: Graphic Cursor Enable Graphic cursor will auto disable if VDIR (REG[12h] bit3) set as "1".	0	RW
3-2	<b>Graphic Cursor Selection</b> Select one from four graphic cursor types. (00b to 11b) 00b: Graphic Cursor Set 1 01b: Graphic Cursor Set 2 10b: Graphic Cursor Set 3 11b: Graphic Cursor Set 4	0	RW

Bit	Description	Default	Access
1	<b>Text Cursor Enable</b> 0: Disable 1: Enable <b>Note:</b> Text cursor & Graphic cursor cannot be enabled simultaneously. Graphic cursor has higher priority than Text cursor if enabled simultaneously.	0	RW
0	<b>Text Cursor Blinking Enable</b> 0: Disable 1: Enable	0	RW

REG[3Dh] Blink Time Control Register (BTCR)

Bit	Description	Default	Access
7-0	<b>Text Cursor Blink Time Setting</b> 00h: 1 Frame Cycle Time 01h: 2 Frames Cycle Time 02h: 3 Frames Cycle Time : : FFh: 256 frames Cycle Time	0	RW

REG[3Eh] Text Cursor Horizontal Size Register (CURHS)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	<b>Text Cursor Horizontal Size Setting</b> 00000b: 1 Pixel 00001b: 2 Pixels : : 11111b: 32 Pixels <b>Note:</b> When the character is enlarged, the cursor will be enlarged at the same time.	07h	RW

REG[3Fh] Text Cursor Vertical Size Register (CURVS)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	<b>Text Cursor Vertical Size Setting</b> Unit: Pixel, Zero-based number. Value "0" means 1 pixel. <b>Note:</b> When the character is enlarged, the cursor will be enlarged at the same time.	0	RW

**REG[40h-41h] Graphic Cursor Horizontal Position Register (GCHP)**

Bit	Description	Default	Access
7-0	<b>Graphic Cursor Horizontal Position</b> REG[40h] mapping to GCHP[7:0] REG[41h] bit[4:0] mapping to GCHP[12:8], bit[7:5] are not used.	0	RW

**REG[42h-43h] Graphic Cursor Vertical Position Register (GCVP)**

Bit	Description	Default	Access
7-0	<b>Graphic Cursor Vertical Position</b> REG[42h] mapping to GCVP[7:0] REG[43h] bit[4:0] mapping to GCVP[12:8], bit[7:5] are not used.	0	RW

**REG[44h] Graphic Cursor Color 0 (GCC0)**

Bit	Description	Default	Access
7-0	<b>Graphic Cursor Color 0 with 256 Colors</b> RGB Format [7:0] = <b>RRRGGBB</b>	0	RW

**REG[45h] Graphic Cursor Color 1 (GCC1)**

Bit	Description	Default	Access
7-0	<b>Graphic Cursor Color 1 with 256 Colors</b> RGB Format [7:0] = <b>RRRGGBB</b>	0	RW

## 19.6 Geometric Engine Control Registers

**REG[53h-50h] Canvas Start Address (CVSSA)**

Bit	Description	Default	Access
7-0	<b>Start Address of Canvas</b> REG[50h] mapping to CVSSA[7:0], bit[1:0] must be 0. REG[51h] mapping to CVSSA[15:8] REG[52h] mapping to CVSSA[23:16] REG[53h] mapping to CVSSA[31:24] These Registers will be ignored if canvas is in linear Addressing mode.	0	RW

**REG[55h-54h] Canvas Image Width (CVS\_IMWTH)**

Bit	Description	Default	Access
7-0	<b>Canvas Image Width</b> REG[54h] mapping to CVS_IMWTH[7:0], bit[1:0] must be 0. REG[55h] bit[5:0] mapping to CVS_IMWTH[13:8], bit[7:6] are not used. Width = Real Image Width These Registers will be ignored if canvas is in linear Addressing mode.	0	RW

**Note:** The unit of REG[54h] ~ REG[5Dh] is Pixel.

**REG[57h-56h] Active Window Upper-Left Corner X-Coordinates (AWUL\_X)**

Bit	Description	Default	Access
7-0	<b>Active Window Upper-Left Corner X-Coordinates</b> REG[56h] mapping to AWUL_X[7:0] REG[57h] bit[4:0] mapping to AWUL_X[12:8], bit[7:5] are not used. Unit: Pixel. The sum of X-axis coordinates and Active Window width cannot be larger than 8,191. These Registers will be ignored if canvas is in linear Addressing mode.	0	RW

**REG[59h-58h] Active Window Upper-Left Corner Y-Coordinates (AWUL\_Y)**

Bit	Description	Default	Access
7-0	<b>Active Window Upper-Left Corner Y-Coordinates</b> REG[58h] mapping to AWUL_Y[7:0] REG[59h] bit[4:0] mapping to AWUL_Y[12:8], bit[7:5] are not used. Unit: Pixel. The sum of Y-axis coordinates and Active Window height cannot be larger than 8,191. These Registers will be ignored if canvas is in linear Addressing mode.	0	RW

**REG[5Bh-5Ah] Active Window Width (AW\_WTH)**

Bit	Description	Default	Access
7-0	<b>Active Window Width [13:8]</b> REG[5Ah] mapping to AW_WTH[7:0] REG[5Bh] bit[5:0] mapping to AW_WTH[13:8], bit[7:6] are not used. Unit: Pixel. The value is physical pixel width. Maximum pixel width is 8,192. These Registers will be ignored if canvas is in linear Addressing mode.	0	RW

**REG[5Dh-5Ch] Active Window Height (AW\_HT)**

Bit	Description	Default	Access
7-0	<b>Height of Active Window [13:8]</b> REG[5Ch] mapping to AW_HT[7:0] REG[5Dh] bit[5:0] mapping to AW_HT[13:8], bit[7:6] are not used. Unit: Pixel. The value is physical pixel width. Maximum pixel width is 8,192. These Registers will be ignored if canvas is in linear Addressing mode.	0	RW

## REG[5Eh] Color Depth of Canvas &amp; Active Window (AW\_COLOR)

Bit	Description	Default	Access
7-4	NA	0	RO
3	<b>Select What will Read Back from Graphic Read/Write Position Register</b> 0: Read back Graphic Write position 1: Read back Graphic Read position (Pre-fetch Address)	0	RW
2	<b>Canvas Addressing Mode</b> 0: Block mode (X-Y coordinates addressing) 1: Linear mode	0	RW
1-0	<b>Canvas Image's Color Depth &amp; Memory R/W Data Width</b> <i>In Block Mode:</i> 00b: 8bpp 01b: 16bpp 1xb: 24bpp <b>Note:</b> Monochrome data can be input under any one color depth and the proper image width. <i>In Linear Mode:</i> X0: 8-bits memory data read/write. X1: 16-bits memory data read/write	0	RW

**Note:** Please refer to Figure 10-2 Canvas Window and Active Window.

## REG[60h-5Fh] Graphic Read/Write X-Coordinate Register (CURH)

Bit	Description	Default	Access
7-0	<b>Write: Set Graphic Read/Write X-Coordinate position</b> <b>Read: Read Graphic Read/Write X-Coordinate position</b> To decide whether it is “Read position” or “Write position”, it depends on the setting of REG[5Eh] bit3. REG[5Fh] mapping to CURH[7:0] REG[60h] bit[4:0] mapping to CURH[12:8], bit[7:5] are not used. <i>When DPRAM in Linear mode:</i> Memory Read/Write address [15:0], Unit: Byte. <i>When DPRAM in Block mode:</i> Graphic Read/Write X-Coordinate [12:0], Unit: Pixel.	0	RW

**Note:** Host should program proper active window related parameters before configuring this register.

## REG[62h-61h] Graphic Read/Write Y-Coordinate Register (CURV)

Bit	Description	Default	Access
7-0	<p><b>Write:</b> Set Graphic Read/Write X-Coordinate position  <b>Read:</b> Read Graphic Read/Write X-Coordinate position</p> <p>To decide whether it is “Read position” or “Write position”, it depends on the setting of REG[5Eh] bits.</p> <p>REG[61h] mapping to CURV[7:0]      REG[62h] bit[4:0] mapping to CURV[12:8], bit[7:5] are not used.</p> <p><b>When DPRAM In Linear Mode:</b>      Memory Read/Write address [31:16], Unit: Byte.</p> <p><b>When DPRAM In Block Mode:</b>      Graphic Read/Write Y-Coordinate [12:0], Unit: Pixel.</p>	0	RW

**Note:** Host should program proper active window related parameters before configuring this register.

## REG[64h-63h] Text Write X-Coordinates Register (F\_CURX)

Bit	Description	Default	Access
7-0	<p><b>Text Write X-coordinates F_CURX[12:0]</b>      REG[63h] mapping to F_CURX[7:0]      REG[64h] bit[4:0] mapping to F_CURX[12:8], bit[7:5] are not used.</p> <p><b>Write:</b> Set Text Write X-Coordinates position  <b>Read:</b> Current Text Write X-Coordinates position</p>	0	RW

## REG[66h-65h] Text Write Y-Coordinates Register (F\_CURY)

Bit	Description	Default	Access
7-0	<p><b>Text Write Y-coordinates F_CURY[12:0]</b>      REG[65h] mapping to F_CURY[7:0]      REG[66h] bit[4:0] mapping to F_CURY[12:8], bit[7:5] are not used.</p> <p><b>Write:</b> Set Text Write Y-Coordinates position  <b>Read:</b> Current Text Write Y-Coordinates position</p>	0	RW

## REG[67h] Draw Line/Triangle Control Register 0 (DCR0)

Bit	Description	Default	Access
7	<b>Draw Line / Triangle Start Control</b> <b>Write:</b> 0: Stop the drawing function. 1: Start the drawing function. <b>Read:</b> 0: Drawing function complete. 1: Drawing function is processing.	0	RW
6	NA	0	RO
5	<b>Fill Function for Triangle</b> 0: Non Fill 1: Fill	0	RW
4-1	<b>Draw Triangle or Line Select</b> 0000b: Draw Line 0001b: Draw Triangle 0010b: Rectangle 0011b: Quadrilateral 0100b: Pentagon 0101b: Polyline (3EP) 0110b: Polyline (4EP) 0111b: Polyline (5EP) 1000b: Ellipse 1001b: Rounded-Rectangle 1010b, 1011b: NA 1100b: Oval Arc on upper-right / 1st Quadrant 1101b: Oval Arc on upper-left / 2nd Quadrant 1110b: Oval Arc on Lower-Left / 3rd Quadrant 1111b: Oval Arc on Lower-Right / 4th Quadrant	0	RW
0	<b>Polyline Style</b> 0: Open-end Polyline, 1: Closed Polyline (connect the last point to the start point)	0	RO

