



LT7381

High Performance TFT-LCD Graphics Controller

Data Sheet

V1.0

www.levetop.cn

Levetop Semiconductor Co., Ltd.

Contents

| | |
|---|----|
| ▶ Introduction | 6 |
| ▶.. Internal Block Diagram | 6 |
| ▶.. System Block Diagram | 7 |
| ▶.. Model Name | 7 |
| ▶.. Pin Assignment | 7 |
| ▶.. Features | 8 |
| ▶.. Pin Description..... | 10 |
| ▶.. Absolute Maximum Ratings | 17 |
| ▶.. Electrical Characteristics | 17 |
| ▶.. Function Description | 19 |
| 1. Clock and Reset..... | 19 |
| 1.1 Clock..... | 19 |
| 1.2 Reset..... | 22 |
| 1.2.1 Power-on Reset..... | 22 |
| 1.2.2 External Reset..... | 22 |
| 1.2.3 Software Reset..... | 22 |
| 2. Host Interface..... | 23 |
| 2.1 Parallel Host Interface..... | 25 |
| 2.2 Serial Host Interface..... | 28 |
| 2.3 Display Input Data Format..... | 32 |
| 2.3.1 Input Data without Opacity (RGB)..... | 32 |
| 2.3.2 Input Data with Opacity (α RGB)..... | 35 |
| 3. Display Memory..... | 37 |
| 3.1 Display RAM Data Structure..... | 38 |
| 3.1.1 8bpp Display Data (RGB 3:3:2)..... | 38 |
| 3.1.2 16bpp Display Data (RGB 5:6:5)..... | 38 |
| 3.1.3 24bpp Display Data (RGB 8:8:8)..... | 38 |
| 3.1.4 Index Display with Opacity (α RGB 2:2:2)..... | 39 |
| 3.1.5 12bpp Display with Opacity (α RGB 4:4:4)..... | 39 |
| 3.2 Color Palette RAM..... | 39 |
| 4. LCD Interface..... | 40 |
| 5. Display Function..... | 42 |

| | | |
|--------|---|----|
| 5.1 | Color Bar..... | 42 |
| 5.2 | Main Window..... | 42 |
| 5.2.1 | Configure Display Image Buffer..... | 42 |
| 5.2.2 | Write Image Data to Display Image Buffer..... | 43 |
| 5.2.3 | Display Main Window Image..... | 44 |
| 5.3 | Picture-In-Picture (PIP)..... | 45 |
| 5.3.1 | PIP Window Setting..... | 45 |
| 5.3.2 | PIP Display Position and PIP Image Position..... | 46 |
| 5.4 | Image Rotate and Mirror..... | 47 |
| 6. | Geometric Drawing Engine..... | 50 |
| 6.1 | Drawing Circle and Ellipse..... | 50 |
| 6.2 | Drawing Curve..... | 51 |
| 6.3 | Drawing Rectangle..... | 52 |
| 6.4 | Draw Line..... | 53 |
| 6.5 | Drawing Triangle..... | 54 |
| 6.6 | Drawing Rounded-Rectangle..... | 55 |
| 7. | Block Transfer Engine (BTE)..... | 56 |
| 7.1 | BTE Basic Settings..... | 58 |
| 7.2 | Color Palette RAM..... | 59 |
| 7.3 | BTE Operation Overview..... | 60 |
| 7.3.1 | MCU Write with ROP..... | 60 |
| 7.3.2 | Memory Copy with ROP..... | 60 |
| 7.3.3 | Solid Fill..... | 60 |
| 7.3.4 | Pattern Fill..... | 60 |
| 7.3.5 | Pattern Fill with Chroma Key..... | 60 |
| 7.3.6 | MCU Write with Chroma Key..... | 60 |
| 7.3.7 | Memory Copy with Chroma Key..... | 60 |
| 7.3.8 | MCU Write with Color Expansion..... | 60 |
| 7.3.9 | Memory Copy with Color Expansion..... | 61 |
| 7.3.10 | Memory Copy with Opacity..... | 61 |
| 7.3.11 | MCU Write with Opacity..... | 61 |
| 7.4 | BTE Memory Access Method..... | 62 |
| 7.5 | BTE Chroma Key (Transparency Color) Function..... | 62 |
| 7.6 | BTE Operation Detail..... | 63 |
| 7.6.1 | MCU Write with ROP..... | 63 |
| 7.6.2 | Memory Copy (move) with ROP..... | 64 |
| 7.6.3 | MCU Write with Chroma Key (w/o ROP)..... | 66 |
| 7.6.4 | Memory Copy with Chroma Key (w/o ROP)..... | 67 |

| | | |
|------------|--|------------|
| 7.6.5 | Pattern Fill with ROP..... | 68 |
| 7.6.6 | Pattern Fill with Chroma Key..... | 69 |
| 7.6.7 | MCU Write with Color Expansion..... | 70 |
| 7.6.8 | MCU Write with Color Expansion and Chroma key..... | 73 |
| 7.6.9 | Memory Copy with Opacity..... | 74 |
| 7.6.10 | MCU Write with Opacity..... | 78 |
| 7.6.11 | Memory Copy with Color Expansion..... | 79 |
| 7.6.12 | Memory Copy with Color Expansion and Chroma Key..... | 81 |
| 7.6.13 | Solid Fill..... | 82 |
| 8. | Display Text..... | 83 |
| 8.1 | Internal CGROM..... | 83 |
| 8.2 | User-defined Character Graphic (UCG)..... | 86 |
| 8.2.1 | 8*16 UCG Data Format..... | 86 |
| 8.2.2 | 16*16 UCG Data Format..... | 87 |
| 8.2.3 | 12*24 UCG Data Format..... | 88 |
| 8.2.4 | 24*24 UCG Data Format..... | 89 |
| 8.2.5 | 16*32 UCG Data Format..... | 90 |
| 8.2.6 | 32*32 UCG Data Format..... | 91 |
| 8.2.7 | Initialize CGRAM from MCU..... | 92 |
| 8.2.8 | Initialize CGRAM from Serial Flash..... | 93 |
| 8.3 | Character Rotation by 90 Degree..... | 94 |
| 8.4 | Size Enlargement..... | 94 |
| 8.5 | Background Transparency..... | 95 |
| 8.6 | Character Full-Alignment..... | 95 |
| 8.7 | Automatic Line Feed..... | 96 |
| 8.8 | Cursor..... | 96 |
| 8.8.1 | Text Cursor..... | 96 |
| 8.8.2 | Graphic Cursor..... | 98 |
| 9. | Pulse Width Modulation (PWM)..... | 100 |
| 9.1 | PWM Clock Source..... | 100 |
| 9.2 | PWM Output..... | 101 |
| 10. | I2C Master..... | 103 |
| 11. | Keypad-Scan..... | 106 |
| 11.1 | Keypad-scan Operation..... | 106 |
| 12. | GPIO Interface..... | 110 |

| | |
|---|-----|
| 13.Power Management..... | 111 |
| 13.1 Normal Mode..... | 111 |
| 13.2 Standby Mode..... | 111 |
| 13.3 Suspend Mode..... | 112 |
| 13.4 Sleep Mode..... | 112 |
| 14.Register Description..... | 113 |
| 14.1 Status Register..... | 113 |
| 14.2 Configuration Registers..... | 115 |
| 14.3 PLL Setting Register..... | 120 |
| 14.4 Interrupt Control Register..... | 122 |
| 14.5 LCD Display Control Registers..... | 127 |
| 14.6 Geometric Engine Control Registers..... | 138 |
| 14.7 PWM Control Registers..... | 147 |
| 14.8 Bit Block Transfer Engine (BTE) Control Registers..... | 150 |
| 14.9 Text Engine Registers..... | 158 |
| 14.10 Power Management Control Register..... | 163 |
| 14.11 Display RAM Control Register..... | 164 |
| 14.12 I2C Master Register..... | 168 |
| 14.13 GPIO Register..... | 170 |
| 14.14 Keypad-scan Control Register..... | 172 |
| ▶.. Package Information | 174 |
| ▶.. Revision | 175 |
| ▶.. Copyright | 175 |

High Performance TFT-LCD Graphics Controller

Introduction

LT7381 is a high-performance TFT-LCD graphics accelerated display chip. Its main function is to assist MCU to display the contents of the TFT screen to the TFT Driver. It provides graphics acceleration, PIP (picture-in-picture), geometry graphics and other functions, in addition to enhance the display efficiency, also greatly reduces the MCU processing graphics display time spent. LT7381 also supports a very broad display resolution, can be from 320*240(QVGA) to 1024*768(SXGA), the display is supported 16/18/24bits RGB interface.



LT7381 supports a variety of MCU interface, including SPI, I2C serial port, or 8-bit, 16-bit parallel interface. In order to achieve multi-layers high-resolution display effect, LT7381 embedded a 32MB Display RAM

This Displays Memory can support 16M color display from 2 gray to 1bit per pixel to up to 24bits per pixel. At the same time to reduce the animation display MCU in the software operation burden. LT7381 built-in geometry drawing engine, supporting the painting point, drawing Line, drawing curve, ellipse, triangle, rectangle, rounded rectangle and other functions. The embedded hardware graphics Acceleration Engine (BTE) provides the command type of graphics operations, such as display rotation, mirror shot, the painting (PIP/Master-Sub Screen) and graphics mixed transparent display and other functions, enhance the display of the product performance, so can greatly reduce the MCU software operating burden. if use the high-speed SPI interface, then it can reduce the MCU I/O port needs, without to upgrade the MCU for TFT display. LT7381 powerful display function is very suitable for the electronic products with TFT-LCD screen, such as home appliances, multi-functional business machines, industrial equipment, industrial control, electronic equipment, medical equipment, human-computer interface, testing equipment and other products.

Internal Block Diagram

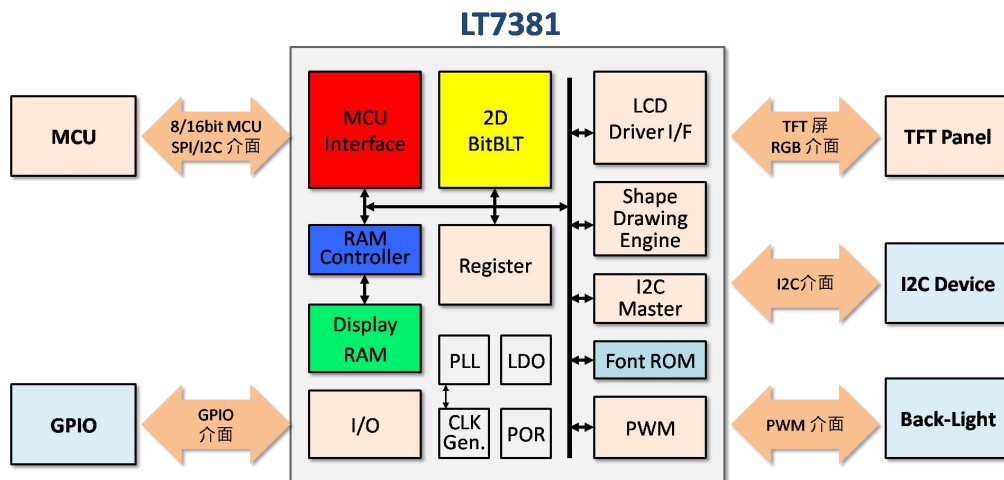


Figure A-1: Internal Block Diagram

System Block Diagram

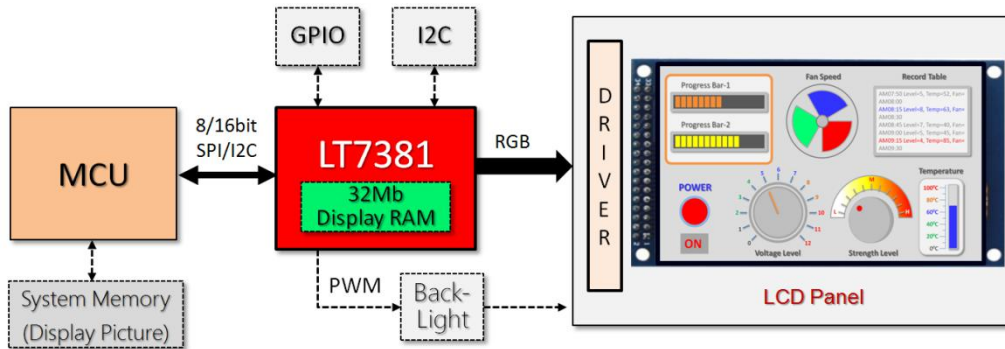


Figure A-2: LT7381 Designed on System Board

Model Name

Table A-1: Model Selection

| Model Name | Package | Embedded Display RAM | Resolution | Colors |
|------------|----------|----------------------|------------|--------|
| LT7381 | LQFP-128 | 32Mb | 1024*768 | 16.7M |

Pin Assignment

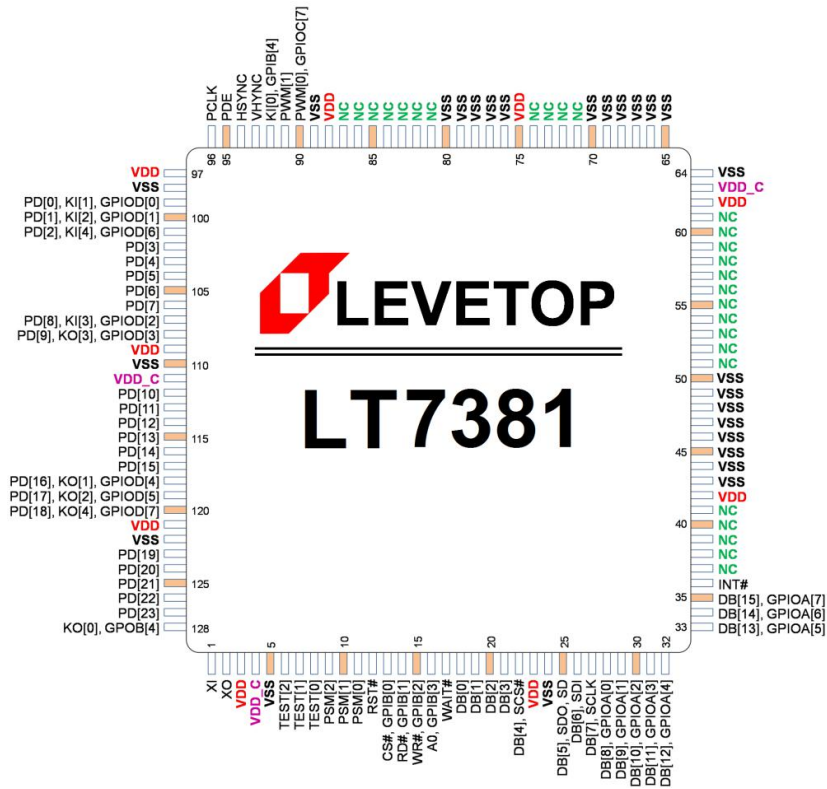


Figure A-3: LT7381 Pin Assignment (LQFP-128Pin)

Features

Host Interface

- Support Three Types 8/16bits Asynchronous Bus Register Interface (or Memory Data)
 - Indirect Intel-80 Bus Interface
 - Indirect Motorola-6800 Bus Interface
- Provides Insert Wait State Mechanism on Parallel Host Cycle
- Support I²C 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)
- αRGB 4:4:4 (4096 Colors/Pixel with Opacity Attribute)

Support Panel and Resolution

- Support 16/18/24-bits RGB Interface Type Panel
- Supported 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

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 enable

- 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: Supported 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 show the display data quickly which data is 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 and use to show display data which stored in the serial flash and need not external MPU participate. It will auto execute after power-on, until program execute complete then handover control rights to external MCU
- 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 transparency rate
 - Window Alpha-Blending Function: Mixes two images according to transparency rate in the specified region (fade-in and fade-out functions are available)
 - Dot Alpha-Blending Function: Mixes images according to transparency rate when the target is a graphics image in the RGB format

Display RAM (Frame Buffer)

- Embedded 32Mb Display RAM

Shape Drawing Engine

- Provide Smart Drawing Features: Line, Rectangle, Triangle, Polygon, Poly-Line, Circle, Ellipse, Arc, Rounded-Rectangle and Circle-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 for Writing with Character
- Character Enlargement Function *1, *2, *3, *4 for Horizontal/Vertical Direction
- Support Character Rotates 90 Degree

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

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

I2C Interface

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

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 128-Pins

Temperature

- -40°C~85°C

Pin Description

Host Interface Select Signals (3 Pins)

Table A-2: Host I/F Select Signals

| Pin # | Pin Name | I/O | Pin Description | | | | | | | | | | | | |
|--|-----------------|-----|---|--|---------------|-------|--|-------|--|-------|-----------------|-------|-----------------|-------|----------|
| 9~11 | PSM[2:0] | I | Host Interface Selection | | | | | | | | | | | | |
| | | | <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> | 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 | | | | | | | | | | | | | | |
| If Host interface set as parallel mode, then PSM[0] pin is external interrupt input pin. | | | | | | | | | | | | | | | |

Host Parallel I/F Signals (22 Pins)

Table A-3: Host Parallel I/F Signals

| Pin # | Pin Name | I/O | Pin Description |
|-----------------|----------------------|-----|---|
| 35~25, 22~18 | DB[15:0] | IO | Host Data Bus These are data bus for data transfer between Host and LT7381. DB[15:8] will become GPIO (GPIOA[7:0]) when parallel Host 8080/6800 16-bits data bus mode doesn't set. DB[7:0] are multiplex pins that share with Serial Host control pins. When serial host mode set then DB[7:0] are defined as the control pins of serial host. Please refer to Host Interface section. |
| 13 | CS# GPIB[0] | I | Chip Select Input Low active chip select pin from Host. If host interface set as serial host mode, then this pin can be set as GPIB[0]. This pin with an internal pull-high resistor. |
| 14 | RD# EN GPIB[1] | I | Read / Enable Input RD#: When host interface is 8080 mode then this is a Read input signal, active low. EN: When interface is 6800 mode then this is a Enable input signal, active high. If host interface set as serial host mode then this pin can be set as GPIB[1]. This pin with an internal pull-high resistor. |

Table A-3: Host MCU Parallel I/F Signals (Continued)

| Pin # | Pin Name | I/O | Pin Description |
|-------|-----------------------|-----|---|
| 15 | WR# RW# GPIB[2] | I | <p>Write / Read-Write Input</p> <p>WR#: When host interface is 8080 mode then this is a Write input signal, active low.</p> <p>RW#: When interface is 6800 mode then this is a Read-Write input signal. It active high in 'Host's read cycle, and active low in Host's write cycle.</p> <p>If host interface set as serial host mode then this pin will be set as GPIB[2]. This pin with an internal pull-high resistor.</p> |
| 16 | A0 GPIB[3] | I | <p>Command / Data Select Input</p> <p>The pin is used to select Command or Data cycle.</p> <p>A0 = 0, Status Read or Command Write cycle is selected.</p> <p>A0 = 1, Data Read or Data Write cycle is selected.</p> <p>If host interface set as serial host mode then this pin will be set as GPIB[3]. This pin with an internal pull-high resistor.</p> |
| 36 | INT# | O | <p>Interrupt Output Signal</p> <p>The interrupt output for host to indicate the status.</p> |
| 17 | WAIT# | O | <p>Wait Output Signal</p> <p>When high, it indicates that the LT7381 is ready to transfer data. When low, then microprocessor is in wait state.</p> |

MCU Serial I/F Signals (8 Pins)
Table A-4: Host Serial I/F Signals

| Pin # | Pin Name | I/O | Pin Description |
|-------|---------------------------------|-----|---|
| 27 | SCLK (DB[7]) | I | <p>SPI or I2C Clock</p> <p>SCLK: Clock of 3-wire, 4-wire Serial or I2C interface.</p> <p>This is a multiplex pin that share with Parallel Host Data Bus DB[7].</p> |
| 26 | SDI I2C_SDA (DB[6]) | I | <p>I2C Data / 4-wire SPI Data Input</p> <p>SDI: Data input pin of 4-wire SPI I/F. Connect to MCU's MOSI.</p> <p>I2C_SDA: Bi-direction data pin of I2C I/F.</p> <p>This pin is not used In 3-Wire serial I/F. Please connect it to GND. This is a multiplex pin that share with Parallel Host Data Bus DB[6].</p> |
| 25 | SD SDO I2CA[5] (DB[5]) | IO | <p>3-wire SPI Data / 4-wire SPI Data Output / I2C Slave Address Select</p> <p>SD: Bi-direction data pin of 3-wire SPI I/F.</p> <p>SDO: Data output pin of 4-wire SPI I/F. Connect to MCU's MISO.</p> <p>I2CA[5]: I2C device address bit[5] of I2C I/F.</p> <p>This is a multiplex pin that share with Parallel Host Data Bus DB[5].</p> |

Table A-4: Host Serial I/F Signals (Continued)

| Pin # | Pin Name | I/O | Pin Description |
|-------|----------------------------|-----|--|
| 22 | SCS# I2CA[4] (DB[4]) | I | SPI Chip Select / I2C Slave Address Select SCS#: Chip select pin for 3-wire or 4-wire serial I/F. I2CA[4]: I2C device address bit[4]. This is a multiplex pin that share with Parallel Host Data Bus DB[4]. |
| 21~18 | I2CA[3:0] (DB[3:0]) | I | I2C Slave Address Select I2CA[3:0]: 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]. |

PWM Output Signals (2 Pins)
Table A-5: PWM Output Signals

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

LCD Driver Signals (28 Pins)