**REG[69h-68h] Draw Line/Rectangle/Triangle Point 1 X-Coordinates Register (DLHSR)**

Bit	Description	Default	Access
7-0	<b>Draw Line/Rectangle/Triangle Point 1 X-coordinates DLHSR[12:0]</b> REG[68h] mapping to DLHSR[7:0] REG[69h] bit[4:0] mapping to DLHSR[12:8], bit[7:5] are not used. Unit: Pixel. When drawing a rectangle, start point & end point cannot be located at the same point or at the same X- Coordinates or Y- Coordinates.	0	RW

**REG[6Bh-6Ah] Draw Line/Rectangle/Triangle Point 1 Y-Coordinates Register (DLVSR)**

Bit	Description	Default	Access
7-0	<b>Draw Line/Rectangle/Triangle Point 1 Y-coordinates DLVSR[12:0]</b> REG[6Ah] mapping to DLVSR[7:0] REG[6Bh] bit[4:0] mapping to DLVSR[12:8], bit[7:5] are not used.	0	RW

**REG[6Dh-6Ch] Draw Line/Rectangle/Triangle Point 2 X-Coordinates Register (DLHER)**

Bit	Description	Default	Access
7-0	<b>Draw Line/Rectangle/Triangle Point 2 X-coordinates DLHER[12:0]</b> REG[6Ch] mapping to DLHER[7:0] REG[6Dh] bit[4:0] mapping to DLHER[12:8], bit[7:5] are not used.	0	RW

**REG[6Fh-6Eh] Draw Line/Rectangle/Triangle Point 2 Y-Coordinates Register (DLVER)**

Bit	Description	Default	Access
7-0	<b>Draw Line/Rectangle/Triangle Point 2 Y-coordinates DLVER[12:0]</b> REG[6Eh] mapping to DLVER[7:0] REG[6Fh] bit[4:0] mapping to DLVER[12:8], bit[7:5] are not used.	0	RW

**REG[71h-70h] Draw Triangle Point 3 X-Coordinates Register (DTPH)**

Bit	Description	Default	Access
7-0	<b>Draw Line/Rectangle/Triangle Point 3 X-coordinates DTPH[12:0]</b> REG[70h] mapping to DTPH[7:0] REG[71h] bit[4:0] mapping to DTPH[12:8], bit[7:5] are not used.	0	RW

**REG[73h-72h] Draw Triangle Point 3 Y-Coordinates Register (DTPV)**

Bit	Description	Default	Access
7-0	<b>Draw Triangle Point 3 Y-coordinates DTPV[12:0]</b> REG[72h] mapping to DTPV[7:0] REG[73h] bit[4:0] mapping to DTPV[12:8], bit[7:5] are not used.	0	RW

**Note:** When Drawing a triangle: If any two endpoints are overlapped, it will draw a line. If three endpoints are overlapped, it will draw a dot.

**REG[76h] Draw Circle/Ellipse/Ellipse Curve/Circle Square Control Register 1 (DCR1)**

Bit	Description	Default	Access
7	<b>Draw Circle / Ellipse / Square / Rounded-rectangle Control</b> <b>Write:</b> 0: Stop the drawing function. 1: Start the drawing function. <b>Read:</b> 0: Drawing function complete. 1: Drawing function is processing.	0	RW
6	<b>Fill the Circle / Ellipse / Square / Rounded-rectangle Control</b> 0: Non Fill 1: Fill	0	RW
5-4	<b>Draw Circle / Ellipse / Square / Ellipse Curve / Rounded-rectangle Select</b> 00b: Draw Circle / Ellipse 01b: Draw Arc 10b: Draw Rectangle 11b: Draw Rounded-Rectangle	0	RW
3-2	NA	NA	NA
1-0	<b>Draw Circle / Ellipse Curve Part Select (DECP)</b> 00b: bottom-left Ellipse Curve 01b: upper-left Ellipse Curve 10b: upper-right Ellipse Curve 11b: bottom-right Ellipse Curve	0	RW

**REG[78h-77h] Draw Circle/Ellipse/Rounded-Rectangle Major-Radius Register (ELL\_A)**

Bit	Description	Default	Access
7-0	<b>Draw Circle/Ellipse/Rounded-rectangle Major-Radius</b> REG[77h] mapping to ELL_A[7:0] REG[78h] bit[4:0] mapping to ELL_A[12:8], bit[7:5] are not used. If drawing a circle, then the major radius must be equal to the minor radius (ELL_A = ELL_B).	0	RW

**REG[7Ah-79h] Draw Circle/Ellipse/Rounded-rectangle Minor-Radius Register (ELL\_B)**

Bit	Description	Default	Access
7-0	<b>Draw Circle/Ellipse/Rounded-rectangle Minor-Radius</b> REG[79h] mapping to ELL_B[7:0] REG[7Ah] bit[4:0] mapping to ELL_B[12:8], bit[7:5] are not used.	0	RW

**REG[7Ch-7Bh] Draw Circle/Ellipse/Rounded-Rectangle Center X-Coordinates Register (DEHR)**

Bit	Description	Default	Access
7-0	<b>Draw Circle/Ellipse/Rounded-rectangle Center X-coordinates DEHR[12:0]</b> REG[7Bh] mapping to DEHR[7:0] REG[7Ch] bit[4:0] mapping to DEHR[12:8], bit[7:5] are not used.	0	RW

**REG[7Eh-7Dh] Draw Circle/Ellipse/Rounded-Rectangle Center Y-Coordinates Register (DEVR)**

Bit	Description	Default	Access
7-0	<b>Draw Circle/Ellipse/Rounded-rectangle Center Y-coordinates DEHR[12:0]</b> REG[7Dh] mapping to DEVR[7:0] REG[7Eh] bit[4:0] mapping to DEVR[12:8], bit[7:5] are not used.	0	RW

**Note:** The unit of REG[77h] ~ REG[7Eh] is Pixel.

**REG[D2h] Foreground Color Register - Red (FGCR)**

Bit	Description	Default	Access
7-0	<b>Foreground Color – Red (for Draw mode, Text mode, and Color Expansion mode)</b> 256 Colors: Red mapping to bit[7:5] 65K Colors: Red mapping to bit[7:3] 16.7M Colors: Red mapping to bit[7:0]	FFh	RW

## REG[D3h] Foreground Color Register - Green (FGCG)

Bit	Description	Default	Access
7-0	<b>Foreground Color – Green (for Draw mode, Text mode, and Color Expansion mode)</b> <ul style="list-style-type: none"> <li>■ 256 Colors: Green mapping to bit[7:5]</li> <li>■ 65K Colors: Green mapping to bit[7:2]</li> <li>■ 16.7M Colors: Green mapping to bit[7:0]</li> </ul>	FFh	RW

## REG[D4h] Foreground Color Register - Blue (FGCB)

Bit	Description	Default	Access
7-0	<b>Foreground Color – Blue (for Draw mode, Text mode and Color Expansion mode)</b> <ul style="list-style-type: none"> <li>■ 256 Colors: Blue mapping to bit[7:6]</li> <li>■ 65K Colors: Blue mapping to bit[7:3]</li> <li>■ 16.7M Colors: Blue mapping to bit[7:0]</li> </ul>	FFh	RW

**Note:** For Background color setting, please refer to the Text Engine Registers REG[D5h–D7h].

## 19.7 PWM Control Registers

REG[84h] PWM Prescaler Register (PSCLR)

Bit	Description	Default	Access
7-0	<b>PWM Pre-scaler Register</b> This register determine the pre-scaler value for Timer 0 and 1. The Base Frequency is: $\text{Core\_Freq} / (\text{Prescaler} + 1)$	0	RW

REG[85h] PWM Clock Mux Register (PMUXR)

Bit	Description	Default	Access
7-6	<b>Setup PWM Timer-1 Divisor</b> (Select 2 <sup>nd</sup> Clock Divider's MUX Input for PWM Timer-1) 00b = 1 01b = 1/2 10b = 1/4 11b = 1/8	0	RW
5-4	<b>Setup PWM Timer-0 Divisor</b> (Select 2 <sup>nd</sup> Clock Divider's MUX Input for PWM Timer-0) 00b = 1 01b = 1/2 10b = 1/4 11b = 1/8	0	RW
3-2	<b>PWM[1] Function Control</b> 0xb: PWM[1] output system error flag (Scan FIFO POP error or Memory access out of range) 10b: PWM[1] output PWM timer 1 event or invert of PWM timer 0 11b: PWM[1] output Oscillator Clock	0	RW
1-0	<b>PWM[0] Function Control</b> 0xb: PWM[0] becomes GPIOC[7] 10b: PWM[0] output PWM Timer 0 11b: PWM[0] output Core Clock (CCLK).	0	RW

## REG[86h] PWM Configuration Register (PCFGR)

Bit	Description	Default	Access
7	NA	NA	NA
6	<b>PWM Timer-1 Output Inverter On/Off</b> Determine the output inverter on/off for Timer 1. 0 = Inverter off 1 = Inverter on for PWM1	0	RW
5	<b>PWM Timer-1 Auto Reload On/Off</b> Determine auto reload on/off for Timer 1. 0: One-Shot Mode 1: Interval Mode (Auto Reload)	1	RW
4	<b>PWM Timer-1 Start/Stop</b> 0: Stop 1: Start  In Interval Mode, Host needs to program it as 0 to stop PWM timer. In One-shot Mode, this bit will be auto cleared.  The Host may read this bit to find out if the current PWMx is running or stopped.	0	RW
3	<b>PWM Timer-0 Dead Zone Enable</b> 0: Disable 1: Enable	0	RW
2	<b>PWM Timer-0 Output Inverter On/Off</b> Determine the output inverter on/off for Timer 0. 0 = Inverter off 1 = Inverter on for PWM0	0	RW
1	<b>PWM Timer-0 Auto Reload On/Off</b> Determine auto reload on/off for Timer 0. 0: One-Shot Mode 1: Interval Mode (Auto Reload)	1	RW
0	<b>PWM Timer-0 Start/Stop</b> 0: Stop 1: Start  In Interval Mode, Host needs to program it as 0 to stop PWM timer. In One-shot Mode, this bit will be auto cleared.  The Host may read this bit to find out if the current PWMx is running or stopped.	0	RW

## REG[87h] Timer-0 Dead Zone Length Register [DZ\_LENGTH]

Bit	Description	Default	Access
7-0	<b>Timer-0 Dead Zone Length Register</b> These 8 bits determine the dead zone length. The 1 unit time of the dead zone length is equal to Timer 0.	0	RW

**REG[88h-89h] Timer-0 Compare Buffer Register [TCMPB0]**

Bit	Description	Default	Access
7-0	<b>Timer-0 compare Buffer Register</b> REG[88h] mapping to TCMPB0 [7:0] REG[89h] mapping to TCMPB0 [15:8] The Timer-0 Compare Buffer Registers have a total of 16bits. When the counter is equal to or less than the value of this register, and if INV_ON bit is Off, then PWM1 output is high level.	0	RW

**REG[8Ah-8Bh] Timer-0 Count Buffer Register [TCNTB0]**

Bit	Description	Default	Access
7-0	<b>Timer-0 Count Buffer Register [15:0]</b> REG[8Ah] mapping to TCNTB0 [7:0] REG[8Bh] mapping to TCNTB0 [15:8] The Timer-0 count registers have a total of 16bit. When the counter is equal to 0 and the Reload_EN is enabled, the PWM will reload the value of this register to the counter. When the PWM begins to count, the current count value can be read back through this register.	0	RW

**REG[8Ch-8Dh] Timer-1 Compare Buffer Register [TCMPB1]**

Bit	Description	Default	Access
7-0	<b>Timer-1 Compare Buffer Register</b> REG[8Ch] mapping to TCMPB1 [7:0] REG[8Dh] mapping to TCMPB1 [15:8] The Timer-1 Compare Buffer Registers have a total of 16bits. When the counter is equal to or less than the value of this register, and if INV_ON bit is Off, then PWM1 output is high level.	0	RW

**REG[8Eh-8Fh] Timer-1 Count Buffer Register [TCNTB1]**

Bit	Description	Default	Access
7-0	<b>Timer-1 Count Buffer Register [15:0]</b> REG[8Eh] mapping to TCNTB1 [7:0] REG[8Fh] mapping to TCNTB1 [15:8] The Timer-1 count registers have a total of 16bit. When the counter is equal to 0 and the Reload_EN is enabled, the PWM will reload the value of this register to the counter. When the PWM begins to count, the current count value can be read back through this register.	0	RW

## 19.8 Bit Block Transfer Engine (BTE) Control Registers

**REG[90h] BTE Function Control Register 0 (BLT\_CTRL0)**

Bit	Description	Default	Access
7-5	NA	0	RO
4	<b>BTE Function Enable / Status</b> <i>Write:</i> 0: No Action 1: BTE Enable <i>Read:</i> 0: BTE function is idle. 1: BTE function is busy. When BTE function is enabled, Host is not allowed to R/W memory through canvas[active window].	0	RW
3-1	NA	0	RO
0	<b>Pattern Format</b> 0: 8*8 1: 16*16	0	RW

## REG[91h] BTE Function Control Register1 (BLT\_CTRL1)

Bit	Description	Default	Access																																		
7-4	<p><b>BTE ROP Code or Color Expansion Starting</b> ROP stands for Raster Operation. Some of BTE operation code has to collocate with ROP for advanced functions.</p> <p style="text-align: center;"><b>Table 19-3: BTE ROP Code</b></p> <table border="1"> <thead> <tr> <th>Bit[7:4]</th><th>Description</th></tr> </thead> <tbody> <tr><td>0000b</td><td>0 (Blackness)</td></tr> <tr><td>0001b</td><td><math>\sim S_0 \cdot \sim S_1</math> or <math>\sim (S_0 + S_1)</math></td></tr> <tr><td>0010b</td><td><math>\sim S_0 \cdot S_1</math></td></tr> <tr><td>0011b</td><td><math>\sim S_0</math></td></tr> <tr><td>0100b</td><td><math>S_0 \cdot \sim S_1</math></td></tr> <tr><td>0101b</td><td><math>\sim S_1</math></td></tr> <tr><td>0110b</td><td><math>S_0 \wedge S_1</math></td></tr> <tr><td>0111b</td><td><math>\sim S_0 + \sim S_1</math> or <math>\sim (S_0 \cdot S_1)</math></td></tr> <tr><td>1000b</td><td><math>S_0 \cdot S_1</math></td></tr> <tr><td>1001b</td><td><math>\sim (S_0 \wedge S_1)</math></td></tr> <tr><td>1010b</td><td><math>S_1</math></td></tr> <tr><td>1011b</td><td><math>\sim S_0 + S_1</math></td></tr> <tr><td>1100b</td><td><math>S_0</math></td></tr> <tr><td>1101b</td><td><math>S_0 + \sim S_1</math></td></tr> <tr><td>1110b</td><td><math>S_0 + S_1</math></td></tr> <tr><td>1111b</td><td>1 (Whiteness)</td></tr> </tbody> </table> <p>If BTE operation code function is color expansion with or without chroma key (08h / 09h / Eh / Fh), then these bits stand for starting bit on BTE window left boundary. MSB stands for left most pixel. For 8-bits Host, the value should be within 0 to 7. For 16-bits Host, the value should be within 0 to 15.</p>	Bit[7:4]	Description	0000b	0 (Blackness)	0001b	$\sim S_0 \cdot \sim S_1$ or $\sim (S_0 + S_1)$	0010b	$\sim S_0 \cdot S_1$	0011b	$\sim S_0$	0100b	$S_0 \cdot \sim S_1$	0101b	$\sim S_1$	0110b	$S_0 \wedge S_1$	0111b	$\sim S_0 + \sim S_1$ or $\sim (S_0 \cdot S_1)$	1000b	$S_0 \cdot S_1$	1001b	$\sim (S_0 \wedge S_1)$	1010b	$S_1$	1011b	$\sim S_0 + S_1$	1100b	$S_0$	1101b	$S_0 + \sim S_1$	1110b	$S_0 + S_1$	1111b	1 (Whiteness)	0	RW
Bit[7:4]	Description																																				
0000b	0 (Blackness)																																				
0001b	$\sim S_0 \cdot \sim S_1$ or $\sim (S_0 + S_1)$																																				
0010b	$\sim S_0 \cdot S_1$																																				
0011b	$\sim S_0$																																				
0100b	$S_0 \cdot \sim S_1$																																				
0101b	$\sim S_1$																																				
0110b	$S_0 \wedge S_1$																																				
0111b	$\sim S_0 + \sim S_1$ or $\sim (S_0 \cdot S_1)$																																				
1000b	$S_0 \cdot S_1$																																				
1001b	$\sim (S_0 \wedge S_1)$																																				
1010b	$S_1$																																				
1011b	$\sim S_0 + S_1$																																				
1100b	$S_0$																																				
1101b	$S_0 + \sim S_1$																																				
1110b	$S_0 + S_1$																																				
1111b	1 (Whiteness)																																				
3-0	<p><b>BTE Operation Code bit[3:0]</b> LT768x has an embedded a 2D BTE Engine, it can execute 13 BTE functions. Some of BTE Operation Code has to collocate with the ROP code for advanced functions.</p>	0	RW																																		

Table 19-4: BTE Operation Code

REG[91h] Bit[3:0]	Description
0000b	<b>MCU Write with ROP</b> S0: Data comes from Host S1: Data comes from Memory D: According to ROP write to Memory
0001b	Reserved
0010b	<b>Memory Copy with ROP</b> S0: Data comes from Memory S1: Data comes from Memory D: According to ROP Write to memory
0011b	Reserved
0100b	<b>MCU Write with Chroma Keying (without ROP)</b> S0: Data comes from Host If the Host data is not the same color as the Chroma Key (Background Color Register), then the data will be written to the destination memory.
0101b	<b>Memory Copy (move) with Chroma keying (without ROP)</b> S0: Data comes from Memory If the S0 data is not the same color as the chroma key Background Color Register, then the data will be written to the destination.
0110b	<b>Pattern Fill with ROP</b> S0: Pattern stored in memory.
0111b	<b>Pattern Fill with Chroma Keying</b> S0: Pattern stored in memory. If the data for S0 is not the same as the Chroma Key (Background Color) color, then it will be written to the destination memory.
1000b	<b>MCU Write with Color Expansion</b> S0: Data comes from Host BTE converts it to the specified color and color depth, and writes it to the destination memory.
1001b	<b>MCU Write with Color Expansion and Chroma Keying</b> The monochrome data required by S0 is written by MCU if the bit of monochrome data is 1. The processed data is a foreground color, and if the monochrome data is 0, it is not written. The data is also referenced to the destination memory setting.
1010b	<b>Memory Copy with Opacity</b> S0, S1 & D: Data come from Memory

REG[91h] Bit[3:0]	Description
1011b	<b>MCU Write with Opacity</b> S0: Data comes from Host S1: Data comes from Memory DT: Refer to the alpha blending operation and write to the destination memory.
1100b	<b>Solid Fill</b> The written value is the register setting value, and the written target is the destination memory.
1101b	Reserved
1110b	<b>Memory Copy with Color Expansion</b> S0 and DT are in memory and S1 is not in use. S0 must be loaded through the microprocessor's write or DMA to preload 8bpp or 16bpp of color depth into memory, so the S0 color depth should follow that color depth.
1111b	<b>Memory Copy with Color Expansion and Chroma Keying</b> S0 and DT are in memory and S1 is not in use. S0 must be loaded through the microprocessor's write or DMA to preload 8bpp or 16bpp of color depth into memory, so the S0 color depth should follow that color depth. If the S0 data bit = 0, then no data will be written to DT. If the S0 data bit = 1, the Foreground Color data will be written to DT.