Table A-6: LCD Driver Signals

| Pin # | Pin Name | I/O | Pin Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|----------|-----|--|-------------------|-------------------|--------------|--------------|--|------------|--------------|--------------|--------------|-------|------------------|--|--|----|-------|------------------|--|--|----|-------|------------------|--|----|----|-------|----------|----|----|----|-------|----------|----|----|----|-------|----------|----|----|----|-------|----------|----|----|----|-------|----------|----|----|----|-------|------------------|--|--|----|-------|------------------|--|--|----|--------|----------|----|----|----|--------|----------|----|----|----|--------|----------|----|----|----|--------|----------|----|----|----|--------|----------|----|----|----|--------|----------|----|----|----|--------|------------------|--|--|----|--------|-------------------|--|--|----|--------|------------------|--|----|----|--------|----------|----|----|----|--------|----------|----|----|----|--------|----------|----|----|----|--------|----------|----|----|----|--------|----------|----|----|----|
| 127~123, 120~112, 108~99 | PD[23:0] | IO | <p>LCD Panel Data Bus</p> <p>TFT LCD data bus output for source driver. LT76x supports 64K/256K/16.7M color depth by register setting; user can connect corresponding RGB bus for different setting.</p> <table border="1" style="margin: 10px auto;"> <thead> <tr> <th rowspan="2">Pin Name</th> <th colspan="4">TFT-LCD Interface</th> </tr> <tr> <th>11b (GPIO)</th> <th>10b (16bits)</th> <th>01b (18bits)</th> <th>00b (24bits)</th> </tr> </thead> <tbody> <tr> <td>PD[0]</td> <td colspan="3">GPIOD[0] / KI[1]</td> <td>B0</td> </tr> <tr> <td>PD[1]</td> <td colspan="3">GPIOD[1] / KI[2]</td> <td>B1</td> </tr> <tr> <td>PD[2]</td> <td colspan="2">GPIOD[6] / KI[4]</td> <td>B0</td> <td>B2</td> </tr> <tr> <td>PD[3]</td> <td>GPIOE[0]</td> <td>B0</td> <td>B1</td> <td>B3</td> </tr> <tr> <td>PD[4]</td> <td>GPIOE[1]</td> <td>B1</td> <td>B2</td> <td>B4</td> </tr> <tr> <td>PD[5]</td> <td>GPIOE[2]</td> <td>B2</td> <td>B3</td> <td>B5</td> </tr> <tr> <td>PD[6]</td> <td>GPIOE[3]</td> <td>B3</td> <td>B4</td> <td>B6</td> </tr> <tr> <td>PD[7]</td> <td>GPIOE[4]</td> <td>B4</td> <td>B5</td> <td>B7</td> </tr> <tr> <td>PD[8]</td> <td colspan="3">GPIOD[2] / KI[3]</td> <td>G0</td> </tr> <tr> <td>PD[9]</td> <td colspan="3">GPIOD[3] / KO[3]</td> <td>G1</td> </tr> <tr> <td>PD[10]</td> <td>GPIOE[5]</td> <td>G0</td> <td>G0</td> <td>G2</td> </tr> <tr> <td>PD[11]</td> <td>GPIOE[6]</td> <td>G1</td> <td>G1</td> <td>G3</td> </tr> <tr> <td>PD[12]</td> <td>GPIOE[7]</td> <td>G2</td> <td>G2</td> <td>G4</td> </tr> <tr> <td>PD[13]</td> <td>GPIOF[0]</td> <td>G3</td> <td>G3</td> <td>G5</td> </tr> <tr> <td>PD[14]</td> <td>GPIOF[1]</td> <td>G4</td> <td>G4</td> <td>G6</td> </tr> <tr> <td>PD[15]</td> <td>GPIOF[2]</td> <td>G5</td> <td>G5</td> <td>G7</td> </tr> <tr> <td>PD[16]</td> <td colspan="3">GPIOD[4] / KO[1]</td> <td>R0</td> </tr> <tr> <td>PD[17]</td> <td colspan="3">GPIOD[5] / KOI[2]</td> <td>R1</td> </tr> <tr> <td>PD[18]</td> <td colspan="2">GPIOD[7] / KO[4]</td> <td>R0</td> <td>R2</td> </tr> <tr> <td>PD[19]</td> <td>GPIOF[3]</td> <td>R0</td> <td>R1</td> <td>R3</td> </tr> <tr> <td>PD[20]</td> <td>GPIOF[4]</td> <td>R1</td> <td>R2</td> <td>R4</td> </tr> <tr> <td>PD[21]</td> <td>GPIOF[5]</td> <td>R2</td> <td>R3</td> <td>R5</td> </tr> <tr> <td>PD[22]</td> <td>GPIOF[6]</td> <td>R3</td> <td>R4</td> <td>R6</td> </tr> <tr> <td>PD[23]</td> <td>GPIOF[7]</td> <td>R4</td> <td>R5</td> <td>R7</td> </tr> </tbody> </table> | Pin Name | 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 | B2 | PD[3] | GPIOE[0] | B0 | B1 | B3 | PD[4] | GPIOE[1] | B1 | B2 | B4 | PD[5] | GPIOE[2] | B2 | B3 | B5 | PD[6] | GPIOE[3] | B3 | B4 | B6 | PD[7] | GPIOE[4] | B4 | B5 | B7 | PD[8] | GPIOD[2] / KI[3] | | | G0 | PD[9] | GPIOD[3] / KO[3] | | | G1 | PD[10] | GPIOE[5] | G0 | G0 | G2 | PD[11] | GPIOE[6] | G1 | G1 | G3 | PD[12] | GPIOE[7] | G2 | G2 | G4 | PD[13] | GPIOF[0] | G3 | G3 | G5 | PD[14] | GPIOF[1] | G4 | G4 | G6 | PD[15] | GPIOF[2] | G5 | G5 | G7 | PD[16] | GPIOD[4] / KO[1] | | | R0 | PD[17] | GPIOD[5] / KOI[2] | | | R1 | PD[18] | GPIOD[7] / KO[4] | | R0 | R2 | PD[19] | GPIOF[3] | R0 | R1 | R3 | PD[20] | GPIOF[4] | R1 | R2 | R4 | PD[21] | GPIOF[5] | R2 | R3 | R5 | PD[22] | GPIOF[6] | R3 | R4 | R6 | PD[23] | GPIOF[7] | R4 | R5 | R7 |
| | | | Pin Name | | 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 | B2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[3] | GPIOE[0] | B0 | B1 | B3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[4] | GPIOE[1] | B1 | B2 | B4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[5] | GPIOE[2] | B2 | B3 | B5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[6] | GPIOE[3] | B3 | B4 | B6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[7] | GPIOE[4] | B4 | B5 | B7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[8] | GPIOD[2] / KI[3] | | | G0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[9] | GPIOD[3] / KO[3] | | | G1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[10] | GPIOE[5] | G0 | G0 | G2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[11] | GPIOE[6] | G1 | G1 | G3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[12] | GPIOE[7] | G2 | G2 | G4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[13] | GPIOF[0] | G3 | G3 | G5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[14] | GPIOF[1] | G4 | G4 | G6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[15] | GPIOF[2] | G5 | G5 | G7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[16] | GPIOD[4] / KO[1] | | | R0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[17] | GPIOD[5] / KOI[2] | | | R1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[18] | GPIOD[7] / KO[4] | | R0 | R2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[19] | GPIOF[3] | R0 | R1 | R3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | PD[20] | GPIOF[4] | R1 | R2 | R4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PD[21] | GPIOF[5] | R2 | R3 | R5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PD[22] | GPIOF[6] | R3 | R4 | R6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PD[23] | GPIOF[7] | R4 | R5 | R7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <p>These are multiplex pins that share with GPIO and Key-Matrix pins. The Default setting of LCD I/F is 18bpp function mode, so PD[17:16 / 8:9 / 1:0] are defined as GPIO pins.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table A-6: LCD Driver Signals (Continued)

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

GPIO Signals (28 Pins)
Table A-7: General Purpose I/O Signals

| Pin # | Pin Name | I/O | Pin Description |
|---|-----------------------------------|-----|---|
| 35~28 | GPIOA[7:0] | IO | GPIO A Group These are general purpose I/O. These are multiplex pins that share with DB[15:8]. They are available when 8bits parallel host mode and serial Host mode. |
| 92, 128, 16~13 | GPIB[4], GPOB[4], GPIB[3:0] | IO | GPIO B Group These are general purpose I/O. GPIB[3:0] are read only and available in serial host mode. GPIB[4] is same pin with KI[0]. GPOB[4] is 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 | GPIO C Group These are general purpose I/O. GPIOC are available when PWM and SPI Master functions disabled. GPIOC[7] is 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 | GPIO D Group 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 16bits or 18bits. GPIOD[7,6] are available when LCD Panel interface is set 16bits. |

Key-Matrix Signals (10 Pins)
Table A-8: Key-Matrix Signals

| Pin # | Pin Name | I/O | Pin Description |
|-----------------------------|----------|-----|---|
| 101, 107, 100, 99, 92 | KI[4:0] | I | Key-Matrix Data Pins Keypad data inputs with internal pull-up resister. 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 are set as 24bits. XKIN[0] also provide the I2CMCK function of I2C Master. |
| 120,108, 119,118, 128 | KO[4:0] | O | Key-Matrix Strobe Pins 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 provide the I2CMDA function of I2C Master. |

Power and Clock Signals (40 Pins)
Table A-9: Power and Clock Signals

| Pin # | Pin Name | I/O | Pin Description |
|---|----------|-----|--|
| 1 | XI | I | Crystal / External Clock Input This input pin is used for internal crystal circuit or external clock that generate clock source for PLL. It should be connected to external crystal or clock, and suggested frequency is 8.0 ~ 12 MHz. |
| 2 | XO | O | Crystal Output This is an output pin for internal crystal circuit. It should be connected to external crystal circuit. |
| 4, 63, 111 | VDD_C | PWR | Internal LDO Output These pins must connect 1uF and 0.1uF capacitor to ground. |
| 3, 23, 42, 62, 75, 88, 97, 109, 121 | VDD | PWR | 3.3V Power Pins |
| 5, 24, 43, 64, 76, 89 98, 110, 122 | VSS | PWR | Ground(GND) Pins |

Reset and Test Signals (4 Pins)**Table A-10: Reset and Test Signals**

| Pin # | Pin Name | I/O | Pin Description |
|-------|-----------|-----|--|
| 12 | RST# | I/O | Reset Signal Input This is a active low Reset pin for LT7381. To avoid noise interfere and cause fake reset behavior, this pin is active at least 256 OSC clocks. |
| 6~8 | TEST[2:0] | I | Test Input These pins are used for testing and normally connect to GND. If TEST[0] keep high, the internal PLL will be disable and the system clock is supply by external. If TEST[2:1] keep 01b, then the SPI Master signals will keep floating. This feature allow external device to program Serial Flash directly. (i.e. ISP, In-System-Programming) |

➤ Absolute Maximum Ratings

Table A-11: Absolute Maximum Ratings

| Symbol | Parameter | Value | Unit |
|------------------|-----------------------------|-----------------------------|------|
| V _{DD} | Supply Voltage Range | -0.3 ~ 4.0 | V |
| V _{IN} | Input Voltage Range | -0.3 ~ V _{DD} +0.3 | V |
| V _{OUT} | Output Voltage Range | -0.3 ~ V _{DD} +0.3 | V |
| P _D | Power Dissipation | ≤300 | mW |
| T _{OPR} | Operation Temperature Range | -45 ~ 85 | °C |
| T _{ST} | Storage Temperature | -45 ~ 125 | °C |
| T _{SOL} | Soldering Temperature | 260 | °C |

Note:

If used beyond the absolute maximum ratings, LT7381 may be permanently damaged. It is strongly recommended that the device be used within the electrical characteristics in normal operations. 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.

➤ Electrical Characteristics (Condition: V_{DD} = 3.3V, T_A = 25°C)

Table A-12: Electrical Characteristics

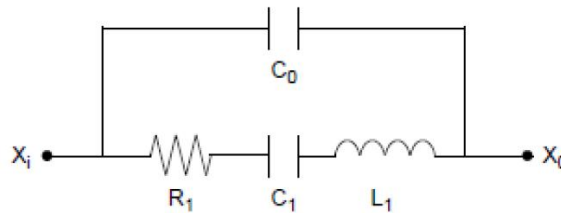
| Symbol | Parameter | Condition | Min. | Typ. | Max. | Unit |
|---------------------|----------------------------|----------------------------------|------|------|------|------|
| V _{DD} | System Voltage | | 3.0 | 3.3 | 3.6 | V |
| C _{VDD} | Loading Capacitor | | 1 | - | 10 | uF |
| I _{OPR} | Operation Current | Note 1 | | 60 | | mA |
| I _{STB} | Standby Mode | Note 1 | | 30 | | mA |
| I _{SUSP} | Suspend Mode | Note 1 | | 10 | | mA |
| I _{SLP} | Sleep Mode | Note 1 | | 7 | | mA |
| T _{RMP} | Power Ramp Up Time | V _{DD} Ramp Up to 3.3 V | 3.5 | | 35 | ms |
| OSC / PLL | | | | | | |
| F _{OSC} | Oscillator Clock | V _{DD} = 3.3 V, Note 2 | | 10 | | MHz |
| F _{VCO} | VCO Output Clock Frequency | | 100 | | 500 | MHz |
| T _{LOCK} | Lock Time | Note 3 | | | 500 | us |
| CLK _{MPLL} | MPLL Output Clock (MCLK) | V _{DD} = 3.3 V | | | 133 | MHz |
| CLK _{CPLL} | CPLL Output Clock (CCLK) | V _{DD} = 3.3 V | | | 100 | MHz |
| CLK _{PPLL} | PPLL Output Clock (PCLK) | V _{DD} = 3.3 V | | | 80 | MHz |

Table A-12: Electrical Characteristics (Continued)

| Symbol | Parameter | Condition | Min. | Typ. | Max. | Unit |
|--|---------------------------------------|-----------|------|------|------|------|
| Serial Host Interface | | | | | | |
| CLK _{SPI} | SPI Input Clock | | | | 50 | MHz |
| Input/Output (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: Measured on tester with 8 bit MPU interface and without extra load.

Note 2: Parasitic effect used in the Crystal Oscillator.



Typical: R1 = 50Ω (25-100Ω), L1 = 3.4mH, C1 = 13fF, C0 = 2.8pF

Figure A-4: Equivalent Circuit

Note 3: Time from power-up or change PLLs' parameters until to PLL have stable clock output.

➤ Function Description

1. Clock and Reset

1.1 Clock

LT7381 embedded three PLL circuit to generate three clock source for internal circuit operation:

- 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:

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

In above formula, XI is the external Oscillator / Clock input. The input frequency "XIN/R" is no less than 1MHz, and the default value is 1MHz. "R" is Input Divider Ratio, it 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 which between 2 ~ 511.

Table 1-1: 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 1-2: PLL Register Setting (2)

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

For example, 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 rule of three clock are::

1. **CCLK * 2 >= MCLK >= CCLK**
2. **CCLK >= PCLK * 1.5**

Usually TFT manufacturers will be based on their TFT characteristics to inform the best display of Pixel Clock (PCLK). Therefore, user can setup the register according to the requirements of the PCLK. And according to the above rule to setup CCLK and MCLK.

According to the different resolution of LCD panel, the PLL output should generate different clock frequency. For example, if LCD panel resolution is 640*480, the recommend values are: PCLK = 20MHz, MCLK = 40MHz, CCLK = 40MHz, then the register are setting as following:

| | |
|--|---|
| PCLK = XI * (N / R) ÷ OD = 10MHz * (80 / 10) ÷ 4 = 20MHz | REG[05h] : OD = 11b, R = 01010b REG[06h] : N = 01010000b |
| MCLK = XI * (N / R) ÷ OD = 10MHz * (160 / 10) ÷ 4 = 40MHz | REG[07h] : OD = 11b, R = 01010b REG[08h] : N = 10100000b |
| CCLK = XI * (N / R) ÷ OD = 10MHz * (160 / 10) ÷ 4 = 40MHz | REG[09h] : OD = 11b, R = 01010b REG[0Ah] : N = 10100000b |

Another example, if resolution is 800*480, the recommend values are: PCLK = 25MHz, MCLK = 50MHz, CCLK = 50MHz. We can setup the registers' values as following:

| | |
|--|---|
| PCLK = XI * (N / R) ÷ OD = 10MHz * (100 / 10) ÷ 4 = 25MHz | REG[05h] : OD = 11b, R = 01010b REG[06h] : N = 01100100b |
| MCLK = XI * (N / R) ÷ OD = 10MHz * (200 / 10) ÷ 4 = 50MHz | REG[07h] : OD = 11b, R = 01010b REG[08h] : N = 11001000b |
| CCLK = XI * (N / R) ÷ OD = 10MHz * (200 / 10) ÷ 4 = 50MHz | REG[09h] : OD = 11b, R = 01010b REG[0Ah] : N = 11001000b |

Table 1-3: PLL Register Setting Example

| Registers | 640*480 | 800*480 |
|------------------|----------------|----------------|
| REG[05h], PPLL1 | 11010100b | 11010100b |
| REG[06h], PPLL2 | 01010000b | 01100100b |
| REG[07h], MPLL1 | 11010100b | 11010100b |
| REG[08h], MPLL2 | 10100000b | 11001000b |
| REG[09h], CPLL1 | 11010100b | 11010100b |
| REG[0Ah], CPLL2 | 10100000b | 11001000b |

1.2 Reset

1.2.1 Power-on Reset

LT7381 embedded a Power-On-Reset for core system. It is an active low signal and may output to external circuits by RST# pin to synchronize whole system. When system power (3.3V) on, internal reset will active until internal power stable and then de-active after 256 OSC(X'tal Oscillator) clocks.

1.2.2 External Reset

LT7381 has capability to receive external reset(RST#) event to synchronize with external system. The external reset event will be admitted when RST# keep low and stable at least 256 OSC clocks.

Before the start to access LT7381, Host should check it's Status Register(STSR) bit [1], i.e. operation mode status bit, and make sure it's in "Normal operation state".

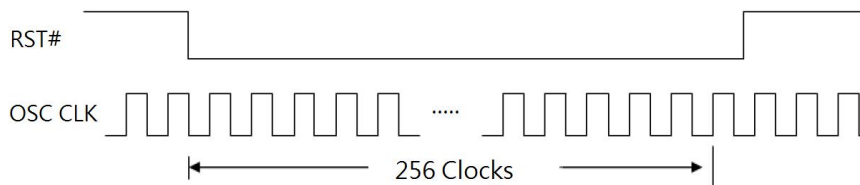


Figure 1-1: External Reset Signal

1.2.3 Software Reset

If the Host write registers REG[00h] bit0 to 1, the LT7381 will be reset by software. The software reset will only reset the internal state machine of LT7381, and the other registers values will not be affected or cleared. After the software reset is complete, the REG[00h] bit0 will automatically be cleared to 0.

2. Host Interface

LT7381 is control by external Host and through the host interface to access LT7381's Registers and Display Memory. LT7381 provide 8bits or 16bits parallel mode, serial SPI mode, and I2C mode for Host's communication. These interface mode is setup by PSM[2:0] pins:

Table 2-1: 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 the different MCU interfaces cannot be used at the same time, so LT7381 provides a shared pin mode. When use Serial Mode, the other parallel pins can also be set to GPIO use. Please refer to the following table:

Table 2-2: 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] | -- | DB[15:0] | -- | DB[15:0] | GPIOA[0:7] | GPIOA[0:7] | GPIOA[0:7] |
| DB[7] | DB[7:0] | | DB[7:0] | | 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 data transfer is determined by the bit0 of Register REG[01h]. When this bit0=0, then 8bits data transfer was selected. And if this bit0=1 then 16bits selected.

The LT7381 supports different host interfaces, and was selected by pins PSM[2:0].

Table 2-3: Host Interface Supporting List of LT7381

| No. | Host Interface Mode | LT7381 |
|-----|-------------------------------------|--------|
| 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 |
| 6 | 4-Wire SPI Mode | v |
| 7 | I2C Mode | v |

2.1 Parallel Host Interface

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

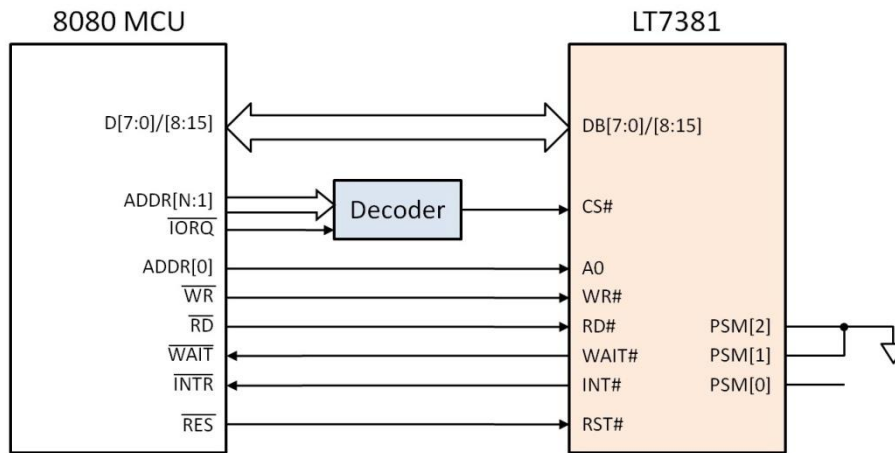


Figure 2-1: 8080 Parallel Mode Interface

The WAIT# signal is used to indicate LT7381 is ready to transfer data or not. If WAIT# signal did not connect, then the Host access cycle time has to length than five CCLK clocks to avoid access fail. If the Host's Reset signal is active low, then it can connect to RST# of LT7381. Of course, the RST# can also control by the I/O pin of host, or connect a RC circuit to generate a low pulse. However, either way to confirm RST#'s active cycle has to keep at least 256 system clock cycle. While using LT7381, Host should first confirm the state register bit1, to know whether the LT7381 in the standard operating state.