REG[92h] Source 0/1 &amp; Destination Color Depth (BLT\_COLR)

Bit	Description	Default	Access
7	NA	NA	NA
6-5	<b>S0 Color Depth</b> 00b: 256 Color (8bpp) 01b: 64k Color (16bpp) 1xb: 16M Color (24bpp)	0	RW
4-2	<b>S1 Color Depth</b> 000b: 256 Color (8bpp) 001b: 64k Color (16bpp) 010b: 16M Color (24bpp) 011b: Constant color (S1 memory start address' setting must be defined as S1 constant color definition) 100b: 8 bit pixel alpha blending 101b: 16 bit pixel alpha blending	0	RW

Bit	Description	Default	Access
1-0	<b>Destination Color Depth</b> 00b: 256 Color (8bpp) 01b: 64k Color (16bpp) 1xb: 16M Color (24bpp)	0	RW

**REG[93h-96h] Source 0 Memory Start Address (S0\_STR)**

Bit	Description	Default	Access
7-0	<b>Source 0 Memory Start Address [31:2]</b> REG[93h] mapping to S0_STR [7:2] REG[94h] mapping to S0_STR [15:8] REG[95h] mapping to S0_STR [23:16] REG[96h] mapping to S0_STR [31:24] <b>Note:</b> REG[93h] bit[1:0] is fixed to 0.	0	RW

**REG[97h-98h] Source 0 Image Width (S0\_WTH)**

Bit	Description	Default	Access
7-0	<b>Source 0 Image Width [12:2]</b> REG[97h] mapping to S0_WTH [7:2] REG[98h] bit[4:0] mapping to S0_WTH [12:8], bit[7-5] are not used. <b>Note:</b> It must be divisible by 4, and REG[97h] bit[1:0] must be fixed to 0. Unit: Pixel.	0	RW

**REG[99h-9Ah] Source 0 Window Upper-Left Corner X-Coordinates (S0\_X)**

Bit	Description	Default	Access
7-0	<b>Source 0 Window Upper-Left Corner X-Coordinates [12:0]</b> REG[99h] mapping to S0_X [7:0] REG[9Ah] bit[4:0] mapping to S0_X [12:8], bit[7-5] are not used.	0	RW

**REG[9Bh-9Ch] Source 0 Window Upper-Left corner Y-Coordinates (S0\_Y)**

Bit	Description	Default	Access
7-0	<b>Source 0 Window Upper-Left Corner Y-Coordinates [12:0]</b> REG[9Bh] mapping to S0_Y [7:0] REG[9Ch] bit[4:0] mapping to S0_Y[12:8], bit[7-5] are not used.	0	RW

**REG[9Dh-A0h] Source 1 Memory Start Address 0 (S1\_STR)**

Bit	Description	Default	Access
7-0	<p><b>Source 1 Memory Start Address [31:2]</b>            REG[9Dh] bit[7:2] mapping to S1_STR[7:2]            REG[9Eh] mapping to S1_STR[15:8]            REG[9Fh] mapping to S1_STR[23:16]            REG[A0h] mapping to S1_STR[31:24]</p> <p><b>Note1:</b> REG[9Dh] bit[1:0] must be fixed to 0.  <b>Note2:</b> If source 1(S1) is set as Constant Color then S1 memory start address' setting will be defined as S1 constant color definition: REG[9Dh] is RED element (S1_RED), REG[9Eh] is Green element (S1_GREEN), REG[9Fh] is Blue Element (S1_BLUE), REG[A0h] is not used.</p>	0	RW

**REG[A1h-A2h] Source 1 Image Width (S1\_WTH)**

Bit	Description	Default	Access
7-0	<p><b>Source 1 Image Width [12:2]</b>            REG[A1h] [7:2] mapping to S1_WTH [7:2]            REG[A2h] bit[4:0] mapping to S1_WTH[12:8], bit[7:5] are not used.</p> <p><b>Note:</b> It must be divisible by 4, and REG[A1h] bit[1:0] must be fixed to 0. Unit: Pixel.</p>	0	RW

**REG[A3h-A4h] Source 1 Window Upper-Left Corner X-Coordinates (S1\_X)**

Bit	Description	Default	Access
7-0	Source 1 Window Upper-Left Corner X-Coordinates [12:0] REG[A3h] mapping to S1_X [7:0] REG[A4h] bit[4:0] mapping to S1_X [12:8], bit[7:5] are not used.	0	RW

**REG[A5h-A6h] Source 1 Window Upper-Left corner Y-Coordinates (S1\_Y)**

Bit	Description	Default	Access
7-0	<p><b>Source 1 Window Upper-Left Corner Y-Coordinates [12:0]</b>            REG[A5h] mapping to S1_Y [7:0]            REG[A6h] bit[4:0] mapping to S1_Y [12:8], bit[7:5] are not used.</p>	0	RW

**REG[A7h-AAh] Destination Memory Start Address (DT\_STR)**

Bit	Description	Default	Access
7-2	<b>Destination Memory Start Address [31:2]</b> REG[A7h] bit[7:2] mapping to DT_STR [7:2] REG[A8h] mapping to DT_STR [15:8] REG[A9h] mapping to DT_STR [23:16] REG[AAh] mapping to DT_STR [31:24] <b>Note:</b> REG[A7h] bit[1:0] must be fixed to 0.	0	RW

**Note:** The destination memory start address cannot be within the source 0 or Source 1 block, otherwise the output will be incorrect.

$$\text{Start Address} \sim S0/1's (\text{Image\_Width}) * (\text{Image\_Height}) * ([1|2|3]\text{Color Depth})$$

**REG[ABh-ACh] Destination Image Width (DT\_WTH)**

Bit	Description	Default	Access
7-0	<b>Destination Image Width [12:2]</b> REG[ABh] mapping to DT_WTH [7:2] REG[ACh] bit[4:0] mapping to DT_WTH [12:8], REG[ACh] bit[7-5] are not used. <b>Note:</b> It must be divisible by 4. REG[ABh] bit[1:0] must be fixed to 0. Unit: Pixel.	0	RW

**REG[ADh-AEh] Destination Window Upper-Left Corner X-Coordinates (DT\_X)**

Bit	Description	Default	Access
7-0	<b>Destination Window Upper-Left Corner X-Coordinates [12:0]</b> REG[ADh] mapping to DT_X [7:0] REG[AEh] bit[4:0] mapping to DT_X [12:8], bit[7-5] are not used.	0	RW

**REG[AFh-B0h] Destination Window Upper-Left Corner Y-Coordinates (DT\_Y)**

Bit	Description	Default	Access
7-0	<b>Destination Window Upper-Left Corner X-Coordinates [12:0]</b> REG[AFh] mapping to DT_Y [7:0] REG[B0h] bit[4:0] mapping to DT_Y [12:8], bit[7-5] are not used.	0	RW

**REG[B1h-B2h] BTE Window Width (BLT\_WTH)**

Bit	Description	Default	Access
7-0	<b>BTE Window Width [12:0]</b> REG[B1h] mapping to BLT_WTH [7:0] REG[B2h] bit[4:0] mapping to BLT_WTH [12:8], bit[7-5] are not used. When all Image Fill (Pattern Fill) functions for BTE are enabled, the BTE window Width is ignored and automatically set to 8 or 16. Unit: Pixel.	0	RW

**REG[B3h-B4h] BTE Window Height (BLT\_HIG)**

Bit	Description	Default	Access
7-0	<b>Destination Image Height [12:0]</b> REG[B3h] mapping to BLT_HIG [7:0] REG[B4h] bit[4:0] mapping to BLT_HIG [12:8], bit[7-5] are not used. When all Image Fill (Pattern Fill) functions for BTE are enabled, the BTE window Height is ignored and automatically set to 8 or 16. Unit: Pixel.	0	RW

**REG[B5h] Alpha Blending (APB\_CTRL)**

Bit	Description	Default	Access
7-4	NA	NA	NA
5-0	<b>Window Alpha Blending Effect for S0 &amp; S1</b> The value of alpha in the color code ranges from 1.0 down to 0.0, where 1.0 represents a fully opaque color, and 0.0 represents a fully transparent color. 00h: 0 01h: 1/32 02h: 2/32 : : 1Eh: 30/32 1Fh: 31/32 2Xh: 1 <b>Output Effect</b> $= [ S0 \text{ image} * (1 - \text{Alpha Setting Value}) ] + (S1 \text{ Image} * \text{Alpha Setting Value})$	0	RW

## 19.9 Serial Flash & SPI Master Control Registers

**REG[B6h] Serial Flash DMA Controller REG (DMA\_CTRL)**

Bit	Description	Default	Access
7-1	NA	0	RO
0	<p><b>Write:</b> DMA Start Bit</p> <p>This bit can be written to 1 via Host, and the circuit will be automatically cleared to 0 right away. This bit cannot be used with Text Write, so if DMA is enabled, it cannot be set to text mode and input character code.</p> <p><b>Read:</b> DMA Busy Check Bit</p> <p>0: Idle 1: Busy</p> <p>The DMA transmission of a Serial Flash must be in Graphic Mode, and it must first set the canvas destination location, destination width, color depth, and addressing mode in the Display RAM.</p>	0	RW

**REG[B7h] Serial Flash/ROM Controller Register (SFL\_CTRL)**

Bit	Description	Default	Access
7	<p><b>Serial Flash/ROM Select</b></p> <p>0: Serial Flash/ROM 0 I/F is selected 1: Serial Flash/ROM 1 I/F is selected</p>	0	RW
6	<p><b>Serial Flash /ROM Access Mode</b></p> <p>0: Text Mode, used for CGRAM 1: DMA Mode, used for CGRAM、Pattern、Boot Start Image or OSD.</p>	0	RW
5	<p><b>Serial Flash/ROM Address Mode</b></p> <p>0: 24bits address mode 1: 32bits address mode</p> <p>If user wants to use 32 bits address mode, Host must send EX4B command (B7h) to serial flash then set this bit to 1. Host also may check this bit to find out whether serial flash entered 32 bits address mode during initial display function.</p>	0	RW
4	NA	NA	NA
3-0	<p><b>Read Command Code &amp; Behavior Selection</b></p> <p>000xb: 1x Read Command - 03h. Normal Read Speed. Data is read from SFDI. Without dummy cycle between address and data</p> <p>010xb: 1x Read Command - 0Bh. Faster Read speed. Data is read from SFDI. LT768x inserts 8 dummy cycles between address and data.</p> <p>1x0xb: 1x Read Command - 1Bh. Fastest Read Speed. Data read from SFDI. LT768x inserts 16 dummy cycles between address and data.</p>	0	R/W

Bit	Description	Default	Access
	<p>xx10b: 2x Read Command - 3Bh. Interleaved data input from SFDI and SFDO. LT768x inserts 8 dummy cycles (Dual Mode 0) between address and data. Refer to Figure 10-8.</p> <p>xx11b: 2x Read Command – BBh. Interleaved data input from SFDI and SFDO. LT768x inserts 4 dummy cycles (Dual Mode 1) between address and data. Refer to Figure 10-9.</p> <p><b>Note:</b> Not all of Serial Flash supports above read command. Please refer to Serial Flash's datasheet to select proper read command.</p>		

## REG[B8h] SPI Master Tx /Rx FIFO Data Register (SPIDR)

Bit	Description	Default	Access
7-0	<p><b>SPI Master Tx /Rx FIFO Data Register</b></p> <p>After Host programming the LT768x's control register, SPI transfers can be initiated. A transfer is initiated by writing to the Serial Peripheral Data Register [SPIDR]. Writing to the Serial Peripheral Data Register is actually writing to a 16 entries deep FIFO called the Write FIFO. Each write access adds a data byte to the Write FIFO. When the core is enabled – SS_ACTIVE is set '1' – and the Write FIFO is not full, the core automatically transfers the oldest data byte.</p> <p>Receiving data is done simultaneously with transmitting data; whenever a data byte is transmitted, a data byte is received. For each byte that needs to be read from a device, a dummy byte needs to be written to the Write FIFO. This instructs the core to initiate an SPI transfer, simultaneously transmitting the dummy byte and receiving the desired data. Whenever a transfer is finished, the received data byte is added to the Read FIFO. The Read FIFO is the counterpart of the Write FIFO. It is an independent 16 entries deep FIFO. The FIFO contents can be read by reading from the Serial Peripheral Data Register [SPIDR]</p>	NA	RW

## REG[B9h] SPI Master Control Register (SPIMCR2)

Bit	Description	Default	Access															
7	NA	0	RO															
6	<b>SPI Master Interrupt Enable</b> 0: Disable interrupt. 1: Enable interrupt. If Host disables SPI Master interrupt flag, then LT768x will not assert interrupt event to inform Host, but Host still may check interrupt event on SPIMSR.	0	RW															
5	<b>Control Slave Select Drive on Which SFCS0# / SFCS1#</b> 0: Slave Select Signal (SS#) drive on SFCS0# 1: Slave Select Signal (SS#) drive on SFCS1#	0	RW															
4	<b>Slave Select Signal Active [SS_ACTIVE]</b> 0: Inactive (SS# drive High) 1: Active (SS# drive Low) While Slave Select signal is inactive, FIFO will be cleared and the engine will stay in idle state. <b>Note:</b> It is suggested when Slave Select Signal is active, the settings of CPOL/CPHA should not be changed.	0	RW															
3	<b>Mask Interrupt for FIFO Overflow Error [OVFIRQMSK]</b> 0: Unmask. 1: Mask.	1	RW															
2	<b>Mask Interrupt for While Tx FIFO Empty &amp; SPI Engine/FSM Idle [EMTIRQMSK]</b> 0: Unmask. 1: Mask.	1	RW															
1:0	<b>SPI Operation Mode</b> When DMA or external CGROM is enabled, only mode 0 & mode 3 will be supported. <b>Table 19-5: SPI Operation Mode</b> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Mode</th> <th>CPOL: Clock Polarity Bit</th> <th>CPHA: Clock Phase Bit</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>2</td> <td>1</td> <td>0</td> </tr> <tr> <td>3</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Mode	CPOL: Clock Polarity Bit	CPHA: Clock Phase Bit	0	0	0	1	0	1	2	1	0	3	1	1	0	RW
Mode	CPOL: Clock Polarity Bit	CPHA: Clock Phase Bit																
0	0	0																
1	0	1																
2	1	0																
3	1	1																

- Please refer to Section 15.2 for SPI Master description.
- When CPOL = 0, SCK is 0 in inactive.
  - \_ CPHA = 0: data are read on rising edge (low->high), and data are changed on a falling edge (high->low).

- CPHA = 1: data are read on falling edge (high->low), and data are changed on a rising edge (low->high).
- When CPOL = 1, SCK is 1 in inactive (inversion of CPOL = 0)
  - CPHA = 0: data are read on falling edge (high->low), and data are changed on a rising edge (low->high).
  - CPHA = 1: data are read on rising edge (low->high), and data are changed on a falling edge (high->low).

REG[BAh] SPI Master Status Register (SPIMSR)

Bit	Description	Default	Access
7	<b>Tx FIFO Empty Flag</b> 0: Not Empty 1: Empty	1	RO
6	<b>Tx FIFO Full Flag</b> 0: Not Full 1: Full	0	RO
5	<b>Rx FIFO Empty Flag</b> 0: Not empty 1: Empty	1	RO
4	<b>Rx FIFO Full Flag</b> 0: Not Full 1: Full	0	RO
3	<b>Overflow Interrupt Flag</b> Write 1 will clear this flag	0	RW
2	<b>Tx FIFO Empty &amp; SPI Engine/FSM Idle Interrupt Flag</b> Write 1 will clear this flag	0	RW
1-0	NA	0	RO

REG[BBh] SPI Clock period (SPI\_DIVSOR)

Bit	Description	Default	Access
7-0	<b>SPI Clock Period</b> Set SPI clock based on the system clock and the clock requested by the SPI device.  $F_{SCK} = F_{CORE} / (\text{Divisor} + 1)^{*}2$	3	RW

## REG[BCh-BFh] Serial Flash DMA Source Starting Address (DMA\_SSTR)

Bit	Description	Default	Access
7-0	<b>Serial Flash DMA Source Start Address[31:0]</b> REG[BCh] mapping to DMA_SSTR[7:0] REG[BDh] mapping to DMA_SSTR[15:8] REG[BEh] mapping to DMA_SSTR[23:16] REG[BFh] mapping to DMA_SSTR[31:24] These registers index Serial Flash Address [31:0]. Directly point to the starting position of the source images.	0	RW

## REG[C0h-C1h] DMA Destination Window Upper-Left Corner X-Coordinates (DMA\_DX)

Bit	Description	Default	Access
7-0	<b>When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode):</b> This register defines DMA Destination Window Upper-Left corner X-coordinates [12:0] on Canvas area. REG[C0h] mapping to DMA_DX[7:0] REG[C1h] bit[4:0] mapping to DMA_DX[12:8], bit[7:5] are not used. <b>When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode):</b> This register defines Destination address [15:2] in Display RAM. REG[C0h] bit[7:2] mapping to DMA_DX[7:2] REG[C1h] mapping to DMA_DX[15:8]	0	RW

## REG[C2h-C3h] DMA Destination Window Upper-Left Corner Y-Coordinates (DMA\_DY)

Bit	Description	Default	Access
7-0	<b>When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode):</b> This register defines DMA Destination Window Upper-Left corner Y-coordinates [12:0] on Canvas area. REG[C2h] mapping to DMA_DY[7:0] REG[C3h] bit[4:0] mapping to DMA_DY[12:8], bit[7:5] are not used. <b>When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode):</b> This register defines Destination address [31:16] in Display RAM. REG[C2h] mapping to DMA_DY[23:16] REG[C3h] mapping to DMA_DY[31:24]	0	RW

## REG[C4h] – REG[C5h]: Reserved

Bit	Description	Default	Access
7-0	NA	0	RO

## REG[C6h-C7h] DMA Block Width (DMAW\_WTH)

Bit	Description	Default	Access
7-0	<p><b>When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode):</b>  <b>DMA Block Width [15:0]</b></p> <p>REG[C6h] mapping to DMAW_WTH[7:0]  REG[C7h] mapping to DMAW_WTH[15:8]</p> <p><b>When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode):</b>  <b>DMA Transfer Number [15:0]</b></p> <p>REG[C6h] mapping to DMAW_WTH[7:0]  REG[C7h] mapping to DMAW_WTH[15:8]</p>	0	RW

## REG[C8h-C9h] DMA Block Height (DMAW\_HIGH)

Bit	Description	Default	Access
7-0	<p><b>When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode):</b>  <b>DMA Block Height [15:0]</b></p> <p>REG[C8h] mapping to DMAW_HIGH[7:0]  REG[C9h] mapping to DMAW_HIGH[15:8]</p> <p><b>When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode):</b>  <b>DMA Transfer Number [31:16]</b></p> <p>REG[C8h] mapping to DMAW_HIGH [23:16]  REG[C9h] mapping to DMAW_HIGH [31:24]</p>	0	RW

## REG[CAh-CBh] DMA Source Picture Width (DMA\_SWTH)

Bit	Description	Default	Access
7-0	<p><b>DMA Source Picture Width [12:0]</b></p> <p>REG[CAh] mapping to DMA_SWTH[7:0]  REG[CBh] bit[4:0] mapping to DMA_SWTH[12:8], bit[7:5] are not used.  Unit: Pixel.</p>	0	RW

Please refer to Section 15.3 description of SPI Flash Controller.

## 19.10 Text Engine Registers

**REG[CCh] Character Control Register 0 (CCR0)**

Bit	Description	Default	Access
7:6	<b>Character Source Selection</b> 00b: Select internal CGROM Character 01b: Select external CGROM Character 10b: Select user-defined Character. 11b: NA	0	RW
5-4	<b>Character Height Setting for user-defined Character</b> 00b: 16 dots, ex. 8*16 / 16*16 01b: 24 dots, ex. 12*24 / 24*24 10b: 32 dots, ex. 16*32 / 32*32 <b>Note:</b> 1. User-defined character width is decided by character code; width for code < 8000h is 8/12/16. width for code >=8000h is 16/24/32. 2.Internal CGROM supports sizes of 8*16 / 12*24 / 16*32.	0	RW
3-2	N	0	RO
1-0	<b>Character Selection for Internal CGROM</b> When bit[7:6] = 00b, Internal CGROM supports character sets with the standard coding of ISO/IEC 8859-1,2,4,5, which supports English and most of European country languages 00b: ISO/IEC 8859-1 01b: ISO/IEC 8859-2 10b: ISO/IEC 8859-4 11b: ISO/IEC 8859-5	0	RW

**REG[CDh] Character Control Register 1 (CCR1)**

Bit	Description	Default	Access
7	<b>Full Alignment Selection</b> 0: Full alignment disable 1: Full alignment enable. When Full alignment is enable, displayed character width is equal to (Character Height)/2 if character width is equal to or smaller than (Character Height)/2, otherwise displayed font width is equal to Character Height.	0	RW
6	<b>Chroma Keying Enable on Text Input</b> 0: Character's background displayed with specified color. 1: Character's background displayed with original canvas' background.	0	RW
5	NA	NA	NA