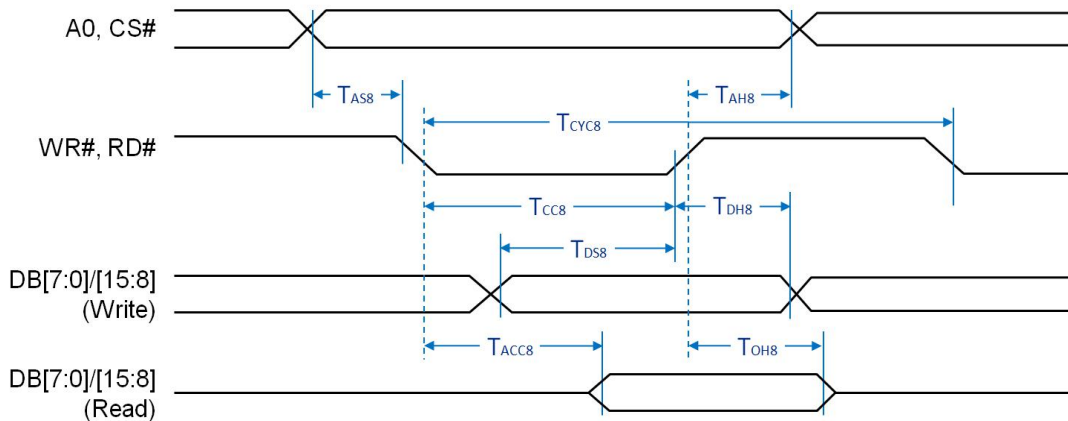


Figure 2-2: 8080 Parallel Mode Interface Timing

Table 2-4: 8080 Parallel Mode Interface Timing Parameter

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

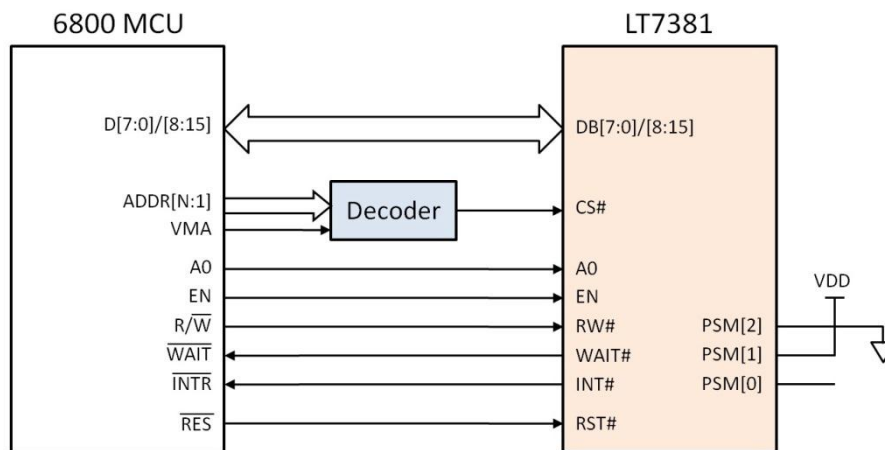


Figure 2-3: 6800 Parallel Mode Interface

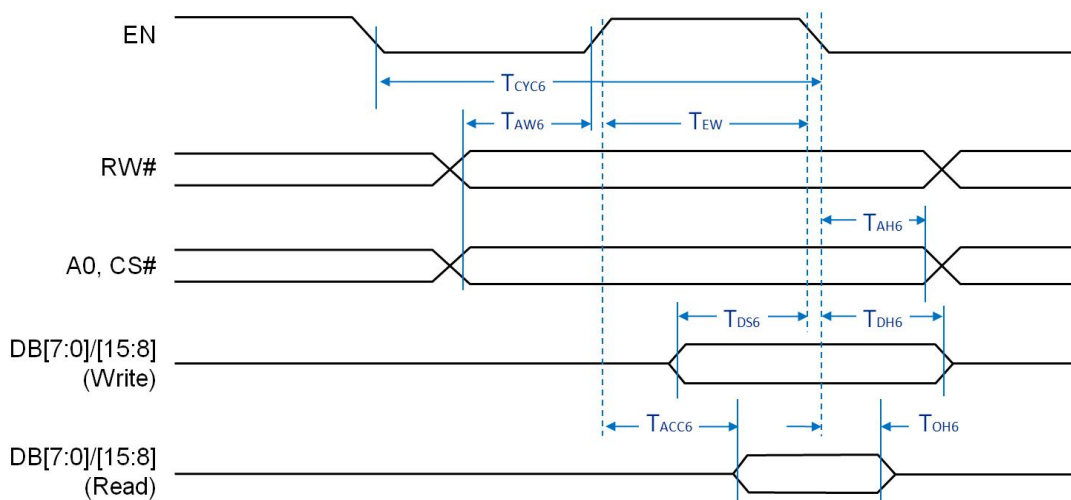


Figure 2-4: 6800 Parallel Mode Interface Timing

Table 2-5: 6800 Parallel Mode Interface Timing Parameter

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

Host through the host interface to access LT7381's Registers and Display Memory. LT7381 has one Status Register and 256 Instruction Registers (i.e. .REG[00h] ~ REG[FF]). The access procedure are 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 through interface and write data into Display RAM. The procedure of access Display RAM is as following:

Display RAM Write:

1. Set the Active Window Registers before writing any image data.
2. Perform an register write to Graphic R/W Position Register 0, REG[5Fh]).
3. Repeat step 2 until setup all the Active Window & Graphic R/W Position Coordinates.
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.

2.2 Serial Host Interface

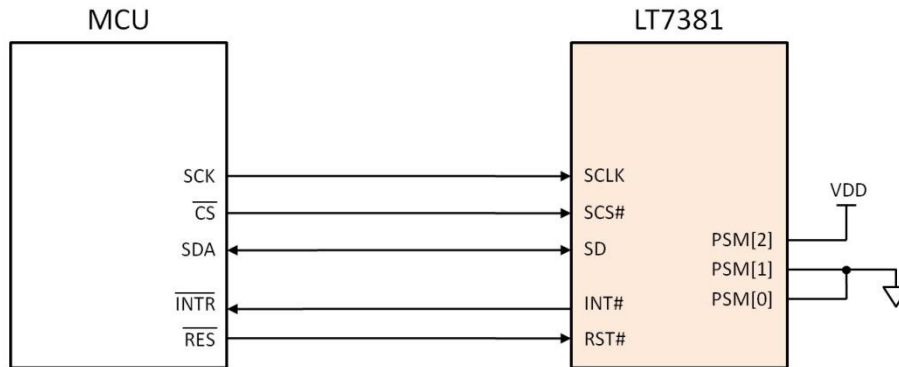


Figure 2-5: 3-Wire SPI Interface

The above circuit is the LT7381'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:

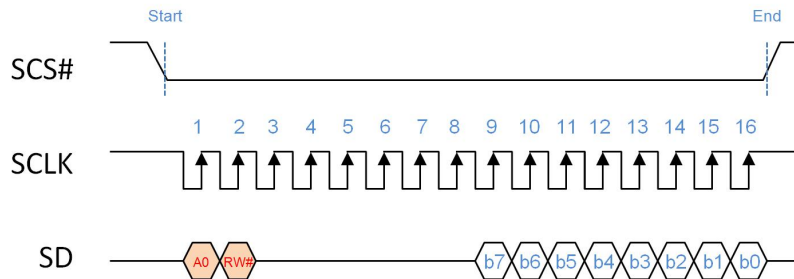


Figure 2-6: 3-Wire SPI Interface Timing

Status Register Read:

1. Host drive SCS#(Low) and SCLK(SPI Clock).
2. Host drive A0(Low), then drive RW#(High).
3. LT7381 will drive the Data of Status Register (b7 ~ b0) at 9th ~ 16th 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 9th ~ 16th Clock to LT7381.

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 9th ~ 16th Clock to LT7381. i.e. Data will be stored in Register or Memory.

Read Register's Data:

1. Host drive SCS#(Low) and SCLK.
2. Host drive A0(High), then drive RW#(High).
3. LT7381 will drive the Data of Register at 9th ~ 16th Clock. Then Host will get the content of Register.

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

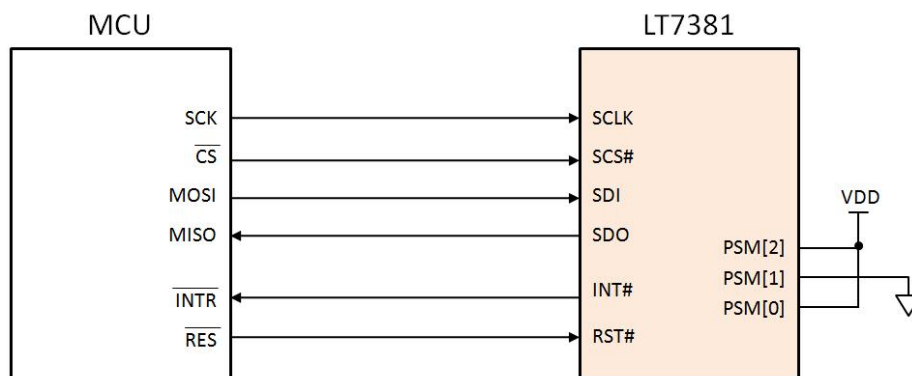


Figure 2-7: 4-Wire SPI Interface

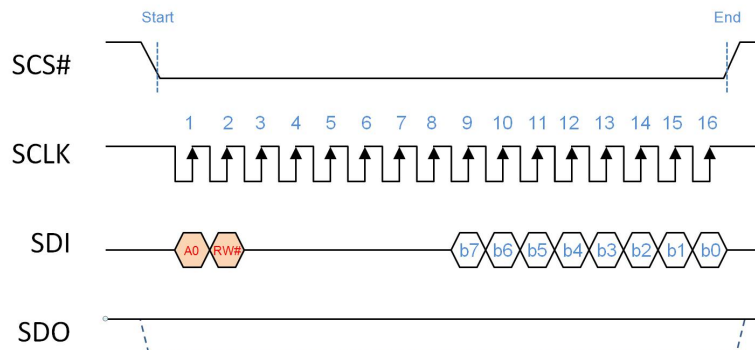


Figure 2-8: 4-Wire SPI Interface Write Timing

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

The following Timing diagram is the Read Cycle of 4-Wires SPI. When Host drive A0(Low) and RW#(High), that's means Host want to read the data of Status Register. LT7381 will drive the Data of Status Register (b7 ~ b0) at 9th ~ 16th Clock. Then Host will get the data of Status Register. When Host drive A0(High), then RW#(High) that's means Host want to read the data of Command Register. LT7381 will drive the Data of Command Register (b7 ~ b0) at 9th ~ 16th Clock for Host. Of course, Host will get the content of Command Register.

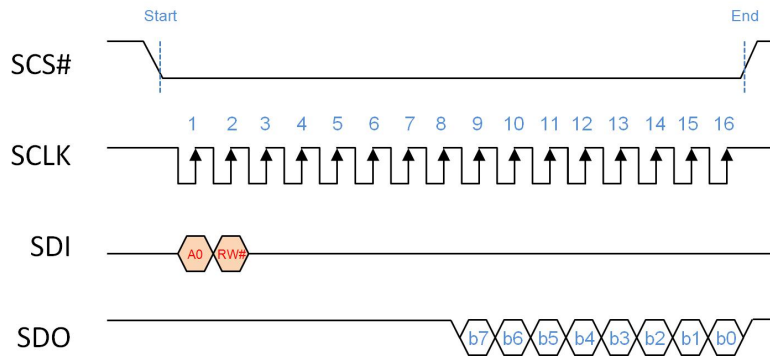


Figure 2-9: 4-Wire SPI Interface Read Timing

The serial I2C interface is also almost same as 3-Wires SPI interface. But I2C interface only need 2 wires for data transfer. The following is the application circuit of I2C interface. Signals I2CA[5:0] are used to setup LT7381's Device ID, and to avoid confuse with other's I2C device. In this example circuit, I2CA[5:3] connect to VDD, and if all DIP Switch are "ON" state, then I2C Device ID is 111000b. i.e. 38h. Therefore if Host drive I2C timing with "111000b" Device ID, then LT7381 will communicate in the I2C cycle time with Host.

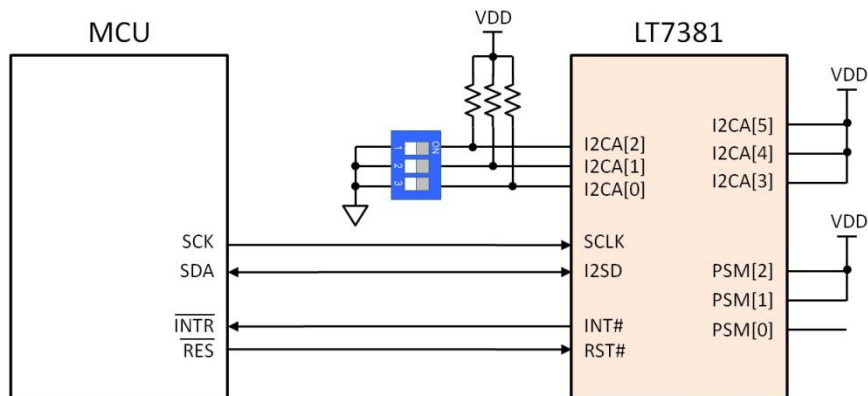


Figure 2-10: I2C Interface

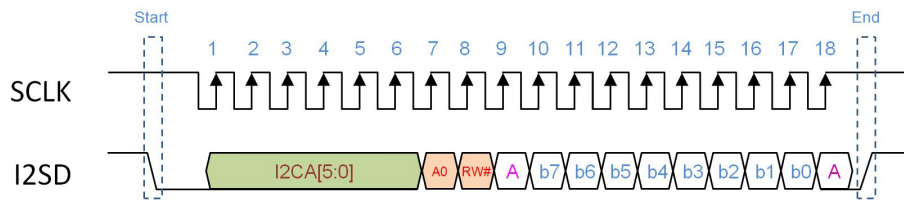


Figure 2-11: I2C Interface Timing

Figure 2-11 is the timing diagram of I2C interface. At first, Host has to know 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 2-10, the Device ID data is 111000. The definition of A0 and RW# are same as SPI interface. When Host release A0(High) and RW#(Low), that means Host will write data(b7 ~ b0, at 10th ~ 17th Clock) to Command Register or Display RAM. If Host release A0(High) and RW#(High), that means Host want to read date from Command Register. LT7381 will release the content(b7 ~b0) of Command Register on 10th ~ 17th Clock. If Host release A0(Low) and RW#(High), that means Host want to read Status Data. LT7381 will release the Status Data(B7 ~ B0) on 10th ~ 17th Clock. Host will get the data of LT7381's Status Register.

2.3 Display Input Data Format

LT7381 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 will be based on 8bits MCU, 16bits MCU and display color (RGB) in different data format.

2.3.1 Input Data without Opacity (RGB)

Table 2-6: 8bits MCU, 1bpp Monochrom Mode

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

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

| Order | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| 1 | R ₀ ⁷ | R ₀ ⁶ | R ₀ ⁵ | G ₀ ⁷ | G ₀ ⁶ | G ₀ ⁵ | B ₀ ⁷ | B ₀ ⁶ |
| 2 | R ₁ ⁷ | R ₁ ⁶ | R ₁ ⁵ | G ₁ ⁷ | G ₁ ⁶ | G ₁ ⁵ | B ₁ ⁷ | B ₁ ⁶ |
| 3 | R ₂ ⁷ | R ₂ ⁶ | R ₂ ⁵ | G ₂ ⁷ | G ₂ ⁶ | G ₂ ⁵ | B ₂ ⁷ | B ₂ ⁶ |
| 4 | R ₃ ⁷ | R ₃ ⁶ | R ₃ ⁵ | G ₃ ⁷ | G ₃ ⁶ | G ₃ ⁵ | B ₃ ⁷ | B ₃ ⁶ |
| 5 | R ₄ ⁷ | R ₄ ⁶ | R ₄ ⁵ | G ₄ ⁷ | G ₄ ⁶ | G ₄ ⁵ | B ₄ ⁷ | B ₄ ⁶ |
| 6 | R ₅ ⁷ | R ₅ ⁶ | R ₅ ⁵ | G ₅ ⁷ | G ₅ ⁶ | G ₅ ⁵ | B ₅ ⁷ | B ₅ ⁶ |

Table 2-8: 8bits MCU, 16bpp Mode (RGB 5:6:5)

| Order | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| 1 | G ₀ ⁴ | G ₀ ³ | G ₀ ² | B ₀ ⁷ | B ₀ ⁶ | B ₀ ⁵ | B ₀ ⁴ | B ₀ ³ |
| 2 | R ₀ ⁷ | R ₀ ⁶ | R ₀ ⁵ | R ₀ ⁴ | R ₀ ³ | G ₀ ⁷ | G ₀ ⁶ | G ₀ ⁵ |
| 3 | G ₁ ⁴ | G ₁ ³ | G ₁ ² | B ₁ ⁷ | B ₁ ⁶ | B ₁ ⁵ | B ₁ ⁴ | B ₁ ³ |
| 4 | R ₁ ⁷ | R ₁ ⁶ | R ₁ ⁵ | R ₁ ⁴ | R ₁ ³ | G ₁ ⁷ | G ₁ ⁶ | G ₁ ⁵ |
| 5 | G ₂ ⁴ | G ₂ ³ | G ₂ ² | B ₂ ⁷ | B ₂ ⁶ | B ₂ ⁵ | B ₂ ⁴ | B ₂ ³ |
| 6 | R ₂ ⁷ | R ₂ ⁶ | R ₂ ⁵ | R ₂ ⁴ | R ₂ ³ | G ₂ ⁷ | G ₂ ⁶ | G ₂ ⁵ |

Table 2-9: 8bits MCU, 24bpp Mode (RGB 8:8:8)

| Order | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| 1 | B ₀ ⁷ | B ₀ ⁶ | B ₀ ⁵ | B ₀ ⁴ | B ₀ ³ | B ₀ ² | B ₀ ¹ | B ₀ ⁰ |
| 2 | G ₀ ⁷ | G ₀ ⁶ | G ₀ ⁵ | G ₀ ⁴ | G ₀ ³ | G ₀ ² | G ₀ ¹ | G ₀ ⁰ |
| 3 | R ₀ ⁷ | R ₀ ⁶ | R ₀ ⁵ | R ₀ ⁴ | R ₀ ³ | R ₀ ² | R ₀ ¹ | R ₀ ⁰ |
| 4 | B ₁ ⁷ | B ₁ ⁶ | B ₁ ⁵ | B ₁ ⁴ | B ₁ ³ | B ₁ ² | B ₁ ¹ | B ₁ ⁰ |
| 5 | G ₁ ⁷ | G ₁ ⁶ | G ₁ ⁵ | G ₁ ⁴ | G ₁ ³ | G ₁ ² | G ₁ ¹ | G ₁ ⁰ |
| 6 | R ₁ ⁷ | R ₁ ⁶ | R ₁ ⁵ | R ₁ ⁴ | R ₁ ³ | R ₁ ² | R ₁ ¹ | R ₁ ⁰ |

Table 2-10: 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 ₇ | P ₆ | P ₅ | P ₄ | P ₃ | P ₂ | P ₁ | P ₀ |
| 2 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | P ₁₅ | P ₁₄ | P ₁₃ | P ₁₂ | P ₁₁ | P ₁₀ | P ₉ | P ₈ |
| 3 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | P ₂₃ | P ₂₂ | P ₂₁ | P ₂₀ | P ₁₉ | P ₁₈ | P ₁₇ | P ₁₆ |
| 4 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | P ₃₁ | P ₃₀ | P ₂₉ | P ₂₈ | P ₂₇ | P ₂₆ | P ₂₅ | P ₂₄ |
| 5 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | P ₃₉ | P ₃₈ | P ₃₇ | P ₃₆ | P ₃₅ | P ₃₄ | P ₃₃ | P ₃₂ |
| 6 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | P ₄₇ | P ₄₆ | P ₄₅ | P ₄₄ | P ₄₃ | P ₄₂ | P ₄₁ | P ₄₀ |

Table 2-11: 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 ₁₅ | P ₁₄ | P ₁₃ | P ₁₂ | P ₁₁ | P ₁₀ | P ₉ | P ₈ | P ₇ | P ₆ | P ₅ | P ₄ | P ₃ | P ₂ | P ₁ | P ₀ |
| 2 | P ₃₁ | P ₃₀ | P ₂₉ | P ₂₈ | P ₂₇ | P ₂₆ | P ₂₅ | P ₂₄ | P ₂₃ | P ₂₂ | P ₂₁ | P ₂₀ | P ₁₉ | P ₁₈ | P ₁₇ | P ₁₆ |
| 3 | P ₄₇ | P ₄₆ | P ₄₅ | P ₄₄ | P ₄₃ | P ₄₂ | P ₄₁ | P ₄₀ | P ₃₉ | P ₃₈ | P ₃₇ | P ₃₆ | P ₃₅ | P ₃₄ | P ₃₃ | P ₃₂ |
| 4 | P ₆₃ | P ₆₂ | P ₆₁ | P ₆₀ | P ₅₉ | P ₅₈ | P ₅₇ | P ₅₆ | P ₅₅ | P ₅₄ | P ₅₃ | P ₅₂ | P ₅₁ | P ₅₀ | P ₄₉ | P ₄₈ |
| 5 | P ₇₉ | P ₇₈ | P ₇₇ | P ₇₆ | P ₇₅ | P ₇₄ | P ₇₃ | P ₇₂ | P ₇₁ | P ₇₀ | P ₆₉ | P ₆₈ | P ₆₇ | P ₆₆ | P ₆₅ | P ₆₄ |
| 6 | P ₉₅ | P ₉₄ | P ₉₃ | P ₉₂ | P ₉₁ | P ₉₀ | P ₈₉ | P ₈₈ | P ₈₇ | P ₈₆ | P ₈₅ | P ₈₄ | P ₈₃ | P ₈₂ | P ₈₁ | P ₈₀ |

Table 2-12: 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 ₀ ⁷ | R ₀ ⁶ | R ₀ ⁵ | G ₀ ⁷ | G ₀ ⁶ | G ₀ ⁵ | B ₀ ⁷ | B ₀ ⁶ |
| 2 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | R ₁ ⁷ | R ₁ ⁶ | R ₁ ⁵ | G ₁ ⁷ | G ₁ ⁶ | G ₁ ⁵ | B ₁ ⁷ | B ₁ ⁶ |
| 3 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | R ₂ ⁷ | R ₂ ⁶ | R ₂ ⁵ | G ₂ ⁷ | G ₂ ⁶ | G ₂ ⁵ | B ₂ ⁷ | B ₂ ⁶ |
| 4 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | R ₃ ⁷ | R ₃ ⁶ | R ₃ ⁵ | G ₃ ⁷ | G ₃ ⁶ | G ₃ ⁵ | B ₃ ⁷ | B ₃ ⁶ |
| 5 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | R ₄ ⁷ | R ₄ ⁶ | R ₄ ⁵ | G ₄ ⁷ | G ₄ ⁶ | G ₄ ⁵ | B ₄ ⁷ | B ₄ ⁶ |
| 6 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | R ₅ ⁷ | R ₅ ⁶ | R ₅ ⁵ | G ₅ ⁷ | G ₅ ⁶ | G ₅ ⁵ | B ₅ ⁷ | B ₅ ⁶ |

Table 2-13: 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 ₁ ⁷ | R ₁ ⁶ | R ₁ ⁵ | G ₁ ⁷ | G ₁ ⁶ | G ₁ ⁵ | B ₁ ⁷ | B ₁ ⁶ | R ₀ ⁷ | R ₀ ⁶ | R ₀ ⁵ | G ₀ ⁷ | G ₀ ⁶ | G ₀ ⁵ | B ₀ ⁷ | B ₀ ⁶ |
| 2 | R ₃ ⁷ | R ₃ ⁶ | R ₃ ⁵ | G ₃ ⁷ | G ₃ ⁶ | G ₃ ⁵ | B ₃ ⁷ | B ₃ ⁶ | R ₂ ⁷ | R ₂ ⁶ | R ₂ ⁵ | G ₂ ⁷ | G ₂ ⁶ | G ₂ ⁵ | B ₂ ⁷ | B ₂ ⁶ |
| 3 | R ₅ ⁷ | R ₅ ⁶ | R ₅ ⁵ | G ₅ ⁷ | G ₅ ⁶ | G ₅ ⁵ | B ₅ ⁷ | B ₅ ⁶ | R ₄ ⁷ | R ₄ ⁶ | R ₄ ⁵ | G ₄ ⁷ | G ₄ ⁶ | G ₄ ⁵ | B ₄ ⁷ | B ₄ ⁶ |
| 4 | R ₇ ⁷ | R ₇ ⁶ | R ₇ ⁵ | G ₇ ⁷ | G ₇ ⁶ | G ₇ ⁵ | B ₇ ⁷ | B ₇ ⁶ | R ₆ ⁷ | R ₆ ⁶ | R ₆ ⁵ | G ₆ ⁷ | G ₆ ⁶ | G ₆ ⁵ | B ₆ ⁷ | B ₆ ⁶ |
| 5 | R ₉ ⁷ | R ₉ ⁶ | R ₉ ⁵ | G ₉ ⁷ | G ₉ ⁶ | G ₉ ⁵ | B ₉ ⁷ | B ₉ ⁶ | R ₈ ⁷ | R ₈ ⁶ | R ₈ ⁵ | G ₈ ⁷ | G ₈ ⁶ | G ₈ ⁵ | B ₈ ⁷ | B ₈ ⁶ |
| 6 | R ₁₁ ⁷ | R ₁₁ ⁶ | R ₁₁ ⁵ | G ₁₁ ⁷ | G ₁₁ ⁶ | G ₁₁ ⁵ | B ₁₁ ⁷ | B ₁₁ ⁶ | R ₁₀ ⁷ | R ₁₀ ⁶ | R ₁₀ ⁵ | G ₁₀ ⁷ | G ₁₀ ⁶ | G ₁₀ ⁵ | B ₁₀ ⁷ | B ₁₀ ⁶ |

Note: The Mode-1 and Mode 2 is determined by bit[7:6] of Register[02h], please refer to the Cpatet-13 Reregister Description.

Table 2-14: 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 ₀ ⁷ | R ₀ ⁶ | R ₀ ⁵ | R ₀ ⁴ | R ₀ ³ | G ₀ ⁷ | G ₀ ⁶ | G ₀ ⁵ | G ₀ ⁴ | G ₀ ³ | G ₀ ² | B ₀ ⁷ | B ₀ ⁶ | B ₀ ⁵ | B ₀ ⁴ | B ₀ ³ |
| 2 | R ₁ ⁷ | R ₁ ⁶ | R ₁ ⁵ | R ₁ ⁴ | R ₁ ³ | G ₁ ⁷ | G ₁ ⁶ | G ₁ ⁵ | G ₁ ⁴ | G ₁ ³ | G ₁ ² | B ₁ ⁷ | B ₁ ⁶ | B ₁ ⁵ | B ₁ ⁴ | B ₁ ³ |
| 3 | R ₂ ⁷ | R ₂ ⁶ | R ₂ ⁵ | R ₂ ⁴ | R ₂ ³ | G ₂ ⁷ | G ₂ ⁶ | G ₂ ⁵ | G ₂ ⁴ | G ₂ ³ | G ₂ ² | B ₂ ⁷ | B ₂ ⁶ | B ₂ ⁵ | B ₂ ⁴ | B ₂ ³ |
| 4 | R ₃ ⁷ | R ₃ ⁶ | R ₃ ⁵ | R ₃ ⁴ | R ₃ ³ | G ₃ ⁷ | G ₃ ⁶ | G ₃ ⁵ | G ₃ ⁴ | G ₃ ³ | G ₃ ² | B ₃ ⁷ | B ₃ ⁶ | B ₃ ⁵ | B ₃ ⁴ | B ₃ ³ |
| 5 | R ₄ ⁷ | R ₄ ⁶ | R ₄ ⁵ | R ₄ ⁴ | R ₄ ³ | G ₄ ⁷ | G ₄ ⁶ | G ₄ ⁵ | G ₄ ⁴ | G ₄ ³ | G ₄ ² | B ₄ ⁷ | B ₄ ⁶ | B ₄ ⁵ | B ₄ ⁴ | B ₄ ³ |
| 6 | R ₅ ⁷ | R ₅ ⁶ | R ₅ ⁵ | R ₅ ⁴ | R ₅ ³ | G ₅ ⁷ | G ₅ ⁶ | G ₅ ⁵ | G ₅ ⁴ | G ₅ ³ | G ₅ ² | B ₅ ⁷ | B ₅ ⁶ | B ₅ ⁵ | B ₅ ⁴ | B ₅ ³ |

Table 2-15: 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 ₀ ⁷ | G ₀ ⁶ | G ₀ ⁵ | G ₀ ⁴ | G ₀ ³ | G ₀ ² | G ₀ ¹ | G ₀ ⁰ | B ₀ ⁷ | B ₀ ⁶ | B ₀ ⁵ | B ₀ ⁴ | B ₀ ³ | B ₀ ² | B ₀ ¹ | B ₀ ⁰ |
| 2 | B ₁ ⁷ | B ₁ ⁶ | B ₁ ⁵ | B ₁ ⁴ | B ₁ ³ | B ₁ ² | B ₁ ¹ | B ₁ ⁰ | R ₀ ⁷ | R ₀ ⁶ | R ₀ ⁵ | R ₀ ⁴ | R ₀ ³ | R ₀ ² | R ₀ ¹ | R ₀ ⁰ |
| 3 | R ₁ ⁷ | R ₁ ⁶ | R ₁ ⁵ | R ₁ ⁴ | R ₁ ³ | R ₁ ² | R ₁ ¹ | R ₁ ⁰ | G ₁ ⁷ | G ₁ ⁶ | G ₁ ⁵ | G ₁ ⁴ | G ₁ ³ | G ₁ ² | G ₁ ¹ | G ₁ ⁰ |
| 4 | G ₂ ⁷ | G ₂ ⁶ | G ₂ ⁵ | G ₂ ⁴ | G ₂ ³ | G ₂ ² | G ₂ ¹ | G ₂ ⁰ | B ₂ ⁷ | B ₂ ⁶ | B ₂ ⁵ | B ₂ ⁴ | B ₂ ³ | B ₂ ² | B ₂ ¹ | B ₂ ⁰ |
| 5 | B ₃ ⁷ | B ₃ ⁶ | B ₃ ⁵ | B ₃ ⁴ | B ₃ ³ | B ₃ ² | B ₃ ¹ | B ₃ ⁰ | R ₂ ⁷ | R ₂ ⁶ | R ₂ ⁵ | R ₂ ⁴ | R ₂ ³ | R ₂ ² | R ₂ ¹ | R ₂ ⁰ |
| 6 | R ₃ ⁷ | R ₃ ⁶ | R ₃ ⁵ | R ₃ ⁴ | R ₃ ³ | R ₃ ² | R ₃ ¹ | R ₃ ⁰ | G ₃ ⁷ | G ₃ ⁶ | G ₃ ⁵ | G ₃ ⁴ | G ₃ ³ | G ₃ ² | G ₃ ¹ | G ₃ ⁰ |

Table 2-16: 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 ₀ ⁷ | G ₀ ⁶ | G ₀ ⁵ | G ₀ ⁴ | G ₀ ³ | G ₀ ² | G ₀ ¹ | G ₀ ⁰ | B ₀ ⁷ | B ₀ ⁶ | B ₀ ⁵ | B ₀ ⁴ | B ₀ ³ | B ₀ ² | B ₀ ¹ | B ₀ ⁰ |
| 2 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | R ₀ ⁷ | R ₀ ⁶ | R ₀ ⁵ | R ₀ ⁴ | R ₀ ³ | R ₀ ² | R ₀ ¹ | R ₀ ⁰ |
| 3 | G ₁ ⁷ | G ₁ ⁶ | G ₁ ⁵ | G ₁ ⁴ | G ₁ ³ | G ₁ ² | G ₁ ¹ | G ₁ ⁰ | B ₁ ⁷ | B ₁ ⁶ | B ₁ ⁵ | B ₁ ⁴ | B ₁ ³ | B ₁ ² | B ₁ ¹ | B ₁ ⁰ |
| 4 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | R ₁ ⁷ | R ₁ ⁶ | R ₁ ⁵ | R ₁ ⁴ | R ₁ ³ | R ₁ ² | R ₁ ¹ | R ₁ ⁰ |
| 5 | G ₂ ⁷ | G ₂ ⁶ | G ₂ ⁵ | G ₂ ⁴ | G ₂ ³ | G ₂ ² | G ₂ ¹ | G ₂ ⁰ | B ₂ ⁷ | B ₂ ⁶ | B ₂ ⁵ | B ₂ ⁴ | B ₂ ³ | B ₂ ² | B ₂ ¹ | B ₂ ⁰ |
| 6 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | R ₂ ⁷ | R ₂ ⁶ | R ₂ ⁵ | R ₂ ⁴ | R ₂ ³ | R ₂ ² | R ₂ ¹ | R ₂ ⁰ |

2.3.2 Input Data with Opacity (α RGB)

LT7381 provide a palette of 64 simultaneous colors from a total of 4096 different colors with opacity attribute for OSD (On-Screen-Display) application. User may load preferred color into embedded color palette then pick it up by index color. The α value stands for opacity.

Table 2-17: 8bits MCU, 8bpp Mode (α Index 2:6)

| Order | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-------|-----------------|-----------------|------------------------|------|------|------|------|------|
| 1 | α_1^3 | α_1^2 | Index color of pixel 0 | | | | | |
| 2 | α_3^3 | α_3^2 | Index color of pixel 1 | | | | | |
| 3 | α_5^3 | α_5^2 | Index color of pixel 2 | | | | | |
| 4 | α_7^3 | α_7^2 | Index color of pixel 3 | | | | | |
| 5 | α_9^3 | α_9^2 | Index color of pixel 4 | | | | | |
| 6 | α_{11}^3 | α_{11}^2 | Index color of pixel 5 | | | | | |

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

Table 2-18: 8bits MCU, 16bpp Mode (α 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 | α_0^3 | α_0^2 | α_0^1 | α_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 | α_1^3 | α_1^2 | α_1^1 | α_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 | α_2^3 | α_2^2 | α_2^1 | α_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→100%, 1→30/32, 2→28/32, 3→26/32,
4→24/32,, 12→8/32, 13→6/32, 14→4/32, 15→0.

Table 2-19: 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 | α_0^3 | α_0^2 | Index color of pixel 0 | | | | | |
| 2 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | α_1^3 | α_1^2 | Index color of pixel 1 | | | | | |
| 3 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | α_2^3 | α_2^2 | Index color of pixel 2 | | | | | |
| 4 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | α_3^3 | α_3^2 | Index color of pixel 3 | | | | | |
| 5 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | α_4^3 | α_4^2 | Index color of pixel 4 | | | | | |
| 6 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | α_5^3 | α_5^2 | Index color of pixel 5 | | | | | |

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

Table 2-20: 16bits MCU, 12bpp Mode (αRGB 4:4:4)

| Order | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-------|--------------|--------------|--------------|--------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 1 | α_0^3 | α_0^2 | α_0^1 | α_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 |
| 2 | α_1^3 | α_1^2 | α_1^1 | α_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 |
| 3 | α_2^3 | α_2^2 | α_2^1 | α_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 |
| 4 | α_3^3 | α_3^2 | α_3^1 | α_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 |
| 5 | α_4^3 | α_4^2 | α_4^1 | α_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 |
| 6 | α_5^3 | α_5^2 | α_5^1 | α_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→0, 1→2/32, 2→4/32, 3→6/32, 4→8/32,, 12→24/32, 13→26/32, 14→28/32, 15→100%.

3. Display Memory

LT7381 embedded a 32Mb Display RAM. The Host(MCU) through the instructions to save the displayed data to internal Display RAM. And LT7381's internal display engine will continue to read the data of memory, then sent to TFT driver. The capacity of the Display RAM is also related to the resolutions and image layers supported. As shown in the following table:

Table 3-1: Embedded Display RAM Capacity

| Display RAM Capacity | Resolution (Max.) | Color (Max.) | Image Layer (@Max Color) |
|----------------------|-------------------|--------------|--------------------------|
| 32Mb | 320*240 | 16.7M | 18 |
| | 480*272 | 16.7M | 10 |
| | 640*480 | 16.7M | 4 |
| | 800*600 | 16.7M | 2 |
| | 1024*768 | 16.7M | 1 |

The Display RAM type of LT7381 embedded is a kind of High-Speed SDRAM (Synchronous Dynamic Random Access Memory). Before the Host access the Display RAM, it must be based on the type of Display RAM to set the relevant register initialization as following steps:

- According to the Display RAM capacity, setup Register REG[E0h]. The value of REG[E0h] must be set according to the LT7381 to avoid display anomalies and image confusion. Please refer to Table 14-5 of Chapter 14.
- According to the type of Display RAM, setup Register REG[E1H], REG[E2h], REG[E3h], including setup the CAS latency, refresh interval, etc..... The typical Display RAM refresh interval is 64ms. The value of these registers are recommended according to the LT7381 used. Please refer to Table 14-6 of Chapter 14.
- Set Register REG[E4h] Bit0 is 1, start Display RAM initialization processing.
- Read Register REG[E4h] bit0, if it becomes 1, that means initialization is complete.

3.1 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 writes the image data must according to these formats, the following tables are 8/16/24bpp RGB data arrangement format:

3.1.1 8bpp Display Data (RGB 3:3:2)

Table 3-2: 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 ₁ ⁷ | R ₁ ⁶ | R ₁ ⁵ | G ₁ ⁷ | G ₁ ⁶ | G ₁ ⁵ | B ₁ ⁷ | B ₁ ⁶ | R ₀ ⁷ | R ₀ ⁶ | R ₀ ⁵ | G ₀ ⁷ | G ₀ ⁶ | G ₀ ⁵ | B ₀ ⁷ | B ₀ ⁶ |
| 0002h | R ₃ ⁷ | R ₃ ⁶ | R ₃ ⁵ | G ₃ ⁷ | G ₃ ⁶ | G ₃ ⁵ | B ₃ ⁷ | B ₃ ⁶ | R ₂ ⁷ | R ₂ ⁶ | R ₂ ⁵ | G ₂ ⁷ | G ₂ ⁶ | G ₂ ⁵ | B ₂ ⁷ | B ₂ ⁶ |
| 0004h | R ₅ ⁷ | R ₅ ⁶ | R ₅ ⁵ | G ₅ ⁷ | G ₅ ⁶ | G ₅ ⁵ | B ₅ ⁷ | B ₅ ⁶ | R ₄ ⁷ | R ₄ ⁶ | R ₄ ⁵ | G ₄ ⁷ | G ₄ ⁶ | G ₄ ⁵ | B ₄ ⁷ | B ₄ ⁶ |
| 0006h | R ₇ ⁷ | R ₇ ⁶ | R ₇ ⁵ | G ₇ ⁷ | G ₇ ⁶ | G ₇ ⁵ | B ₇ ⁷ | B ₇ ⁶ | R ₆ ⁷ | R ₆ ⁶ | R ₆ ⁵ | G ₆ ⁷ | G ₆ ⁶ | G ₆ ⁵ | B ₆ ⁷ | B ₆ ⁶ |
| 0008h | R ₉ ⁷ | R ₉ ⁶ | R ₉ ⁵ | G ₉ ⁷ | G ₉ ⁶ | G ₉ ⁵ | B ₉ ⁷ | B ₉ ⁶ | R ₈ ⁷ | R ₈ ⁶ | R ₈ ⁵ | G ₈ ⁷ | G ₈ ⁶ | G ₈ ⁵ | B ₈ ⁷ | B ₈ ⁶ |
| 000Ah | R ₁₁ ⁷ | R ₁₁ ⁶ | R ₁₁ ⁵ | G ₁₁ ⁷ | G ₁₁ ⁶ | G ₁₁ ⁵ | B ₁₁ ⁷ | B ₁₁ ⁶ | R ₁₀ ⁷ | R ₁₀ ⁶ | R ₁₀ ⁵ | G ₁₀ ⁷ | G ₁₀ ⁶ | G ₁₀ ⁵ | B ₁₀ ⁷ | B ₁₀ ⁶ |

3.1.2 16bpp Display Data (RGB 5:6:5)

Table 3-3: 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 ₀ ⁷ | R ₀ ⁶ | R ₀ ⁵ | R ₀ ⁴ | R ₀ ³ | G ₀ ⁷ | G ₀ ⁶ | G ₀ ⁵ | G ₀ ⁴ | G ₀ ³ | G ₀ ² | B ₀ ⁷ | B ₀ ⁶ | B ₀ ⁵ | B ₀ ⁴ | B ₀ ³ |
| 0002h | R ₁ ⁷ | R ₁ ⁶ | R ₁ ⁵ | R ₁ ⁴ | R ₁ ³ | G ₁ ⁷ | G ₁ ⁶ | G ₁ ⁵ | G ₁ ⁴ | G ₁ ³ | G ₁ ² | B ₁ ⁷ | B ₁ ⁶ | B ₁ ⁵ | B ₁ ⁴ | B ₁ ³ |
| 0004h | R ₂ ⁷ | R ₂ ⁶ | R ₂ ⁵ | R ₂ ⁴ | R ₂ ³ | G ₂ ⁷ | G ₂ ⁶ | G ₂ ⁵ | G ₂ ⁴ | G ₂ ³ | G ₂ ² | B ₂ ⁷ | B ₂ ⁶ | B ₂ ⁵ | B ₂ ⁴ | B ₂ ³ |
| 0006h | R ₃ ⁷ | R ₃ ⁶ | R ₃ ⁵ | R ₃ ⁴ | R ₃ ³ | G ₃ ⁷ | G ₃ ⁶ | G ₃ ⁵ | G ₃ ⁴ | G ₃ ³ | G ₃ ² | B ₃ ⁷ | B ₃ ⁶ | B ₃ ⁵ | B ₃ ⁴ | B ₃ ³ |
| 0008h | R ₄ ⁷ | R ₄ ⁶ | R ₄ ⁵ | R ₄ ⁴ | R ₄ ³ | G ₄ ⁷ | G ₄ ⁶ | G ₄ ⁵ | G ₄ ⁴ | G ₄ ³ | G ₄ ² | B ₄ ⁷ | B ₄ ⁶ | B ₄ ⁵ | B ₄ ⁴ | B ₄ ³ |
| 000Ah | R ₅ ⁷ | R ₅ ⁶ | R ₅ ⁵ | R ₅ ⁴ | R ₅ ³ | G ₅ ⁷ | G ₅ ⁶ | G ₅ ⁵ | G ₅ ⁴ | G ₅ ³ | G ₅ ² | B ₅ ⁷ | B ₅ ⁶ | B ₅ ⁵ | B ₅ ⁴ | B ₅ ³ |

3.1.3 24bpp Display Data (RGB 8:8:8)

Table 3-4: 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 ₀ ⁷ | G ₀ ⁶ | G ₀ ⁵ | G ₀ ⁴ | G ₀ ³ | G ₀ ² | G ₀ ¹ | G ₀ ⁰ | B ₀ ⁷ | B ₀ ⁶ | B ₀ ⁵ | B ₀ ⁴ | B ₀ ³ | B ₀ ² | B ₀ ¹ | B ₀ ⁰ |
| 0002h | B ₁ ⁷ | B ₁ ⁶ | B ₁ ⁵ | B ₁ ⁴ | B ₁ ³ | B ₁ ² | B ₁ ¹ | B ₁ ⁰ | R ₀ ⁷ | R ₀ ⁶ | R ₀ ⁵ | R ₀ ⁴ | R ₀ ³ | R ₀ ² | R ₀ ¹ | R ₀ ⁰ |
| 0004h | R ₁ ⁷ | R ₁ ⁶ | R ₁ ⁵ | R ₁ ⁴ | R ₁ ³ | R ₁ ² | R ₁ ¹ | R ₁ ⁰ | G ₁ ⁷ | G ₁ ⁶ | G ₁ ⁵ | G ₁ ⁴ | G ₁ ³ | G ₁ ² | G ₁ ¹ | G ₁ ⁰ |
| 0006h | G ₂ ⁷ | G ₂ ⁶ | G ₂ ⁵ | G ₂ ⁴ | G ₂ ³ | G ₂ ² | G ₂ ¹ | G ₂ ⁰ | B ₂ ⁷ | B ₂ ⁶ | B ₂ ⁵ | B ₂ ⁴ | B ₂ ³ | B ₂ ² | B ₂ ¹ | B ₂ ⁰ |
| 0008h | B ₃ ⁷ | B ₃ ⁶ | B ₃ ⁵ | B ₃ ⁴ | B ₃ ³ | B ₃ ² | B ₃ ¹ | B ₃ ⁰ | R ₂ ⁷ | R ₂ ⁶ | R ₂ ⁵ | R ₂ ⁴ | R ₂ ³ | R ₂ ² | R ₂ ¹ | R ₂ ⁰ |
| 000Ah | R ₃ ⁷ | R ₃ ⁶ | R ₃ ⁵ | R ₃ ⁴ | R ₃ ³ | R ₃ ² | R ₃ ¹ | R ₃ ⁰ | G ₃ ⁷ | G ₃ ⁶ | G ₃ ⁵ | G ₃ ⁴ | G ₃ ³ | G ₃ ² | G ₃ ¹ | G ₃ ⁰ |

3.1.4 Index Display with Opacity (α RGB 2:2:2:2)

Table 3-5: Index Display with Opacity (α RGB 2:2:2:2)

| Addr | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-------|-----------------|-----------------|-------------------------|-------|-------|-------|------|------|-----------------|-----------------|-------------------------|------|------|------|------|------|
| 0000h | α_1^3 | α_1^2 | Index color of pixel 1 | | | | | | α_0^3 | α_0^2 | Index color of pixel 0 | | | | | |
| 0002h | α_3^3 | α_3^2 | Index color of pixel 3 | | | | | | α_2^3 | α_2^2 | Index color of pixel 2 | | | | | |
| 0004h | α_5^3 | α_5^2 | Index color of pixel 5 | | | | | | α_4^3 | α_4^2 | Index color of pixel 4 | | | | | |
| 0006h | α_7^3 | α_7^2 | Index color of pixel 7 | | | | | | α_6^3 | α_6^2 | Index color of pixel 6 | | | | | |
| 0008h | α_9^3 | α_9^2 | Index color of pixel 9 | | | | | | α_8^3 | α_8^2 | Index color of pixel 8 | | | | | |
| 000Ah | α_{11}^3 | α_{11}^2 | Index color of pixel 11 | | | | | | α_{10}^3 | α_{10}^2 | Index color of pixel 10 | | | | | |

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

3.1.5 12bpp Display with Opacity (RGB 4:4:4:4)

Table 3-6: 12bpp Display with Opacity (RGB 4:4:4:4)

| Addr | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-------|--------------|--------------|--------------|--------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 0000h | α_0^3 | α_0^2 | α_0^1 | α_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 | α_1^3 | α_1^2 | α_1^1 | α_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 | α_2^3 | α_2^2 | α_2^1 | α_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 | α_3^3 | α_3^2 | α_3^1 | α_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 | α_4^3 | α_4^2 | α_4^1 | α_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 | α_5^3 | α_5^2 | α_5^1 | α_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→0, 1→2/32, 2→4/32, 3→6/32, 4→8/32,, 12→24/32, 13→26/32, 14→28/32, 15→100%.