Bit	Description	Default	Access
4	<b>Character Rotation</b> 0: Normal. Text direction from left to right then from top to bottom 1: Counterclockwise 90 degree & vertical flip. Text direction from top to bottom then from left to right (VDIR should be set as 1). This attribute can be changed only when previous Text write is finished (Core_Busy = 0)	0	RW
3-2	<b>Character Width Enlargement Factor</b> 00b: X1. 01b: X2. 10b: X3. 11b: X4.	0	RW
1-0	<b>Character Height Enlargement Factor</b> 00b: X1. 01b: X2. 10b: X3. 11b: X4.	0	RW

## REG[CEh-CFh] RESERVED

Bit	Description	Default	Access
7-0	NA	NA	NA

## REG[D0h] Character Line gap Setting Register (FLDR)

Bit	Description	Default	Access
7-5	NA	NA	NA
4-0	<b>Character Line Gap Setting</b> Setting the character line gap. When entered character reaches the boundary of the active window, the controller jump to the next line. (Unit: pixel) Color of gap will be the same as the background color. It will not be enlarged by character enlargement function.	0	RW

## REG[D1h] Character to Character Space Setting Register (F2FSSR)

Bit	Description	Default	Access
7-6	NA	NA	NA
5-0	<b>Character to Character Space Setting</b> 00h: 0 pixel 01h: 1 pixel 02h: 2 pixels : : 3Fh: 63 pixels Color of space will be the same as the background color. It will not be enlarged by character enlargement function.	0	RW

## REG[D2h] Foreground Color Register - Red (FGCR)

Bit	Description	Default	Access
7-0	<b>Foreground Color – Red</b> (for Draw, Text, and Color Expansion) <ul style="list-style-type: none"> <li>■ 256 Colors: Red mapping to bit[7:5]</li> <li>■ 65K Colors: Red mapping to bit[7:3]</li> <li>■ 16.7M Colors: Red mapping to bit[7:0]</li> </ul>	FFh	RW

## REG[D3h] Foreground Color Register - Green (FGCG)

Bit	Description	Default	Access
7-0	<b>Foreground Color – Green</b> (for Draw, Text, and Color Expansion) <ul style="list-style-type: none"> <li>■ 256 Colors: Green mapping to bit[7:5]</li> <li>■ 65K Colors: Green mapping to bit[7:2]</li> <li>■ 16.7M Colors: Green mapping to bit[7:0]</li> </ul>	FFh	RW

## REG[D4h] Foreground Color Register - Blue (FGCB)

Bit	Description	Default	Access
7-0	<b>Foreground Color – Blue</b> (for Draw, Text, and Color Expansion) <ul style="list-style-type: none"> <li>■ 256 Colors: Blue mapping to bit[7:6]</li> <li>■ 65K Colors: Blue mapping to bit[7:3]</li> <li>■ 16.7M Colors: Blue mapping to bit[7:0]</li> </ul>	FFh	RW

## REG[D5h] Background Color Register - Red (BGCR)

Bit	Description	Default	Access
7-0	<b>Background Color – Red (for Text and Color Expansion)</b> <ul style="list-style-type: none"> <li>■ 256 Colors: Red mapping to bit[7:5]</li> <li>■ 65K Colors: Red mapping to bit[7:3]</li> <li>■ 16.7M Colors: Red mapping to bit[7:0]</li> </ul> <p><b>Note:</b> No matter background transparency is enabled or not, please do not set to the same value as Foreground Color, otherwise, image and text will become a square with Foreground Color. When using BTE function, these two values should not be set to the same either.</p>	00h	RW

## REG[D6h] Background Color Register - Green (BGCG)

Bit	Description	Default	Access
7-0	<b>Background Color – Green (for Text and Color Expansion)</b> <ul style="list-style-type: none"> <li>■ 256 Colors: Green mapping to bit[7:5]</li> <li>■ 65K Colors: Green mapping to bit[7:2]</li> <li>■ 16.7M Colors: Green mapping to bit[7:0]</li> </ul> <p><b>Note:</b> No matter background transparency is enabled or not, please do not set to the same value as Foreground Color, otherwise, image and text will become a square with Foreground Color. When using BTE function, these two values should not be set to the same either.</p>	00h	RW

## REG[D7h] Background Color Register - Blue (BGCB)

Bit	Description	Default	Access
7-0	<b>Background Color – Blue (for Text and Color Expansion)</b> <ul style="list-style-type: none"> <li>■ 256 Colors: Blue mapping to bit[7:6]</li> <li>■ 65K Colors: Blue mapping to bit[7:3]</li> <li>■ 16.7M Colors: Blue mapping to bit[7:0]</li> </ul> <p><b>Note:</b> No matter background transparency is enabled or not, please do not set to the same value as Foreground Color, otherwise, image and text will become a square with Foreground Color. When using BTE function, these two values should not be set to the same either.</p>	00h	RW

## REG[D8h] – REG[DAh]: Reserved

Bit	Description	Default	Access
7-0	NA	NA	NA

## REG[DBh] CGRAM Start Address 0 (CGRAM\_STR0)

Bit	Description	Default	Access
7-0	<b>CGRAM Start ADDRESS [7:0]</b> User-defined Characters space REG[DBh] mapping to CGRAM_STR [7:0] REG[DCh] mapping to CGRAM_STR [15:8] REG[DEh] mapping to CGRAM_STR [23:16] REG[DEh] mapping to CGRAM_STR [31:24] Host must save CGRAM data based on the setting of canvas, and set CGRAM address so that Text engine knows where to fetch CGRAM data.	0	RW

If users want to change rotate attribute, character line gap, character-to-character space, foreground color, background color and Text/graphic mode settings, please make sure Task\_Busy (Status Register bit3) status bit is low.

## 19.11 Power Management Control Register

**REG[DFh]: Power Management Register (PMU)**

Bit	Description	Default	Access
7	<p><b>Enter Power Saving State</b></p> <p>0: Normal state or wake up from power saving state 1: Enter power saving state.</p> <p><b>Note:</b> There are 3 ways to wake up from power saving state: External interrupt event, Key Scan wakeup and Software wakeup. Writing this bit to 0 will generate Software wakeup. It will not be cleared until chip resume. MCU must wait until system is awoken before writing to other registers. Users may check this bit or check status bit1 (power saving) to find out if the system is back to the normal state.</p>	0	RW
6-2	NA	NA	NA
1-0	<p><b>Power Saving Mode Definition</b></p> <p>00b: NA 01b: Standby Mode, CCLK &amp; PCLK will Stop. MCLK is provided by MPLL. 10b: Suspend Mode, CCLK &amp; PCLK will Stop. MCLK is provided by OSC. 11b: Sleep Mode, All of Clocks and PLL will Stop.</p>	3	RW

## 19.12 Display RAM Control Register

REG[E0h] SDRAM Attribute Register (SDRAR)

Bit	Description	Default	Access
7	<b>SDRAM Power Saving</b> 0: Execute power down command to enter power saving mode 1: Execute self refresh command to enter power saving mode	0	RW
6	Must keep 0.	0	RW
5	<b>SDRAM Bank Number, SDR_BANK</b> 0: 2 Banks (column addressing size only support 256 words) 1: 4 Banks	1	RW
4-3	<b>SDRAM Row Addressing, SDR_ROW</b> 00b: 2K (A0-A10) 01b: 4K (A0-A11) 1xb: 8K (A0-A12)	1	RW
2-0	<b>SDRAM Column Addressing, SDR_COL</b> 000b: 256 (A0-A7) 001b: 512 (A0-A8) 010b: 1,024 (A0-A9) 011b: 2,048 (A0-A9, A11) 1xxb: 4,096 (A0-A9, A11-A12)	0	RW

Table 19-6: The initialize of REG[E0h]

Model	Embedded Display RAM Type	REG[E0h]	Description
LT7681			
LT7683			
LT7686			
LT7680A-R	128Mb, 16MB, 8M*16	0x29	Bank no: 4, Row Size: 4096, Column Size: 512
LT7680B-R			

**Note:** The value of register REG[E0h] must be set according to the LT768x model name. Otherwise, the display of TFT panel will be abnormal and the image will be garbled.

## REG[E1h] SDRAM Mode Register &amp; Extended Mode Register (SDRMD)

Bit	Description	Default	Access
7-3	Must keep 0.	0	RW
2-0	<b>SDRAM CAS latency, SDR_CASLAT</b> 010b: 2 SDRAM clock 011b: 3 SDRAM clock Others: Reserved <b>Note:</b> The suggest setting value of this register is <b>03h</b> . This register is locked after SDR_INITDONE (REG[E4h] bit0) is set as 1.	011b	RW

## REG[E2h-E3h] SDRAM Auto Refresh Interval (SDR\_REF)

Bit	Description	Default	Access
7-0	<b>SDRAM Auto Refresh Interval</b> REG[E2h] mapping to SDR_REF [7:0]. REG[E3h] mapping to SDR_REF [15:8] The internal refresh time is determined by the refresh cycle of the SDRAM and the row size. For example, if the SDRAM frequency is 100MHz, SDRAM's refresh cycle Tref is 64ms, and the row size is 4,096, then the internal refresh time should be less than $64 \times 10^{-3} / 4096 * 100 \times 10^6 \approx 1562 = 61Ah$ . Therefore the REG[E3h][ E2h] is set to 061Ah. <b>Note:</b> If this register is set to 0000h, SDRAM automatic refresh will be prohibited.	00h	RW

Table 19-7: The Reference Setting of REG[E3h-E2h]

Model	REG[E3h]	REG[E2h]
LT7681		
LT7683		
LT7686		
LT7680A-R	06h	1Ah
LT7680B-R		

## REG[E4h] SDRAM Control Register (SDRCR)

Bit	Description	Default	Access
7-4	Must keep 0.	0	RW
3	<b>Report Warning Condition</b> 0: Disable or Clear warning flag 1: Enable warning flag Warning conditions include (1) memory read cycle is close to the maximum address boundary of SDRAM (exceeding maximum address minus 512 bytes); (2) out of memory access range; (3) insufficient SDRAM bandwidth to fulfill panel's frame rate. Under the above situations, the warning event will be latched. The warning flag could be cleared by setting this bit to 0.	0	RW
2	<b>SDRAM Timing Parameter Register Enable, SDR_PARAMEN</b> 0: Disable Display RAM timing parameter registers 1: Enable Display RAM timing parameter registers	0	RW
1	<b>Enter Power Saving Mode, SDR_PSAVING</b> 0 to 1 transition will enter power saving mode 1 to 0 transition will exit power saving mode	0	RW
0	<b>Start SDRAM Initialization Procedure, SDR_INITDONE</b> 0 to 1 transition will execute Display RAM initialization procedure. Read value '1' means Display RAM is initialized and ready for access. Once it is written as 1, it cannot be rewritten as 0. 1 to 0 transition: No operation. "Write 1" will execute Display RAM initialization procedure.	0	RW

**Note:** The following Display RAM Timing Registers (REG[E0h-E3h]) are available only when SDR\_PARAMEN (REG[E4] bit2) is set to 1.