3.2 Color Palette RAM

Table 3-7: 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 |

4. LCD Interface

LT7381 Support 16, 18, 24bits RGB interface of TFT Panel, whether 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 LT7381 LCD Display data corresponds to the RGB data as shown in the below table. The Host can setup REG[01h] Bit[4:3] to select 16bits, 18bits or 24bits resolution.

Table 4-1: 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 4-2: RGB Data Signals of LT7381

| LCD Data Bus | RGB Signal Number | Colors |
|--------------|-------------------|-------------|
| PD[23~0] | R:G:B = 8:8:8 | 16.7M Color |

Figure 4-1 is the timing diagram of LT7381 TFT-LCD output signals. In addition to the RGB data lines of above mentioned, LT7381 also provides PCLK (Panel Scan Clock), VSYNC Pulse, HSYNC Pulse and Data Enable signal. The frequency of PCLK is setup by REG[05h] and REG[06h]. Please refer to the description in Section 1.1 and Chapter 14th.

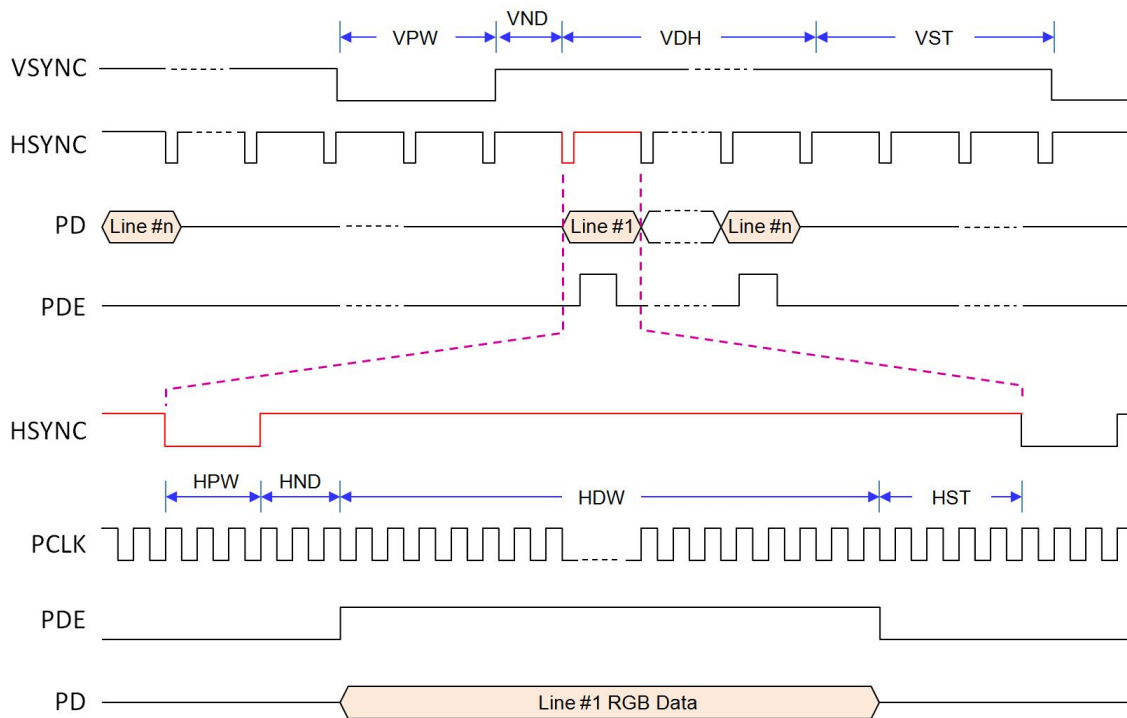


Figure 4-1: TFT-LCD Interface Timing

5. Display Function

5.1 Color Bar

The LT7381 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 Host to set REG[12h] bit5 to 1.

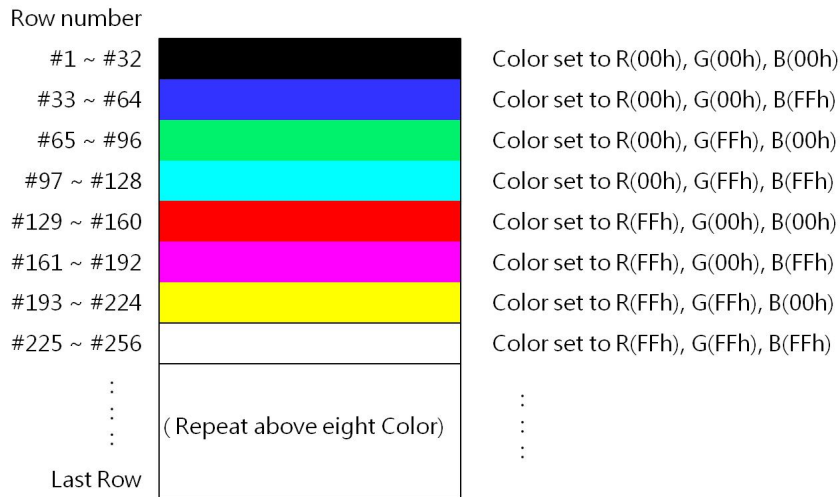


Figure 5-1: Color Bar

5.2 Main Window

The LCD main window size can be defined by setting Registers REG[14h] to REG[1Fh]. At first, Host can save different images in memory buffer. Then setup the related Registers (REG[20h] ~ REG[29h]) to select a different buffer for show different images.

5.2.1 Configure Display Image Buffer

Display RAM is used to store the displayed image. And how many images can be stored is determined by the image resolution and color. For example, the image resolution is 640*480, with 65K color (16bits), then 13 image data can be stored in 64Mbits Display RAM:

$$\text{Number of Image} = (64 \times 1024 \times 1024) / (640 \times 480 \times 16) = 13.6$$

If the image resolution is 800*600, with 256 color (8bits), then 34 image data can be stored in 64Mbits Display RAM:

$$\text{Number of Image} = (128 \times 1024 \times 1024) / (800 \times 600 \times 8) = 34.9$$

LT7381 must set the Starting Position of the Canvas, Width and the Working window range before writing the image data to Display RAM. And also set the Working window range. Please refer the registers REG[50H] to REG[5Eh] for detail description.

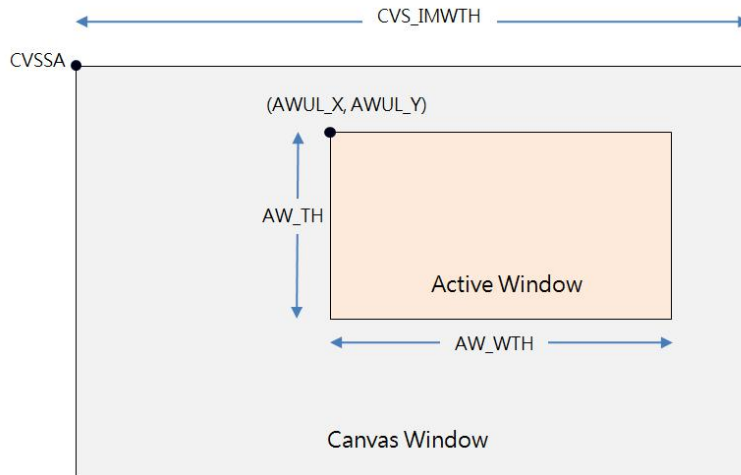


Figure 5-1: Canvas Window and Active Window

5.2.2 Write Image Data to Display Image Buffer

The following diagram is a flowchart that writes image data to Display RAM and displays the main window image on an LCD screen:

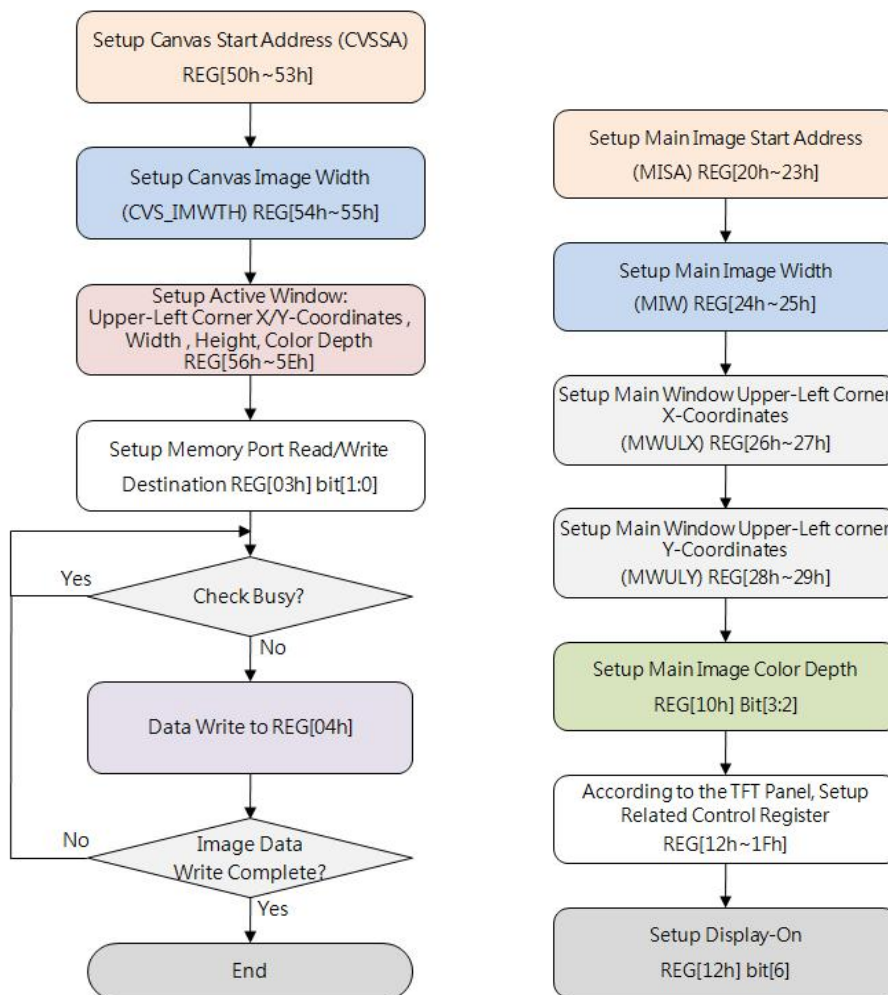


Figure 5-3: Write Image Data to Display Image Buffer

5.2.3 Display Main Window Image

Main window displays the image was selected by setting the Registers REG[20h] ~ REG[29h]. That is, by the register setting to select image within the Display RAM which images to be displayed. The following figure is the process for setting up the main window:

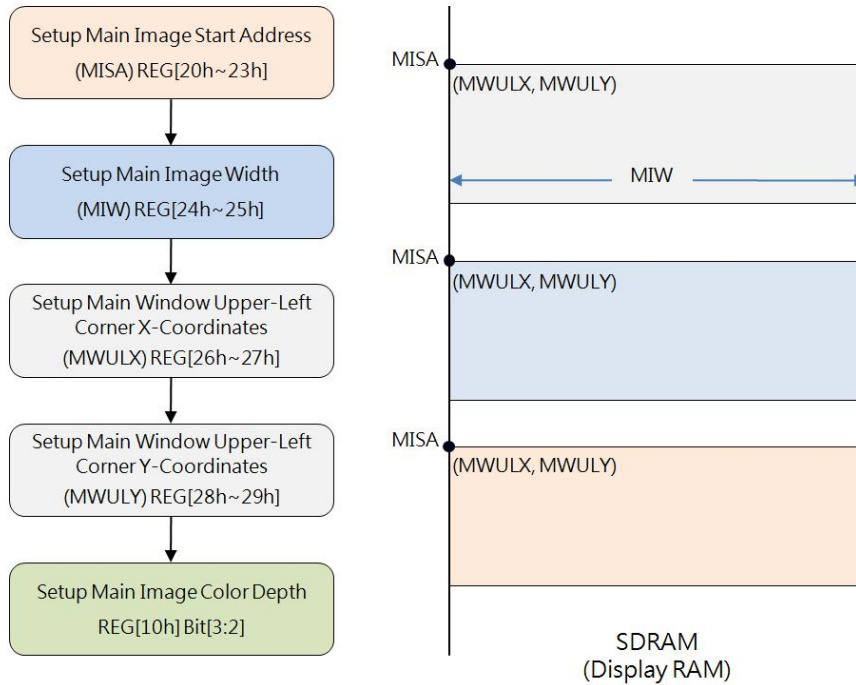


Figure 5-4: Setup and Select Main Window Image

5.3 Picture-In-Picture (PIP)

LT767 supports Picture-in-Picture feature. User can display secondary image(PIP-1) on the main screen(PIP-2) without to overwrite the image data of the main display window. If both images are overlapping, then the PIP-1 images 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]. Because the parameters of PIP-1 and PIP-2 are using the same registers, so REG[10H] Bit4 is used to choose PIP-1 or PIP-2 will be setup by these registers. And in the use of PIP function must first set the relevant parameters of the display window.

The unit of PIP windows sizes and start positions is 4 pixels in horizontal, and 1 pixel in vertical. In PIP mode, LT7381 does not support the overlapping of transparent display, and when REG[12h] Bit3 VDIR = 1, then PIP windows, graphics cursors and text cursors functions will be automatically prohibited.

5.3.1 PIP Window Setting

For using PIP feature, 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 show the corresponding display to main window.

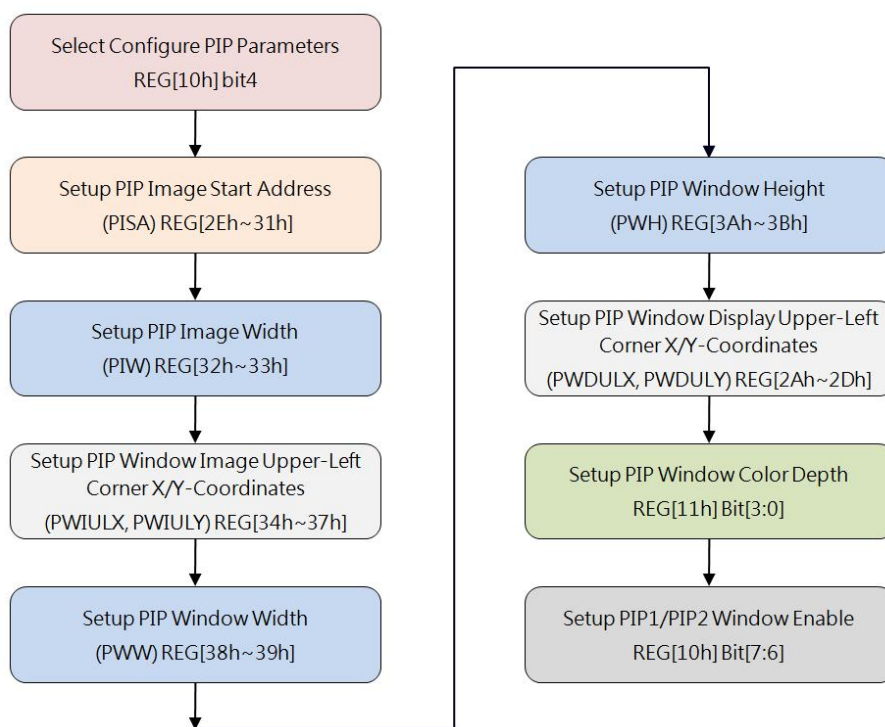


Figure 5-5: Initialize PIP Function

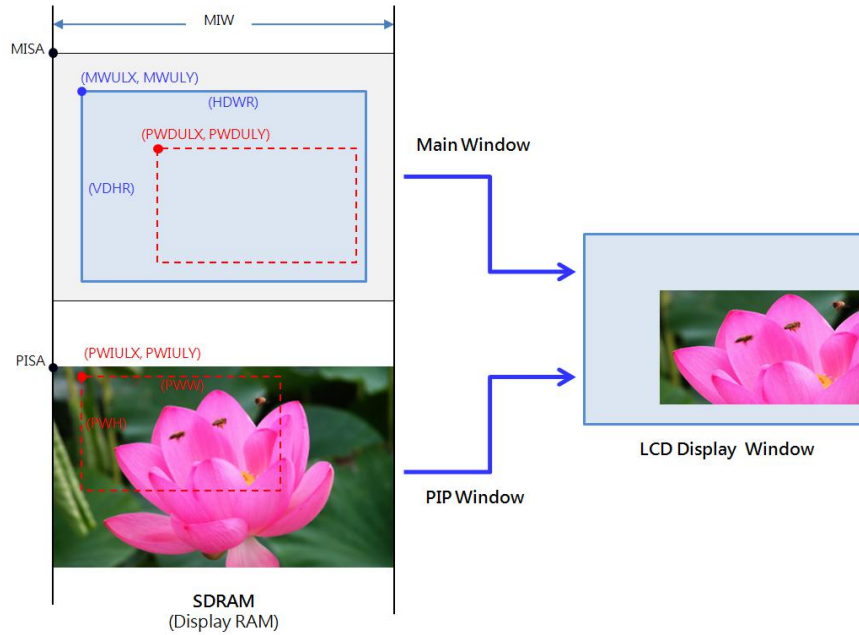


Figure 5-6: PIP Example

5.3.2 PIP Display Position and PIP Image Position

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

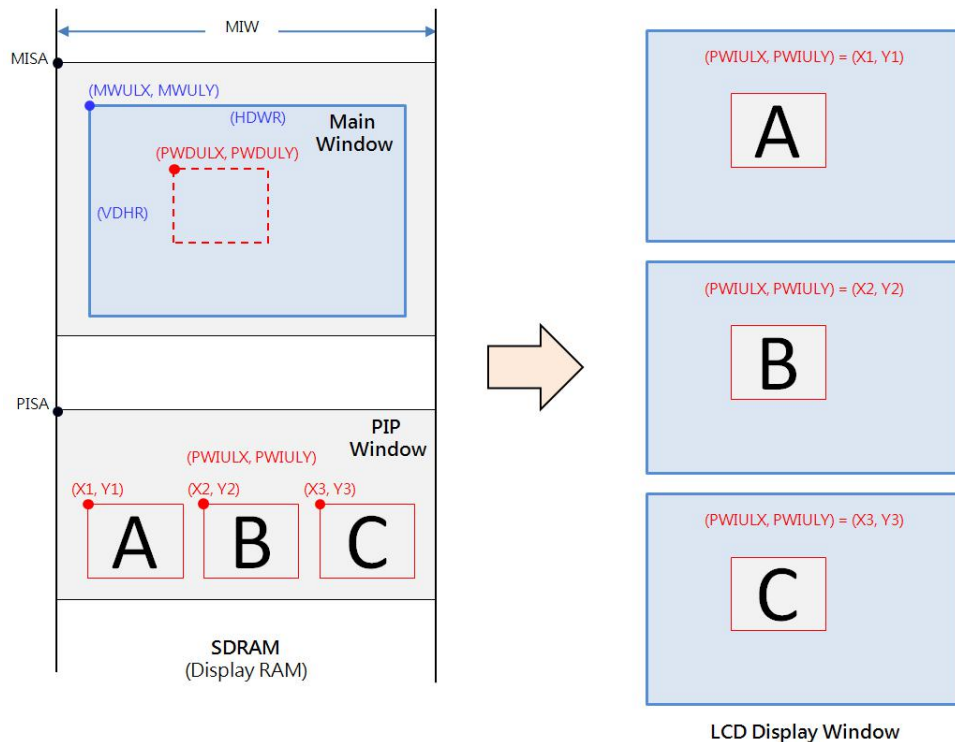


Figure 5-7: Select Different PIP image on the Screen

5.4 Image Rotate and Mirror

Usually LCD displays are refreshed horizontally (from left to right and from top to bottom), and the displayed images are stored in the same way. LT7381 provides the ability to Rotate and Mirror, in which 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 to design mirroring. LT7381 embedded a hardware controller to execute Rotation and Mirroring features, so it's greatly saving the Host processing time.

The REG[02H] bit[2:1] are used to control the Memory Store Direction in which the Host write Image Data to Display RAM. And these two bits is 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°)

The following are some examples for Image Rotate and Mirror:



Figure 5-8: Original Image – without Rotation

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

When set 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 that same as above. If set 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 of the following:



Figure 5-9: Horizontal Mirror Image

When set REG[02H] bit[2:1] 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 following Figure:



Figure 5-10: Rotated to the Right 90° and Flipped Horizontally

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



Figure 5-11: Rotate 90° to the Left

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

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



Figure 5-12: Flip with Vertical

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



Figure 5-13: Rotated 180°

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



Figure 5-14: Rotate 90° to the Left

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



Figure 5-15: Rotated to the Left 90° then Vertically Mirrored

6. Geometric Drawing Engine

6.1 Drawing Circle and Ellipse

LT7381 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 specify REG[76h] bit[5:4] to 00b. Finally enable the Drawing Circle function (REG[76h] bit7 = 1), then LT7381 will drawing a Circle or Ellipse on the screen in automatically.