## REG[E0h] SDRAM Timing Parameter 1

Bit	Description	Default	Access
7-4	NA	0	RO
3-0	<b>Time from Load Mode Command to Active/Refresh Command (<math>T_{MRD}</math>)</b> 0000b: 1 SDRAM Clock 0001b: 2 SDRAM Clock 0010b: 3 SDRAM Clock : : 1111b: 16 SDRAM Clock	2	RW

## REG[E1h] SDRAM Timing Parameter 2

Bit	Description	Default	Access
7-4	<b>Auto Refresh Period, <math>T_{RFC}</math></b> 0h – Fh: 1 ~ 16 SDRAM Clock (As REG[E0h] bit[3:0])	8	RW
3-0	<b>Time of Exit SELF Refresh-to-ACTIVE Command (<math>T_{XSR}</math>)</b> 0h – Fh: 1 ~ 16 SDRAM Clock	7	RW

## REG[E2h] SDRAM Timing Parameter 3

Bit	Description	Default	Access
7-4	<b>Time of Pre-charge Command Period (<math>T_{RP}</math>, 15/20ns)</b> 0h – Fh: 1 ~ 16 SDRAM Clock	2	RW
3-0	<b>Time of WRITE Recovery Time (<math>T_{WR}</math>)</b> 0h – Fh: 1 ~ 16 SDRAM Clock	0	RW

## REG[E3h] SDRAM Timing Parameter 4

Bit	Description	Default	Access
7-4	<b>Delay Time of Active-to-Read/Write (<math>T_{RCD}</math>)</b> 0h – Fh: 1 ~ 16 SDRAM Clock	2	RW
3-0	<b>Time of Active-to-Precache (<math>T_{RAS}</math>)</b> 0h – Fh: 1 ~ 16 SDRAM Clock	6	RW

### 19.13 I2C Master Register

**REG[E5h-E6h] I2C Master Clock Prescaler Register (I2CMCK)**

Bit	Description	Default	Access
7-0	<b>I2C Master Clock Pre-scaler [15:0]</b> REG[E5h] mapping to I2CMCK[7:0]. REG[E6h] mapping to I2CMCK[15:8].	0	RW

**REG[E7h] I2C Master Transmit Register (I2CMTXR)**

Bit	Description	Default	Access
7-0	<b>I2C Master Transmit [7:0]</b>	0	RW

**REG[E8h] I2C Master Receiver Register (I2CMRXR)**

Bit	Description	Default	Access
7-0	<b>I2C Master Receiver [7:0]</b>	0	RW

**REG[E9h] I2C Master Command Register (I2CMCMD)**

Bit	Description	Default	Access
7	<b>Start Command</b> Write 1: Generate (repeated) start condition and be cleared by hardware automatically <b>Note:</b> This bit is always read as 0.	0	RW
6	<b>Stop Command</b> Write 1: Generate stop condition and be cleared by hardware. <b>Note:</b> This bit is always read as 0.	0	RW
5	<b>Read Command</b> Write 1: Read from slave and be cleared by hardware automatically. <b>Note:</b> This bit is always read as 0.	0	RW
4	<b>Write Command</b> Write 1: Write to slave and be cleared by hardware automatically. <b>Note:</b> This bit is always read as 0.	0	RW
3	<b>Acknowledge Command</b> Write 0: Send ACK Write 1: Send NACK <b>Note:</b> This bit is always read as 0.	0	RW
2-1	NA	0	RO

Bit	Description	Default	Access
0	<b>Noise Filter</b> 0: Disable 1: Enable	0	RW

**REG[EAh] I2C Master Status Register (I2CMST)**

Bit	Description	Default	Access
7	<b>Received Acknowledge from Slave</b> 0: Acknowledge received. 1: No Acknowledge received.	0	RO
6	<b>I2C Bus is Busy</b> 0: Idle. '0' after STOP signal detected 1: Busy. '1' after START signal detected	0	RO
5-2	NA	0	RO
1	<b>I2C Transfer in Progress</b> 0: when transfer complete 1: when transferring data	0	RO
0	<b>Arbitration Lost State</b> When LT768x lost Arbitration, this bit will be set to 1. <b>Note:</b> When a Stop signal is detected, but it is not required, then it means arbitration loss. The LT768x I2C Master will drive SDA to 1, but the other Master will drive SDA to 0.	0	RO

## 19.14 GPIO Register

**REG[F0h] GPIO-A Direction (GPIOAD)**

Bit	Description	Default	Access
7-0	<b>PIOA In/Out Control</b> 0: Output. 1: Input.	FFh	RW

**REG[F1h] GPIO-A (GPIOA)**

Bit	Description	Default	Access
7-0	<b>GPIOA Data</b> Write: Output Data to GPIOA Read: Read Data from GPIOA GPIOA[7:0] are General Purpose I/O. These signals are shared with DB[15:8]. They are available only when Host is in 8bits parallel or Serial mode.	NA	RW

**REG[F2h] GPIO-B (GPIOB)**

Bit	Description	Default	Access
7-5	NA	NA	NA
4	<b>GPIOB[4] Data</b> Write: Output Data to GPOB[4] Read: Read Data from GPIB[4] GPOB[4] is shared with KO[0], GPIB[4] is shared with KI[0].	NA	RW
3-0	<b>GPIB[3:0] Data</b> Read: Read Data from GPIB[3:0] GPIB[3:0] are shared with { A0, WR#, RD#, CS# }. They are available only in Serial Host I/F.	NA	RO

**REG[F3h] GPIO-C Direction (GPIOCD)**

Bit	Description	Default	Access
7-0	<b>GPIOC In/Out Control</b> 0: Output 1: Input.	FFh	RW

## REG[F4h] GPIO-C (GPIOC)

Bit	Description	Default	Access
7	<b>GPIOC[7] Data</b> Write: Output Data to GPIOC[7] Read: Read Data from GPIOC[7] GPIOC[7] is shared with PWM[0]. GPIOC is available only when PWM and SPI Master functions are disabled.	NA	RW
6-5	NA	NA	NA
4-0	<b>GPIOC[4:0] Data</b> Write: Output Data to GPIOC[4:0] Read: Read Data from GPIOC[4:0] GPIOC[4:0] are shared with { SFCS1#, SFCS0#, SFDI, SFDO, SFCLK }. They are available when PWM and SPI Master functions are disabled.	NA	RW

## REG[F5h] GPIO-D Direction (GPIODD)

Bit	Description	Default	Access
7-0	<b>GPIOD In/Out Control</b> 0: Output 1: Input	FFh	RW

## REG[F6h] GPIO-D (GPIOD)

Bit	Description	Default	Access
7-0	<b>GPIOD Data</b> Write: Output Data to GPIOD[7:0] Read: Read Data from GPIOD[7:0] GPIOD[7:0] are shared with PD[18, 2, 17, 16, 9, 8, 1, 0]. GPIOD[5,4,1,0] are available when LCD panel data bus is 16 or 12bits. GPIOD[7,6,3,2] are available when LCD panel data bus is 16bits.	NA	RW

REG[F7h-FAh]: Reserved.

### 19.15 Keypad-scan Control Register

**REG[FBh] Keypad-scan Control Register 1 (KSCR1)**

Bit	Description	Default	Access
7	Must set as 0.	0	0
6	<b>Long Key Enable</b> 1: Enable. Long key cycle is set by KSCR2 bit[4:2]. 0: Disable.	0	RW
5-4	<b>Short Key de-bounce Times</b> De-bounce times of keypad scan frequency. 00b: 4 01b: 8 10b: 16 11b: 32	0	RW
3	<b>Repeatable Key Enable</b> 0: Disable Repeatable Key 1: Enable Repeatable Key i.e., if a key is always pressed, and Long Key is disabled, then controller will issue key interrupt repeatedly in every short key de-bounce time. On the other hand, if Long Key is enabled , then controller will issue key interrupt in every long key recognition time. Please note once an interrupt is issued, MCU needs to clear the interrupt flag so that the next interrupt can be recognized.	0	RW
2-0	<b>Row Scan Time</b> $T_{KEYCLK} = (1/F_{SYSCLK}) * 2,048$ 000b: Row_Scan_Time = $T_{KEYCLK}$ 001b: Row_Scan_Time = $T_{KEYCLK} * 2$ 010b: Row_Scan_Time = $T_{KEYCLK} * 4$ 011b: Row_Scan_Time = $T_{KEYCLK} * 8$ 100b: Row_Scan_Time = $T_{KEYCLK} * 16$ 101b: Row_Scan_Time = $T_{KEYCLK} * 32$ 110b: Row_Scan_Time = $T_{KEYCLK} * 64$ 111b: Row_Scan_Time = $T_{KEYCLK} * 128$ LT768x's Keypad-scan controller supports 5x5 keys. Total Key pads scan time = Row Scan Time * 5.	0	RW

**REG[FCh] Keypad-scan Controller Register 2 (KSCR2)**

Bit	Description	Default	Access
7	<b>Keypad-scan Wakeup Enable</b> 0: Keypad-scan Wakeup function is disabled. 1: Keypad-scan Wakeup function is enabled	0	R/W
6	<b>Key Released Interrupt Enable</b>	0	RW

Bit	Description	Default	Access
	0: Without interrupt event when all key released 1: Generate an interrupt when all key released		
5	NA	NA	NA
4-2	<b>Long Key Recognition Factor</b> It determines long key recognition time. Value from 0 to 7. Please note that short key will be recognized before long key is recognized. <b>LongKey Recognition Time</b> $= \text{RowScanTime} * 5 * (\text{Long Key Recognition Factor} + 1) * 1,024$	0	RW
1-0	<b>Numbers of Key Hit</b> 00b: No key is pressed 01b: One key is pressed, REG[FDh] for the key code. 10b: Two keys are pressed, REG[FEh] for the 2nd key code. 11b: Three keys are pressed, REG[FFh] for the 3rd key code. These bits will automatically return to 00h if there is no key pressed within a de-bounce time.	0	RO