Host can also set REG[76h] Bit6 to 1 to fill a Circle or an Ellipse. Therefore, Host(MCU) does not need use many resources to calculate the location and a series of writing data to display memory. The Ellipse has two radius. For drawing Circle, just set the long radius and short radius register to the same value ($R1 = R2$).

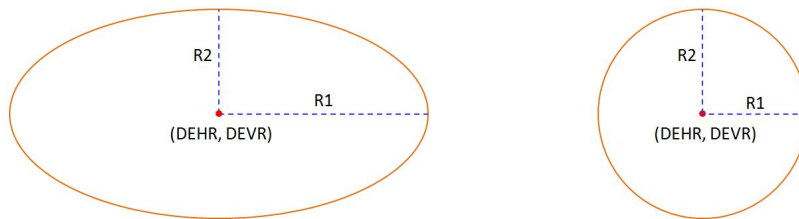


Figure 6-1: Drawing Ellipse and Circle

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

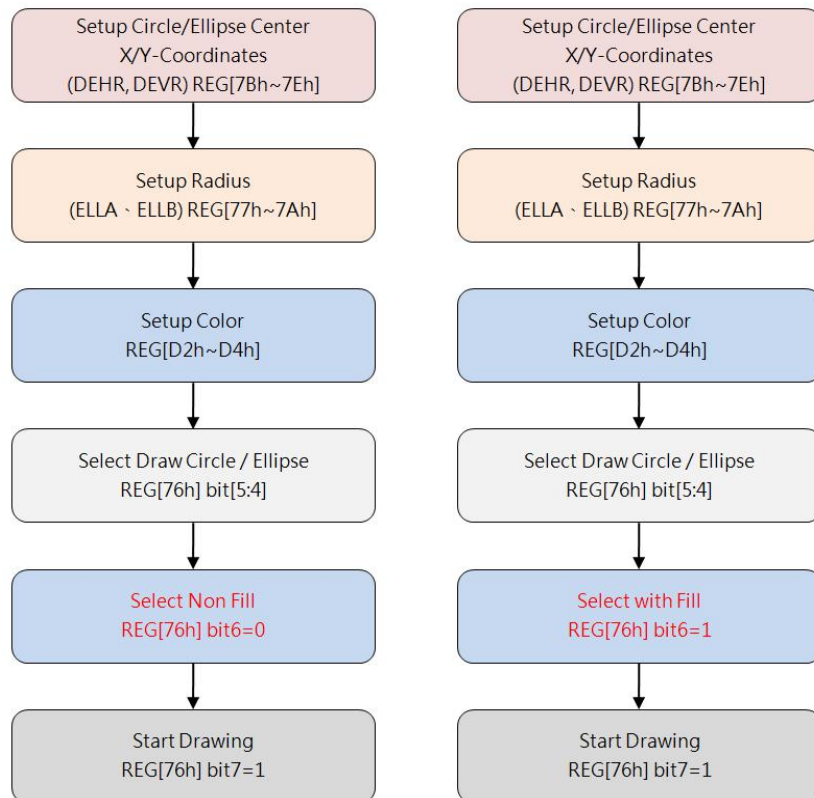


Figure 6-2: Flowchart of Drawing Circle and Ellipse

6.2 Drawing Curve

LT7381 supports the function of drawing Curve. 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 specify REG[76h] bit[5:4] to 01b. Finally enable the Drawing Curve function (REG[76h] bit7 = 1), then LT7381 will drawing a one-fourth Curve on the screen in automatically.

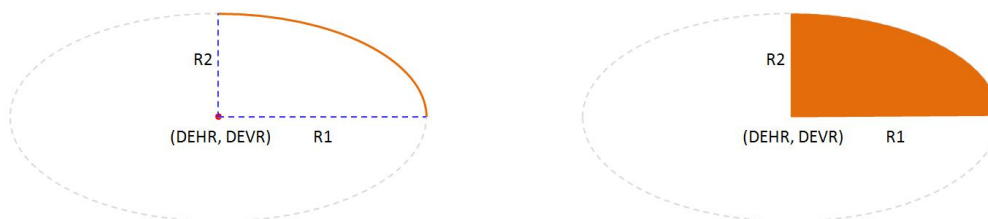


Figure 6-3: Drawing Curve and one-fourth Ellipse

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

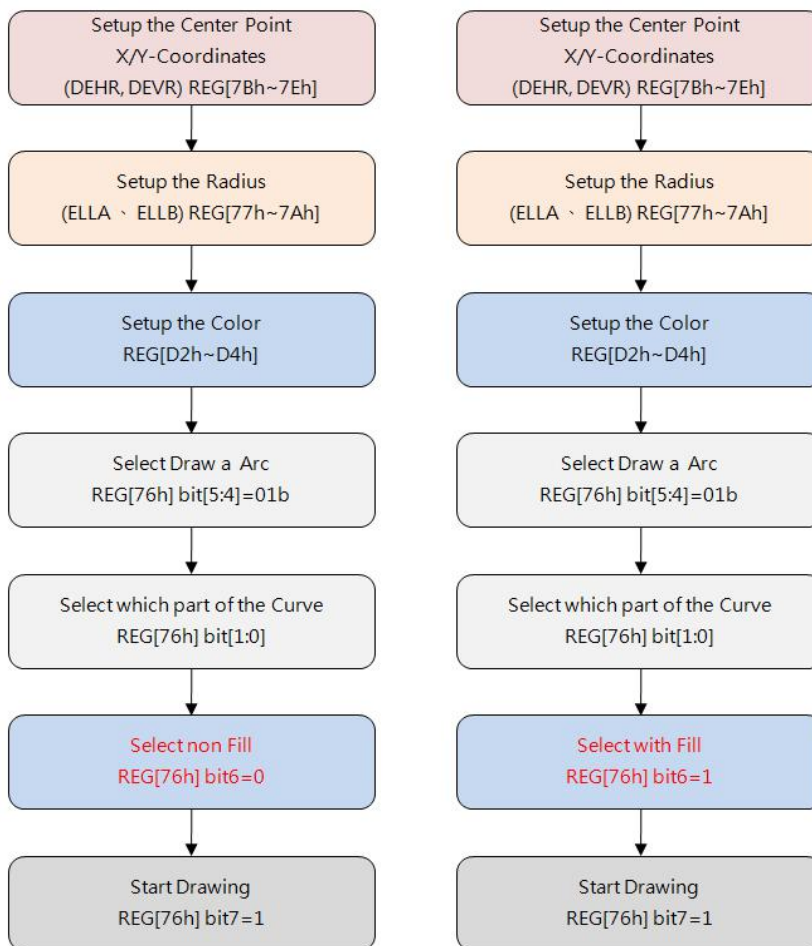


Figure 6-4: Flowchart of Drawing Curve and one-fourth Ellipse

6.3 Drawing Rectangle

For Drawing a Rectangle, the Host has to sets 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. Finally enable the drawing function (REG[76h] bit7 = 1), then LT7381 will drawing a Rectangle on the screen in automatically. Host can also set REG[76h] Bit6 to 1 to fill the Rectangle with specified color.

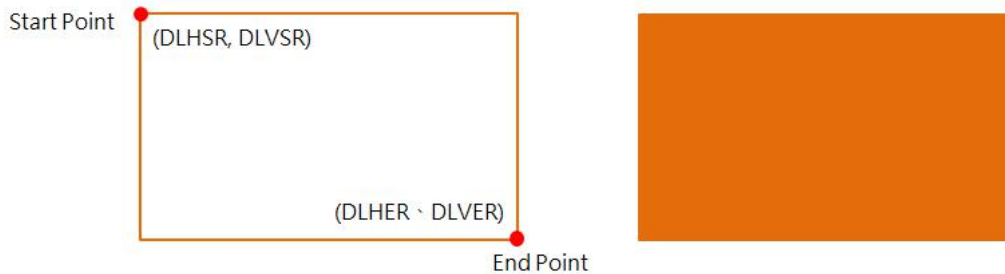


Figure 6-5: Drawing Rectangle

The following figure is the flowchart of drawing Rectangle. The left flowchart is draw a Rectangle without fill, and the right flowchart is with fill. Please note the Start and Stop point must be located in the Active Window.

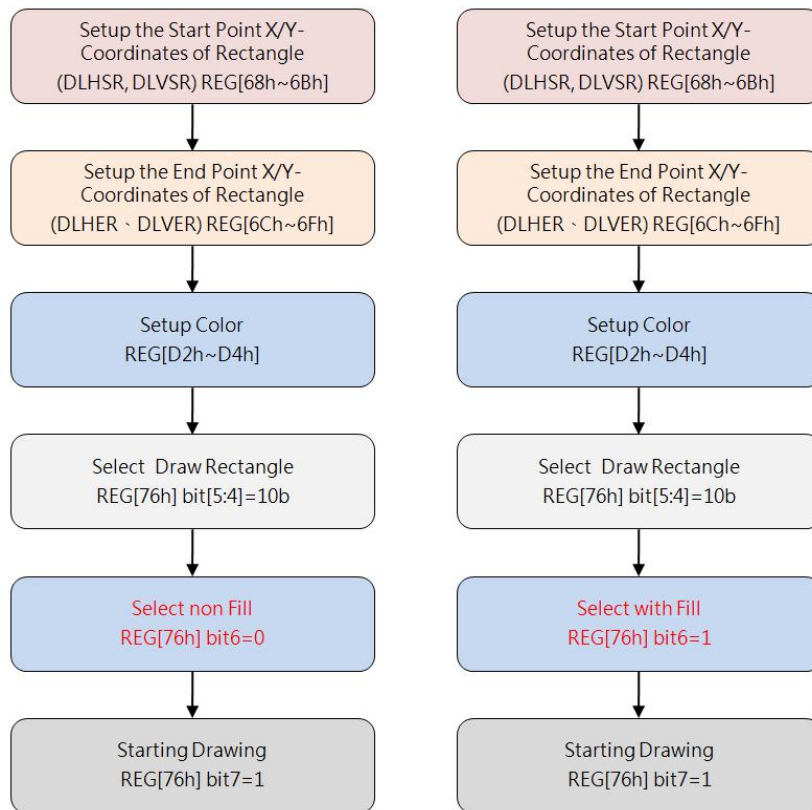


Figure 6-6: Flowchart of Drawing a Rectangle

6.4 Draw Line

For Drawing a Line, the Host has to sets the Start Point Coordinates (REG[68h ~ 6Bh]), Stop Point Coordinates (REG[6Ch ~ 6Fh]), and the Color of the Line (REG[D2h ~ D4h]), then specify and specify REG[67h] bit1 to 0. Finally enable the Line Drawing function (REG[67h] bit7 = 1), then LT7381 will drawing a Line on the screen in automatically.

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

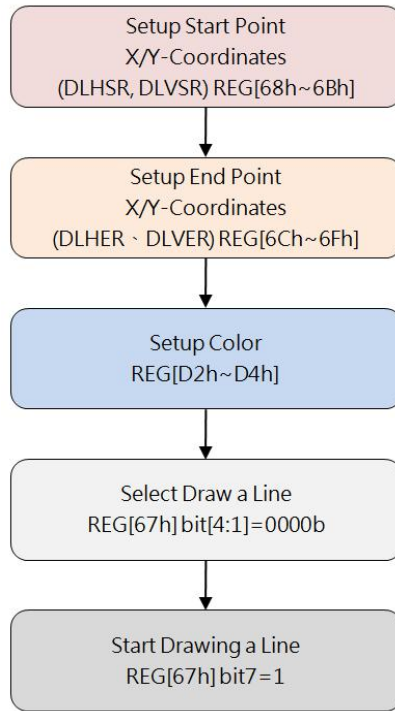


Figure 6-7: Flowchart of Drawing a Line

6.5 Drawing Triangle

For Drawing a Triangle, the Host has to sets the 1ST Point Coordinates (REG[68h ~ 6Bh]) of Triangle, 2nd Point Coordinates (REG[6Ch ~ 6Fh]), 3rd Point Coordinates (REG[70h ~ 73h]), and the Color of the Triangle (REG[D2h ~ D4h]), then specify REG[67h] bit1 to 1. Finally enable the drawing function (REG[67h] bit7 = 1), then LT7381 will drawing a Triangle on the screen in automatically. Host can also set REG[67h] bit5 to 1 to fill the Rectangle with specified color.

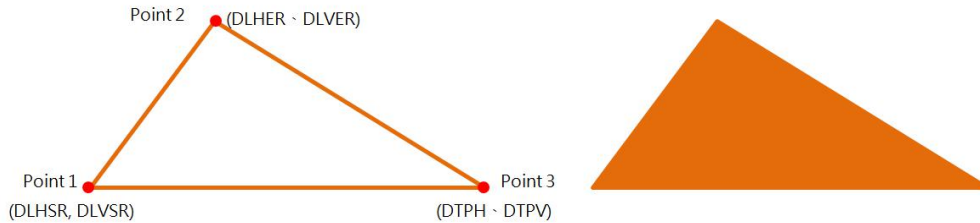


Figure 6-8: Drawing Triangle

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

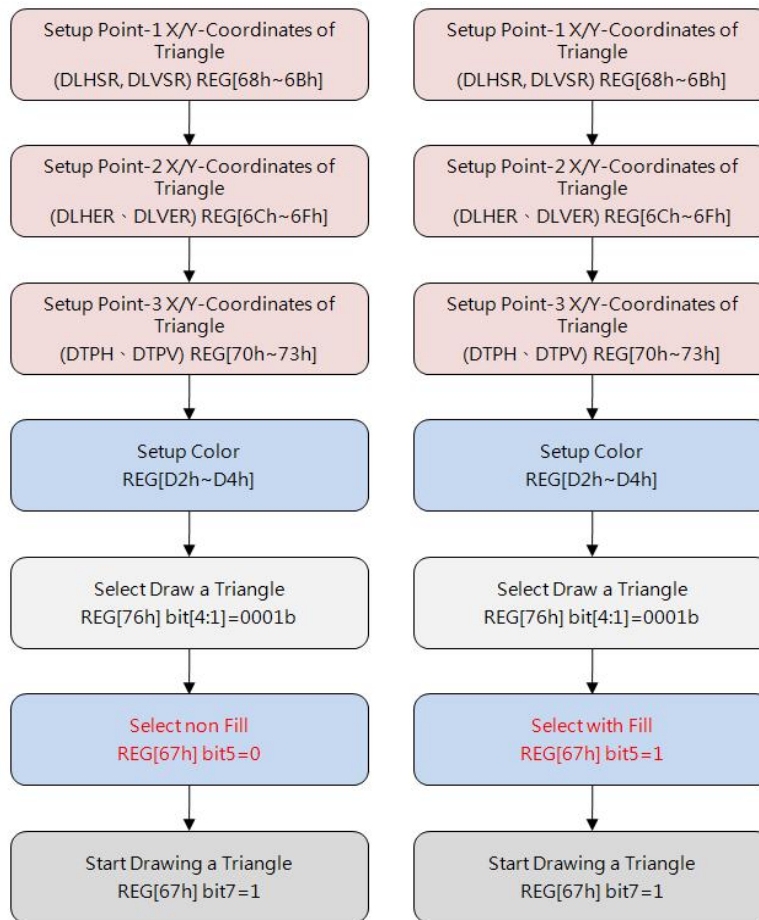


Figure 6-9: Flowchart of Drawing a Triangle

6.6 Drawing Rounded-Rectangle

For Drawing a Rounded-Rectangle, the Host has to sets the Start Point Coordinates (REG[68h ~ 6Bh]), Stop 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. Finally enable the drawing function (REG[76h] bit7 = 1), then LT7381 will drawing a Rounded-Rectangle on the screen in automatically. Host can also set REG[76h] Bit6 to 1 to fill the Rounded-Rectangle with 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.

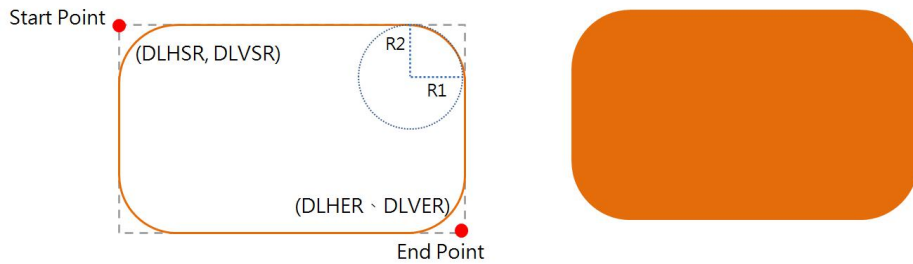


Figure 6-10: Drawing Rounded Triangle

Note1: DLHER-DLHSR must large than $2 * R1 + 1$

Note2: DLVER-DLVSR must large than $2 * R2 + 1$

The following figure is the flowchart of drawing Rounded-Rectangle. The left flowchart is draw a Triangle without fill, and the right flowchart is with fill. Please note the Start and Stop point must be located in the Active Window.

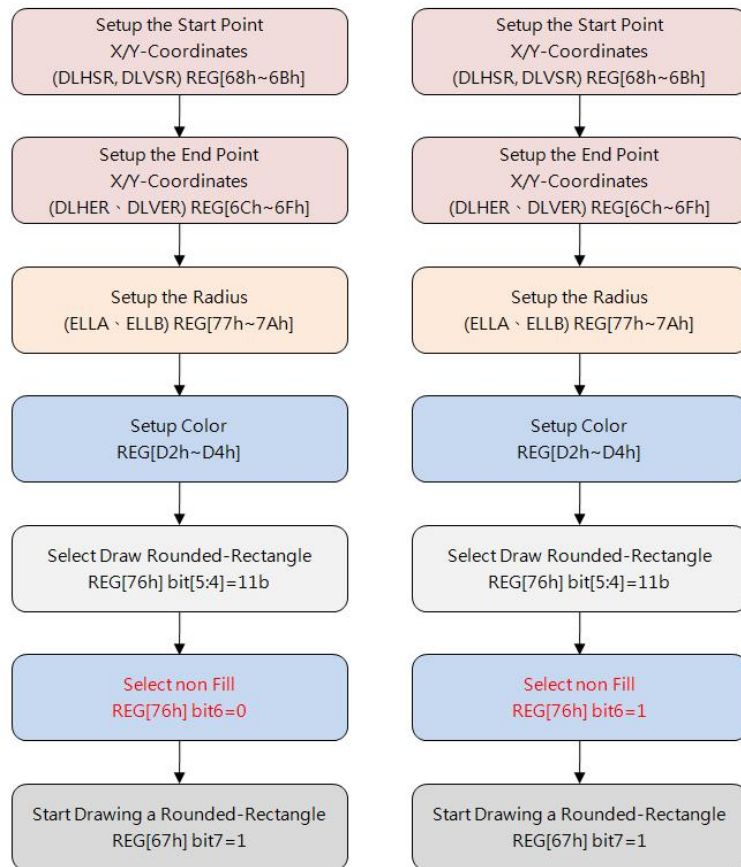


Figure 6-11: Flowchart of Drawing a Rounded-Rectangle

7. Block Transfer Engine (BTE)

LT7381 has a 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 7-1), as well as additional functions of Raster Operation (ROP, Table 7-2), 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 release MCU working loads and resources, further to greatly improve the performance for all the system.

If a data block needs to be loaded, moved, processed at one times, user could consider to use BTE, by any available combinations of basic BTE Operations and additional Functions as well as various available logic combinations of Source and Destination, further to achieve many useful application purposes. Once the BTE started under proper settings, it will keep active working (busy) until the working completed. There are two ways to get BTE working status, the one is to monitor Hardware Interruption: connecting INT# to MCU, once the interruption presented, then MCU responds and starts the routine to get the Interruption Source by checking REG[0Ch] ; the another one is to check corresponding Flag Bits of register BTE_CTRL0 (REG[90h]) Bit4, or Status Register (STSR) Bit3.

Table 7-1: 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 7-2: ROP Function

| ROP Function Code REG[91h] bit[7:4] | Function Description (Boolean) |
|--|---|
| 0000b | 0 (Blackness) |
| 0001b | $\sim S0 \cdot \sim S1$ or $\sim (S0+S1)$ |
| 0010b | $\sim S0 \cdot S1$ |
| 0011b | $\sim S0$ |
| 0100b | $S0 \cdot \sim S1$ |
| 0101b | $\sim S1$ |
| 0110b | $S0 \wedge S1$ |
| 0111b | $\sim S0 + \sim S1$ or $\sim (S0 \cdot S1)$ |
| 1000b | $S0 \cdot S1$ |
| 1001b | $\sim (S0 \wedge S1)$ |
| 1010b | $S1$ |
| 1011b | $\sim S0 + S1$ |
| 1100b | $S0$ |
| 1101b | $S0 + \sim S1$ |
| 1110b | $S0 + S1$ |
| 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 wrote to Memory
4. Memory means internal Display RAM
5. Memory, S0, S1, DT, these Short Names will be always quoted in this Chapter. For Example: If ROP function REG[91h] bit[7:4]=0xCh, then "DT = S0", means "Transfer Source 0 data to Destination"; If ROP function REG[91h] bit[7:4]=0xEh, then "DT = S0 + S1", means "Source 0 data + Source 1 data then transfer to Destination"

Table 7-3: 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 |

7.1 BTE Basic Settings

The BTE basic settings for all two Sources and one Destination, including Memory Start Address, Image Width, and Start Point Position (Coordinate) for each, could 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[A Eh], REG[AFh], REG[B0h]

7.2 Color Palette RAM

The LT7381 has an embedded Color Palette RAM for 8-bits Alpha Blend function, organized by 64*12bits. Color Index is an Address of Color Palette RAM space to save a series of 12-bits Word of needed color data (refer to Figure 7-1). Color Palette RAM must be written by 128 Bytes (8-bits per Byte) sequence in one times, but cannot be read, bit[7:4] of the Byte with Even Sequence Number will be discarded. The diagram Figure 7-2 shows the initialization flow.