**REG[FDh] Keypad-scan Data Register (KSDR1)**

Bit	Description	Default	Access
7-0	<b>Key Strobe Data1</b> The corresponding key code 1 that is pressed. These bit will automatically return to FFh if there is no key pressed within a de-bounce time.	TBD	RO

**REG[FEh] Keypad-scan Data Register (KSDR2)**

Bit	Description	Default	Access
7-0	<b>Key Strobe Data2</b> The corresponding key code 2 that is pressed. These bits will automatically return to FFh if there is no key pressed within a de-bounce time.	TBD	RO

**REG[FFh] Keypad-scan Data Register (KSDR3)**

Bit	Description	Default	Access
7-0	<b>Key Strobe Data3</b> The corresponding key code 3 that is pressed. These bits will automatically return to FFh if there is no key pressed within a de-bounce time.	TBD	RO

## 20. Package Information

### 20.1 LT7681/LT7683/LT7686 (LQFP-128pin)

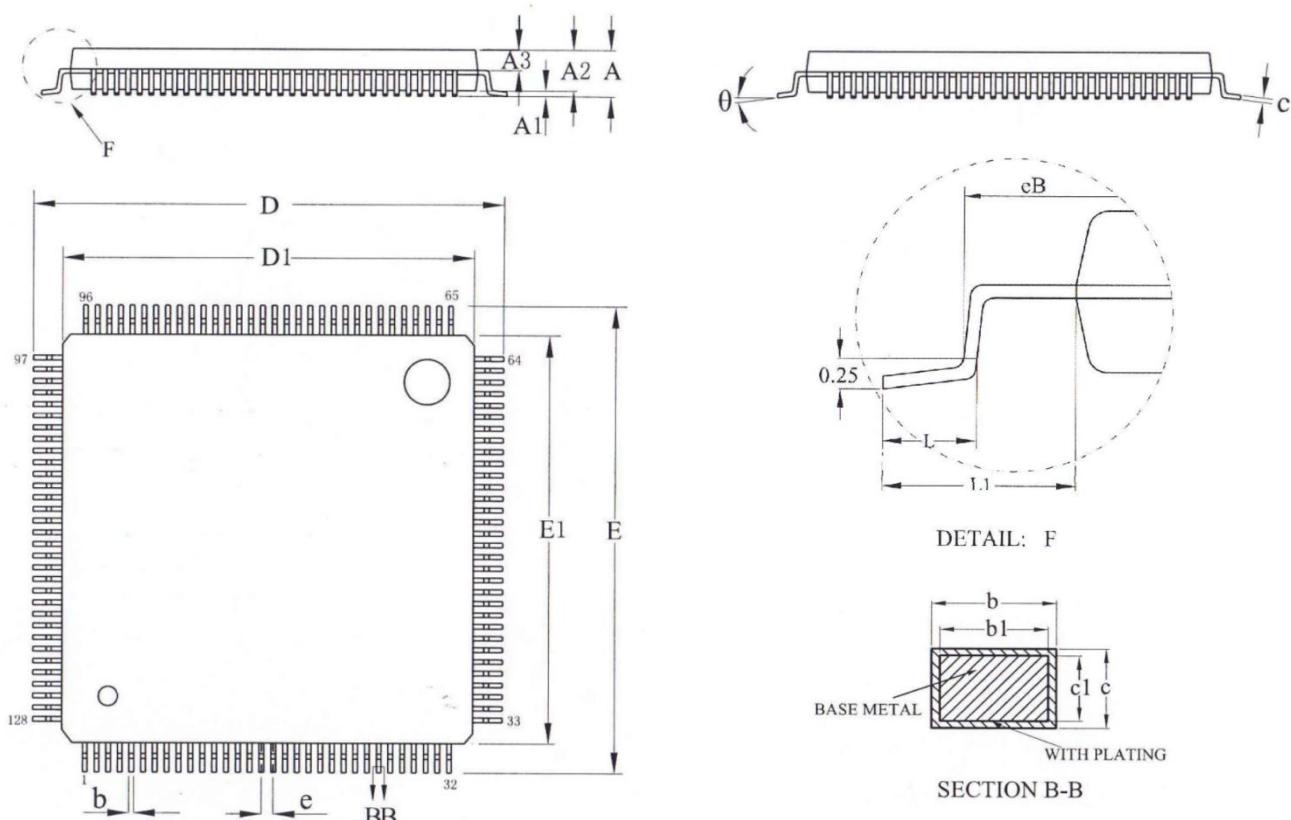


Figure 20-1: 128Pin LQFP Outline

Table 20-1: 128Pin LQFP Dimension

Symbol	Millimeter			Symbol	Millimeter		
	Min.	Nom.	Max		Min.	Nom.	Max
A	-	-	1.60	D1	13.9	14.0	14.1
A1	0.05	-	0.15	E	15.8	16.0	16.2
A2	1.35	1.40	1.45	E1	13.9	14.0	14.1
A3	0.59	0.64	0.69	eB	15.05	-	15.35
b	0.14	-	0.22	e	0.40BSC		
b1	0.13	0.16	0.19	L	0.45	-	0.75
c	0.13	-	0.17	L1	1.00REF		
c1	0.12	0.13	0.14	θ	0		7
D	15.8	16.00	16.2				

## 20.2 LT7680 (QFN-68pin)

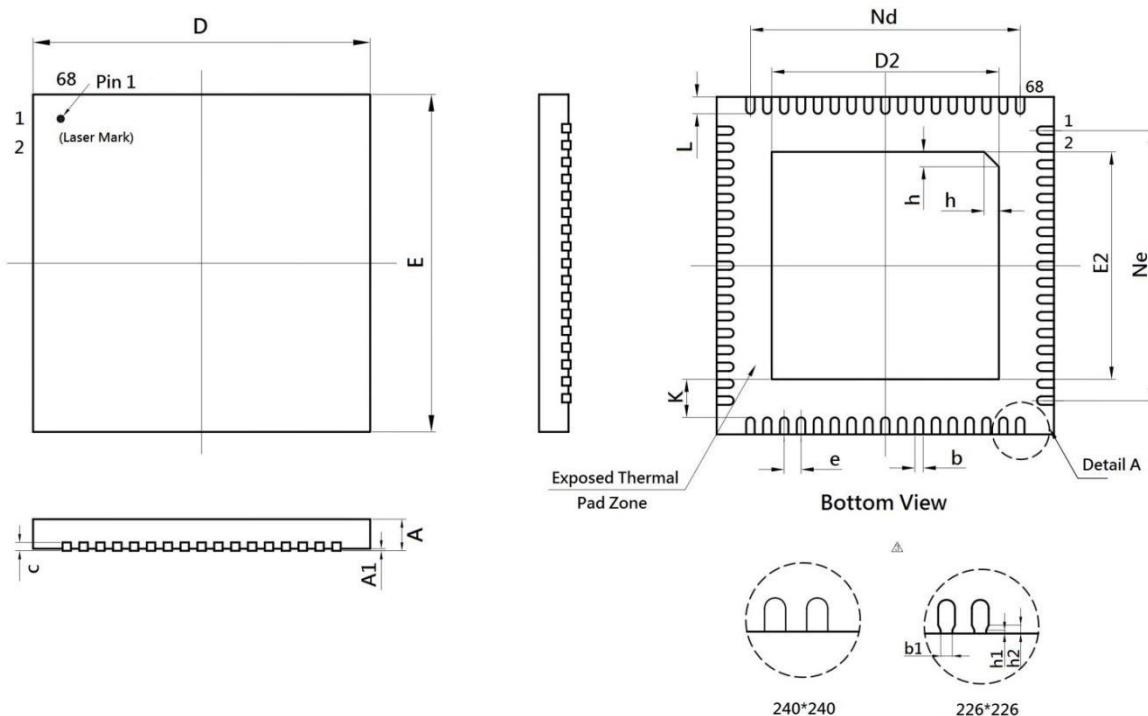


Figure 20-2: 68Pin QFN Outline

**Note:** When PCB layout, the heat pad of LT7680's back (thermal Pad Zone) must be directly grounded.

Table 20-2: 68Pin QFN Dimension

Symbol	Millimeter			Symbol	Millimeter		
	Min.	Nom.	Max		Min.	Nom.	Max
A	0.70	0.75	0.8	E	7.9	8.0	8.10
A1	-	0.02	0.05	Ne	6.40BSC		
b	0.15	0.20	0.25	L	0.35	0.40	0.45
b1	0.14REF			K	0.20	-	-
c	0.18	0.20	0.25	h	0.30	0.35	0.40
D	7.90	8.00	8.10	h1	0.04REF		
e	0.40BSC			h2	0.10REF		
Nd	6.40BSC						

Table 20-3: Lead Frame Dimension

L/F 载体尺寸	Symbol	Millimeter	L/F Dimension	Symbol	Millimeter
240*240	D2	5.49+/- 0.10	226*226	D2	5.39+/- 0.10
	E2	5.49+/- 0.10		E2	5.39+/- 0.10

## 21. Revision

**Table 21-1: Revision**

Version	Date	Description
V1.0	2017/11/01	LT768x Preliminary Release
V1.1	2018/01/26	<ul style="list-style-type: none"> <li>1. Modify Table 7-2, Table 6-3, Table 18-1 Note, REG[0Bh], REG[0Ch], REG[0Dh].</li> <li>2. Modify Section 1.1, 1.2</li> <li>3. Modify LT7680B Embedded Display RAM Size.</li> </ul>
V2.0	2018/05/01	<ul style="list-style-type: none"> <li>1. Modify Section 1.1 Clock design guidelines of OD (Page27) and three clock signals design rule (Page28)</li> <li>2. Remove all product information about LT7685.</li> </ul>
V3.0	2019/08/03	<ul style="list-style-type: none"> <li>1. Update Table 1-1: Model Selection</li> <li>2. Table 8-1: LT768x' Model vs. Embedded Display RAM Capacity</li> </ul>
V4.0	2020/07/10	Update LT7680 MCU's I2C Interface to 4-Wires SPI.
V4.1	2021/07/14	Update pdf file format.
V4.2	2022/02/06	Add ESD and Latch-up testing data

## 22. Copyright

This document is the copyright of Levetop Semiconductor Co., Ltd. No part of this document may be reproduced or duplicated in any form or by any means without the prior permission of Levetop. The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Levetop assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Levetop makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Levetop's products are not authorized for use as critical components in life support devices or systems. Levetop reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at <https://www.levetop.tw/en/index.html>