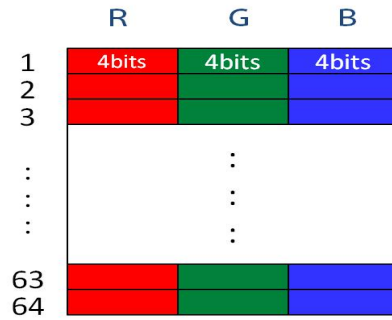


Figure 7-1: Color Palette RAM Organization

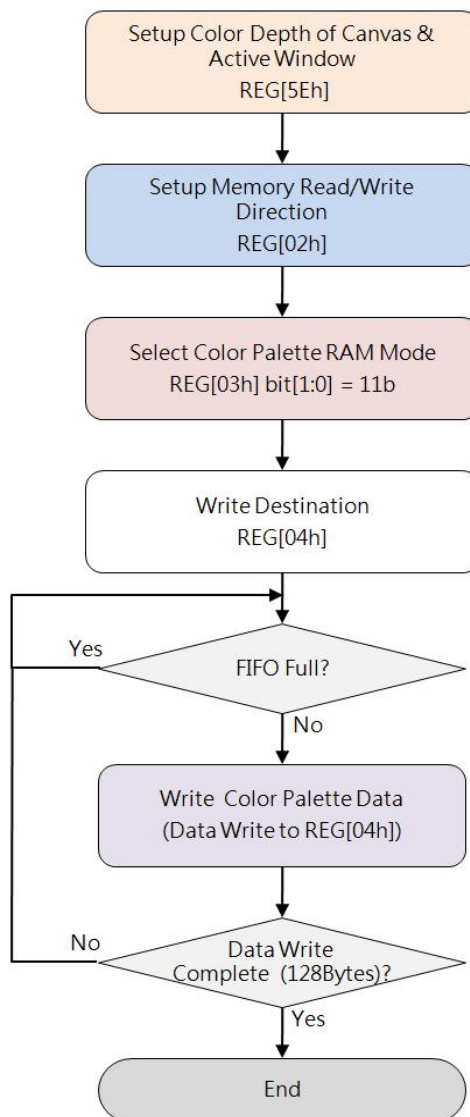


Figure 7-2: Flow Chart of Color Palette RAM Initialization

7.3 BTE Operation Overview

7.3.1 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.

7.3.2 Memory Copy with ROP

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

7.3.3 Solid Fill

This Operation fills a specified Rectangle Area of DT with a Solid Color data defined in the Foreground Color Register.

7.3.4 Pattern Fill

This Operation duplicates to fill DT with an 8*8 or 16*16 pixel Pattern.

7.3.5 Pattern Fill with Chroma Key

This Operation duplicates to fill DT with an 8*8 or 16*16 pixel Pattern, combined with Chroma Key function but without ROP function. If Pattern Color is equal to the Chroma Key Color, which is defined in Background Color Register, destination BTE Window will not be changed.

7.3.6 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, destination BTE Window will not be changed, no ROP applied for this operation.

7.3.7 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, destination BTE Window will not be changed, no ROP applied for this operation.

7.3.8 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, must set the Different Color data in Foreground Color Register (D2h~D4h) and Background Color Register (D5h~D7h).

7.3.9 Memory Copy with Color Expansion

This Operation performs Color Expansion for Source 0 data from Memory, please refer to 7.3.8 for other description and note.

7.3.10 Memory Copy with Opacity

This Operation performs Alpha Blending for S0 data and S1 data and transferring the result data to the destination BTE Window, it requires S0, S1, DT located in Memory. The Alpha Blending has 2 modes, Picture Mode: for all pixels, blending by a mutual Alpha the level data saved in the register. Pixel Mode: for each pixel, blending by individual Alpha Level of each pixel.

7.3.11 MCU Write with Opacity

This Operation performs Alpha Blending for S0 data and S1 data and transferring the result data to destination BTE Window, it requires S0 coming from MCU and S1/DT from Memory, please refer to 7.3.10 for the descriptions of 2 modes.

7.4 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 to define S0 / S1 / DT / BTE_Window, and shows the Directions of Memory Access:

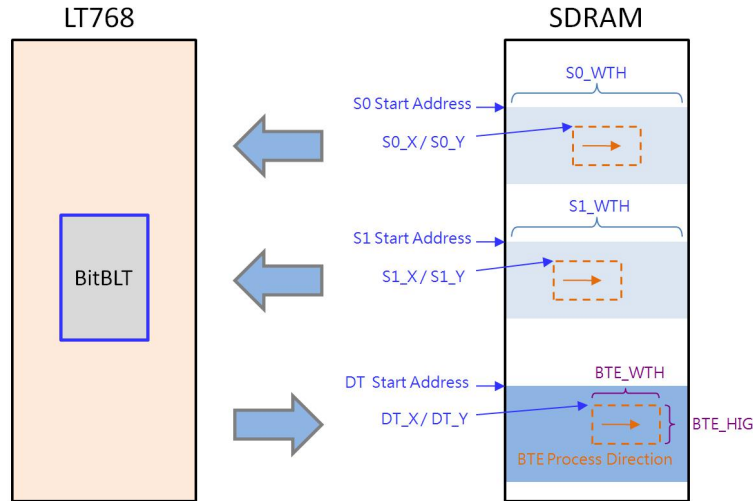


Figure 7-3: BTE Memory Access Method

7.5 BTE Chroma Key (Transparency Color) Function

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

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]

7.6 BTE Operation Detail

7.6.1 MCU Write with ROP

This Operation performs to transfer S0 (form MCU Write) to DT, and supports all 16 ROP functions. Below diagram is an example for the operation result while BTE Control Register 1 REG[91h] set as 0xC0h.

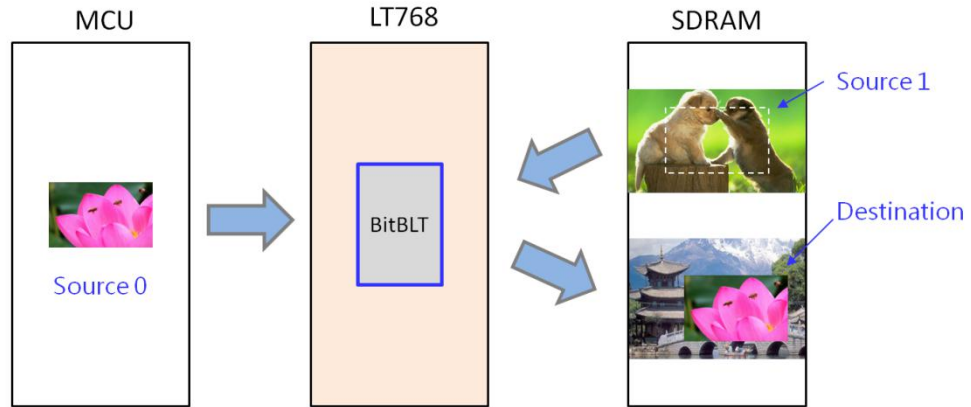


Figure 7-4: Example of MCU Write with ROP

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

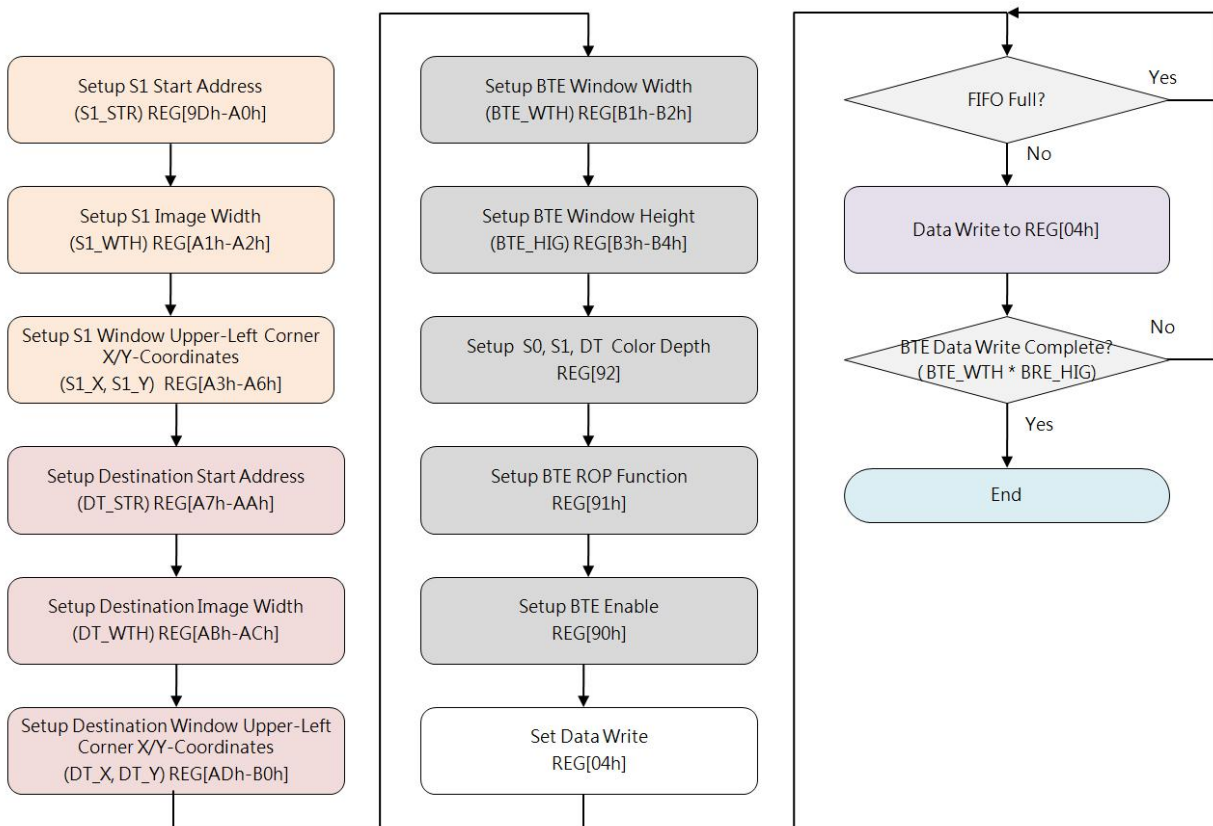


Figure 7-5: Flow Chart of MCU Write with ROP

7.6.2 Memory Copy (move) with ROP

This Operation performs Memory Copy from S0 (from Memory) to DT. Below diagram is an example for the operation result while BTE Control Register 1 REG[91h] set as 0xC2h.

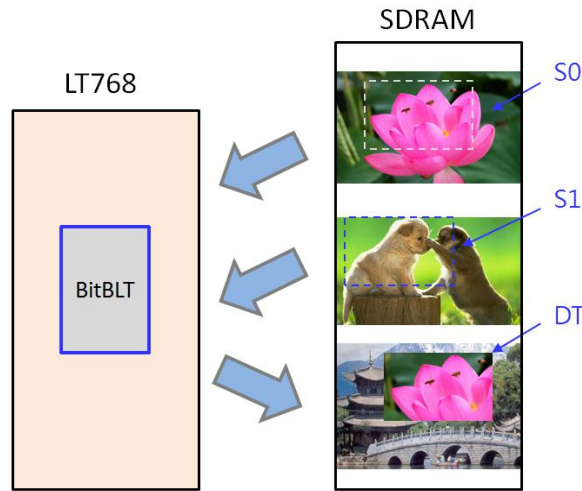


Figure 7-6: Example of Memory Copy with ROP

Figure 7-7 is Flow Chart by checking Status Register STSR bit3 to monitor BTE busy or not. Figure 7-8 is Flow Chart by responding Hardware Interruption to get BTE processing result.

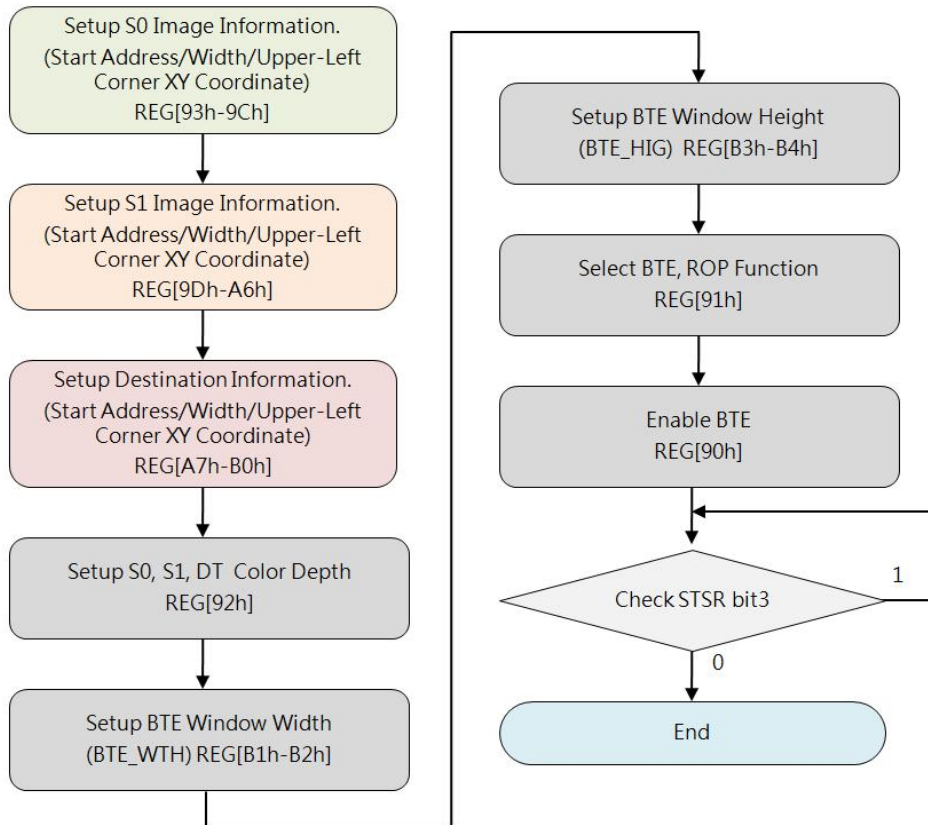


Figure 7-7: Flow Chart 1 of Memory Copy with ROP

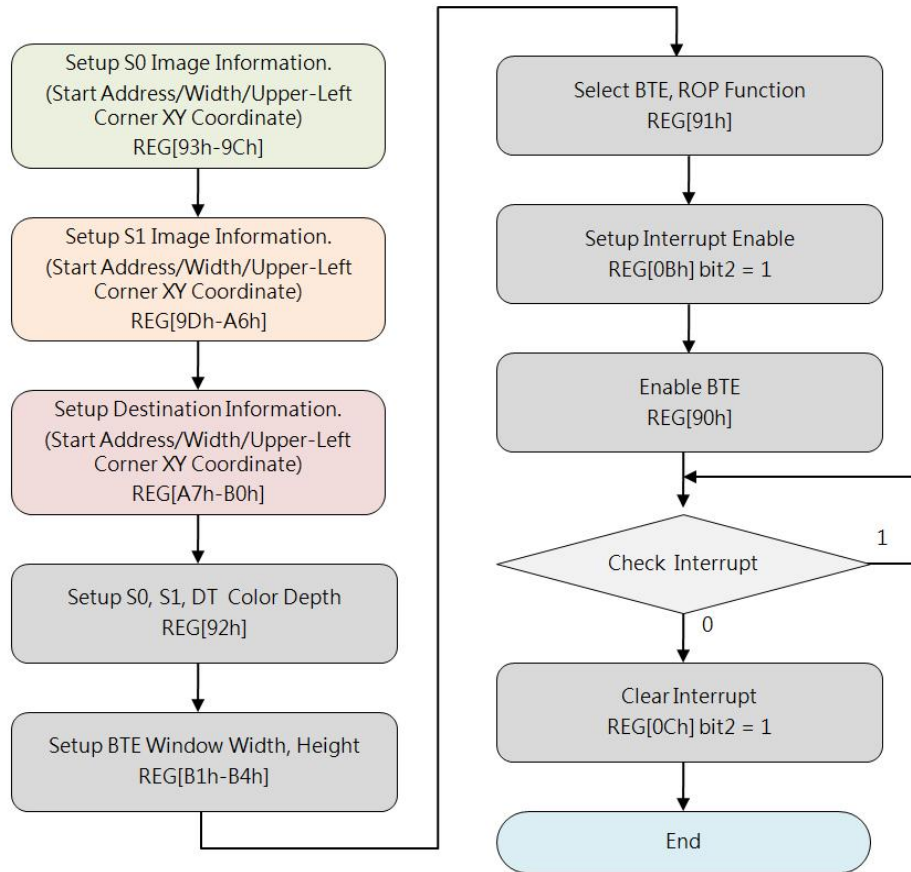


Figure 7-8: Flow Chart 2 of Memory Copy with ROP

7.6.3 MCU Write with Chroma Key (w/o ROP)

This Operation performs to transfer S0 (from MCU Write) to DT, and supports Chroma Key function (referring to 7.5).

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, means BTE will discard writing that Color data to DT. Below Example shows GREEN is background color of RED "TOP", and GREEN is set as Chroma Key, BTE will write RED "TOP" only to DT:

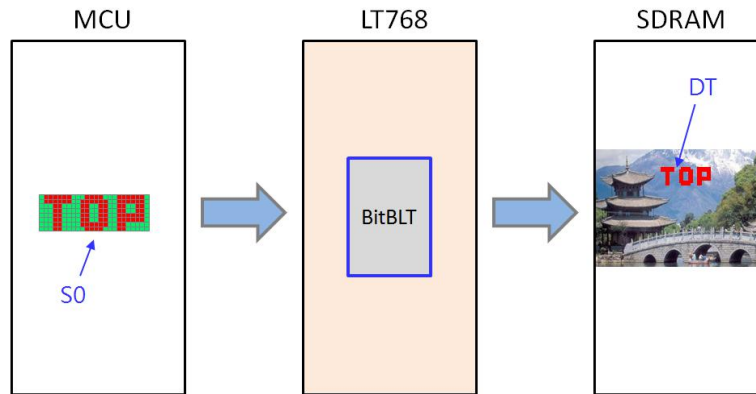


Figure 7-9: Example of MCU Write with Chroma Key

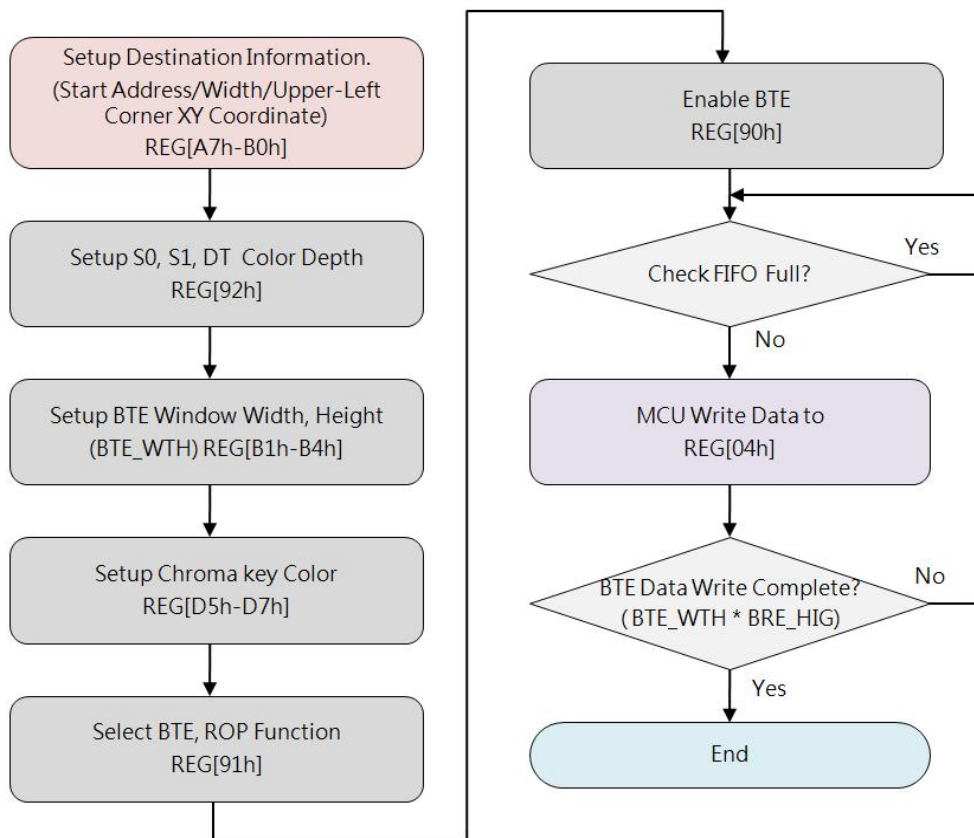


Figure 7-10: Flow Chart of MCU Write with Chroma Key

7.6.4 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 7.5).

The difference comparing with “MCU Write with Chroma Key” is that S0 comes from Memory but not MCU Write, please refer to Section 7.6.3 for more descriptions and Figure 7-11 for an example:

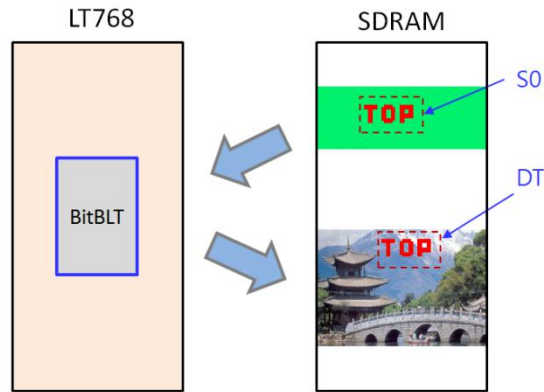


Figure 7-11: Example of Memory Copy with Chroma Key

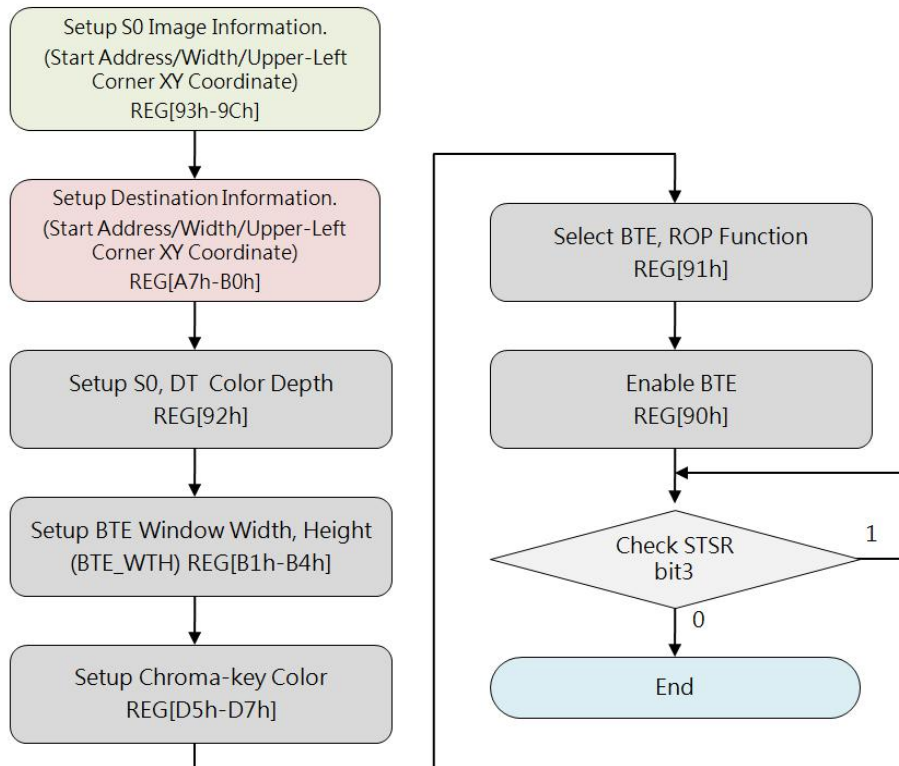


Figure 7-12: Flow Chart of Memory Copy with Chroma Key

7.6.5 Pattern Fill with ROP

This Operation performs 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] set as 0x0C:

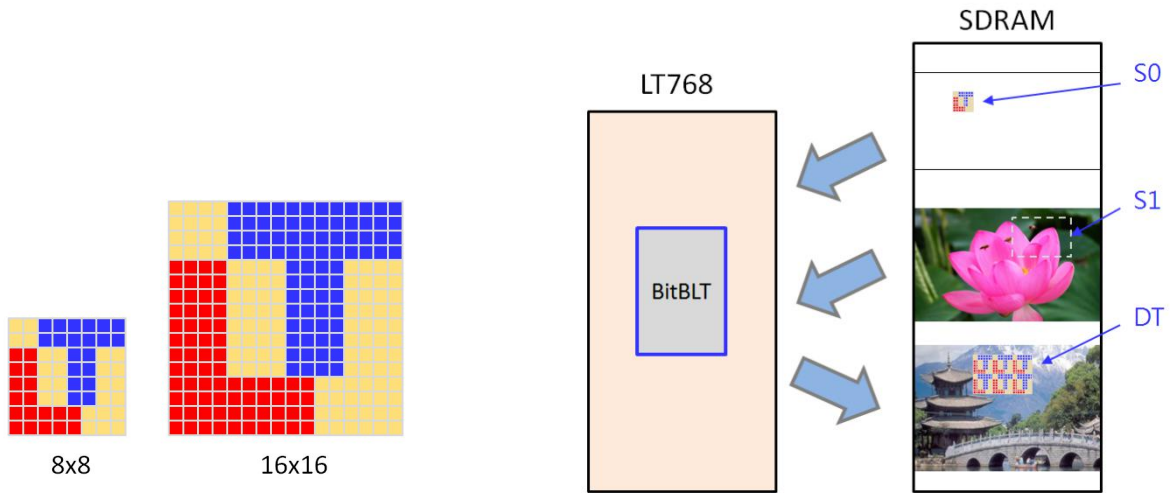


Figure 7-13: Pattern Format

Figure 7-14: Example of Pattern Fill with ROP

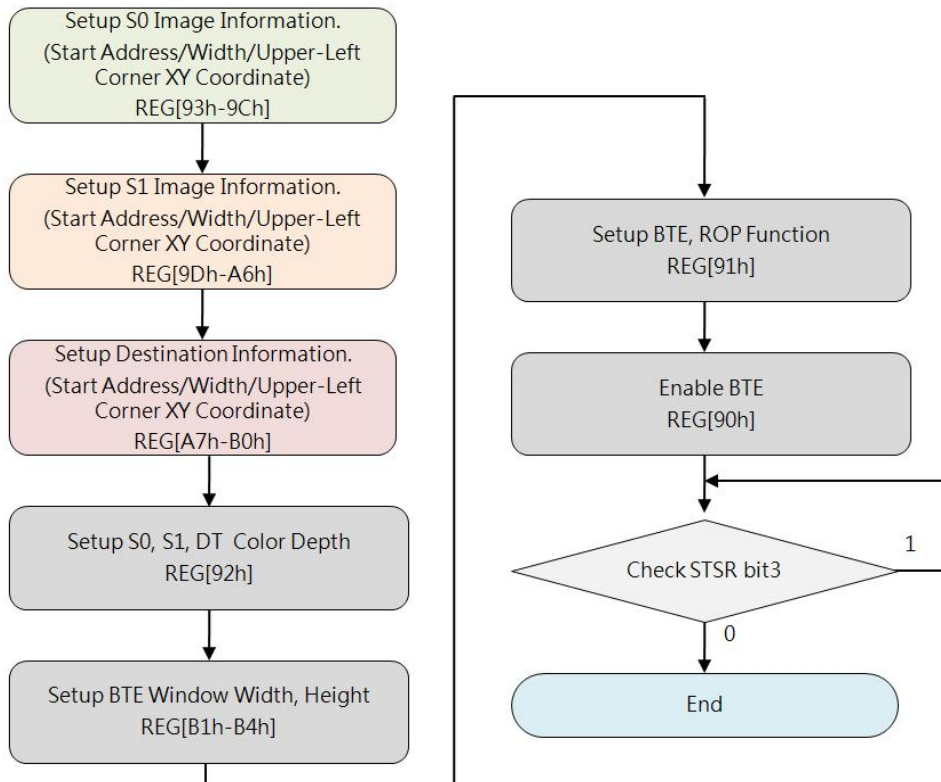


Figure 7-15: Flow Chart of Pattern Fill with ROP

7.6.6 Pattern Fill with Chroma Key

This Operation performs 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 7.5). If any partial Color Data of the pattern is equal to the Chroma Key data, BTE will discard that partial fillings.

Below Example shows ORANGE is background color of DT, RED “T”, and ORANGE is set as Chroma Key, BTE will fill RED “T” only to DT one by one:

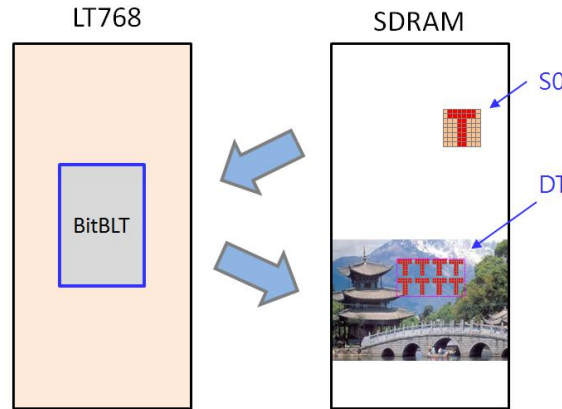


Figure 7-16: Example of Pattern Fill Chroma Key

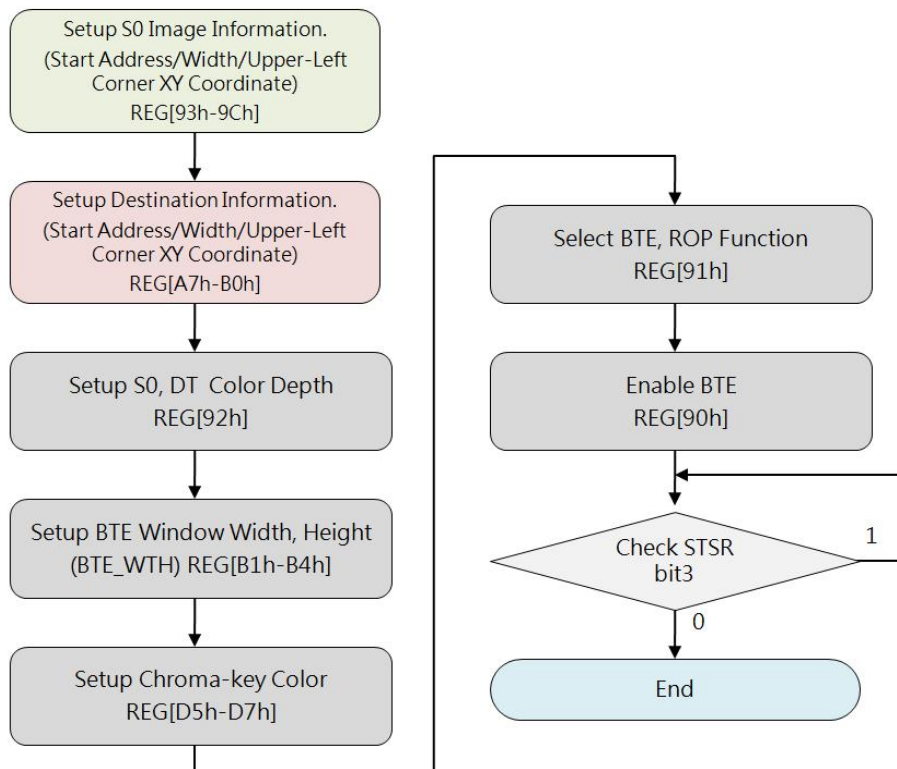


Figure 7-17: Flow Chart of Pattern Fill with Chroma Key

7.6.7 MCU Write with Color Expansion

This Operation performs Color Expansion for S0 (from MCU Write), it 's 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), such as 8-bits MCU Interface the Color Depth is 8-bits and 16-bits MCU Interface the Color Depth is 16-bits. User 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 sequently expands bit by bit until the BTE Width reached 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:

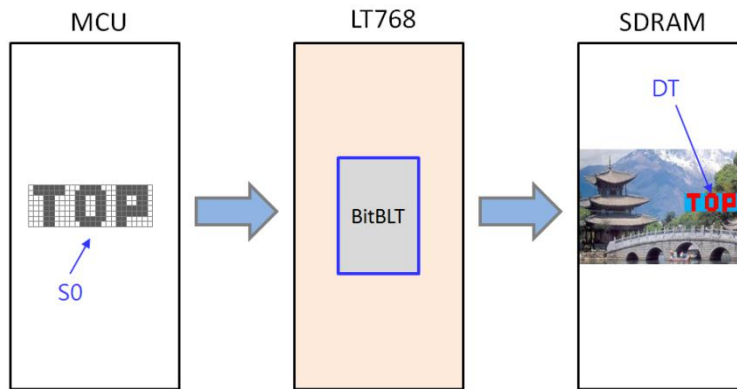


Figure 7-18: Example of MCU Write with Color Expansion

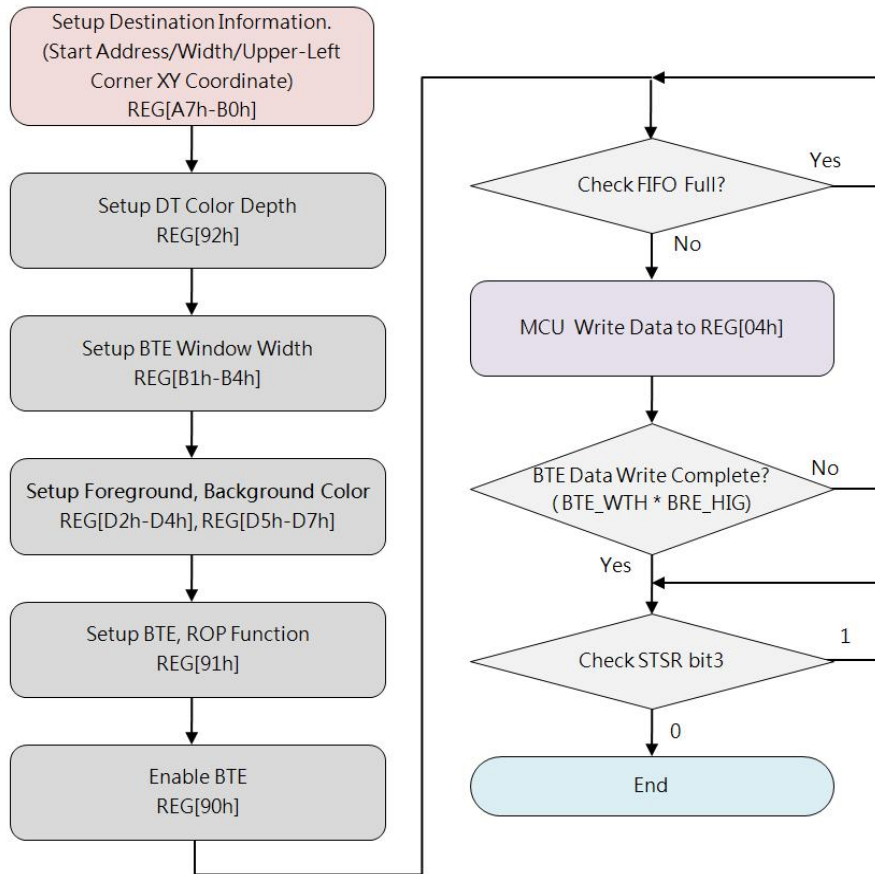


Figure 7-19: Flow Chart of MCU Write with Color Expansion

For 8-bits MCU Interface, if Foreground Color set to RED, Background Color set to KHAKI, and BTE Width set to 23, more examples please refer to Figure 7-20 (ROP = 7) and Figure 7-21 (ROP = 3).

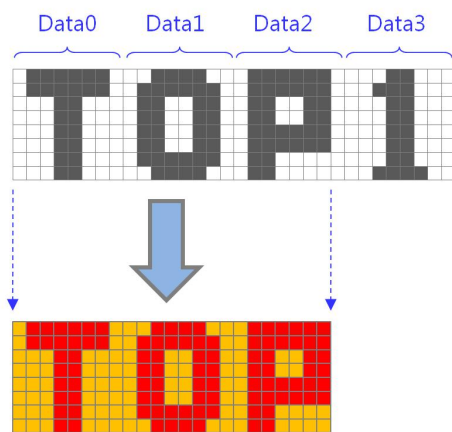


Figure 7-20: Example 1 of MCU Write with Color Expansion (ROP=7)

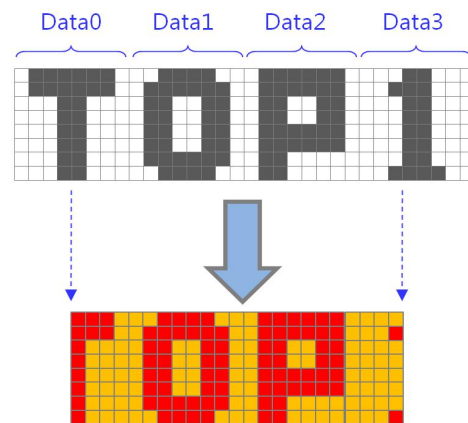


Figure 7-21: Example 2 of MCU Write with Color Expansion (ROP=3)

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

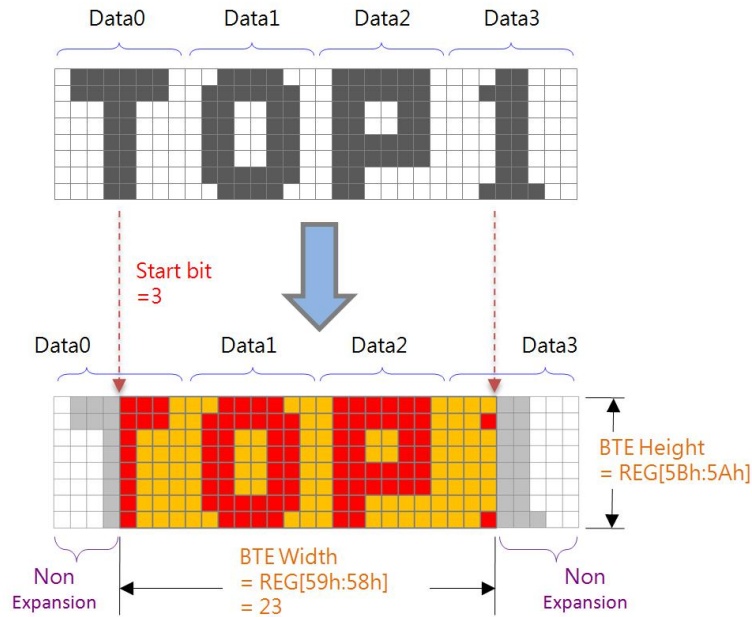


Figure 7-22: Data Format for Color Expansion

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:

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

7.6.8 MCU Write with Color Expansion and Chroma key

This Operation is similar with 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 expanding data_1b to RED (Foreground Color) and data_0b discarded, so the result to DT is RED "TOP" with TRANSPARENT background:

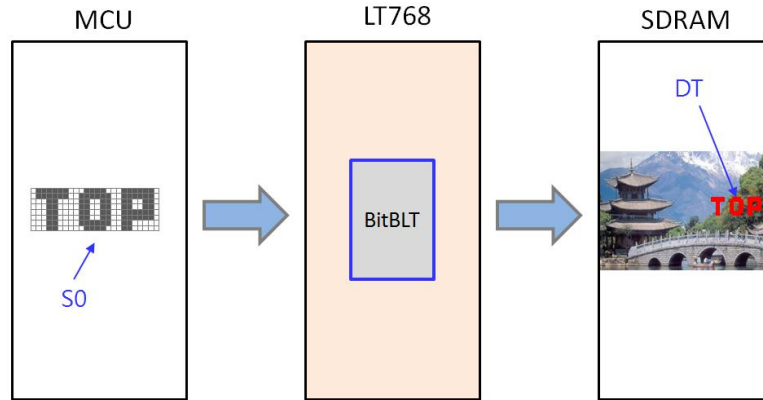


Figure 7-23: Example of MCU Write with Color Expansion and Chroma key

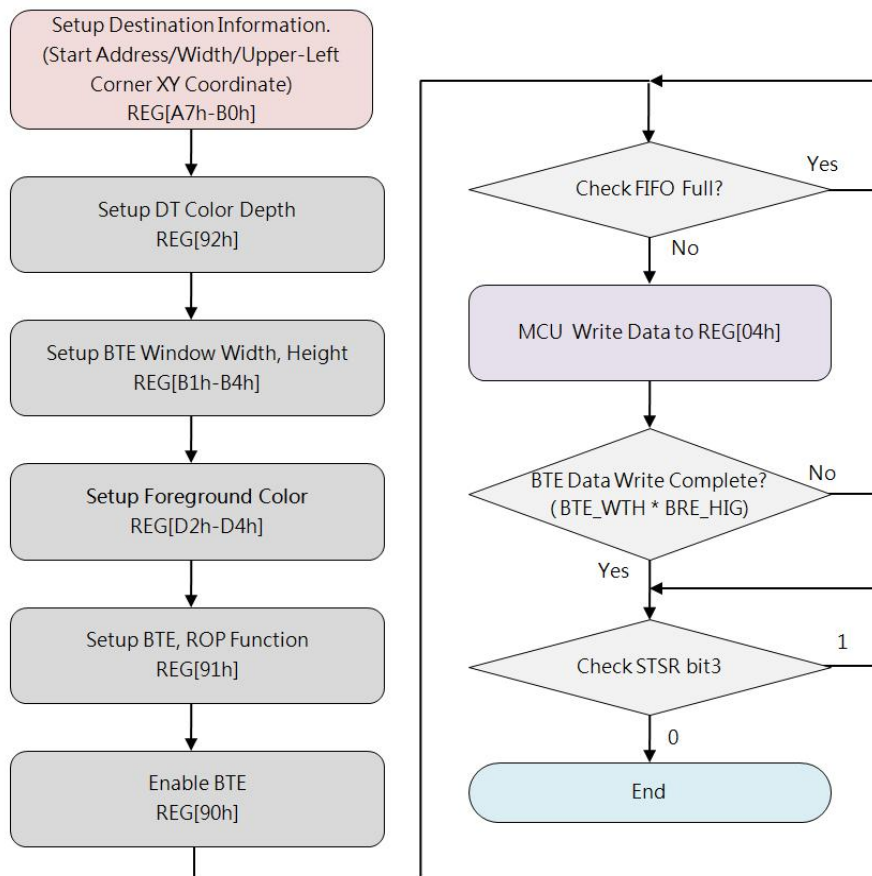


Figure 7-24: Flow Chart of MCU Write with Color Expansion and Chroma key

7.6.9 Memory Copy with Opacity

This Operation performs blending S0 and S1 data and transferring the result to DT, two mode are available: Picture Mode and Pixel Mode.

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

Pixel Mode is for 8bpp/16bpp format and using individual Opacity Value of S1, each pixel of S1 has its own Opacity Value, such as: for one 16bpp data, the bit[15:12] is Opacity Value, the bit[11:0] is color data; for one 8bpp data of S1, the bit[7:6] is Opacity Value, 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))

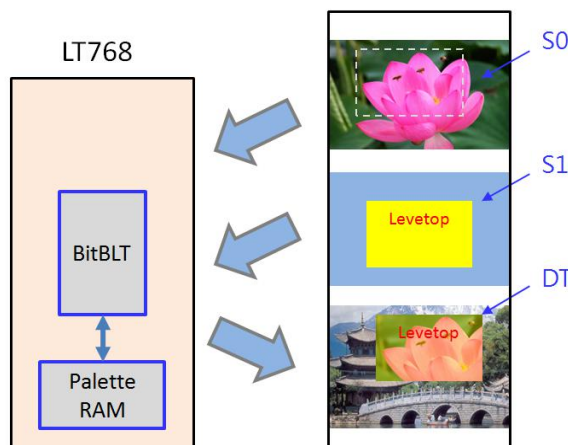


Figure 7-25: Example of Pixel Mode - 8bpp

Table 7-4: Alpha Blending of Pixel Mode - 8bpp

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

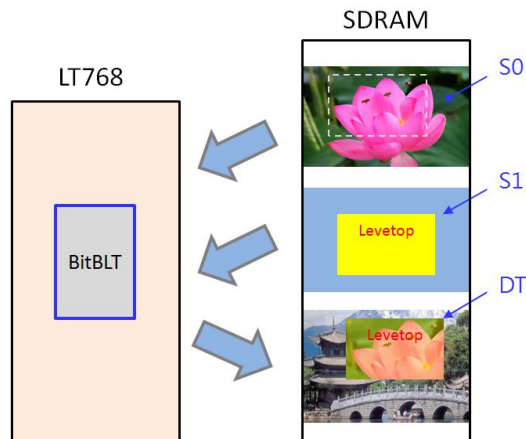


Figure 7-26: Example of Pixel Mode - 16bpp

Table 7-5: 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 |

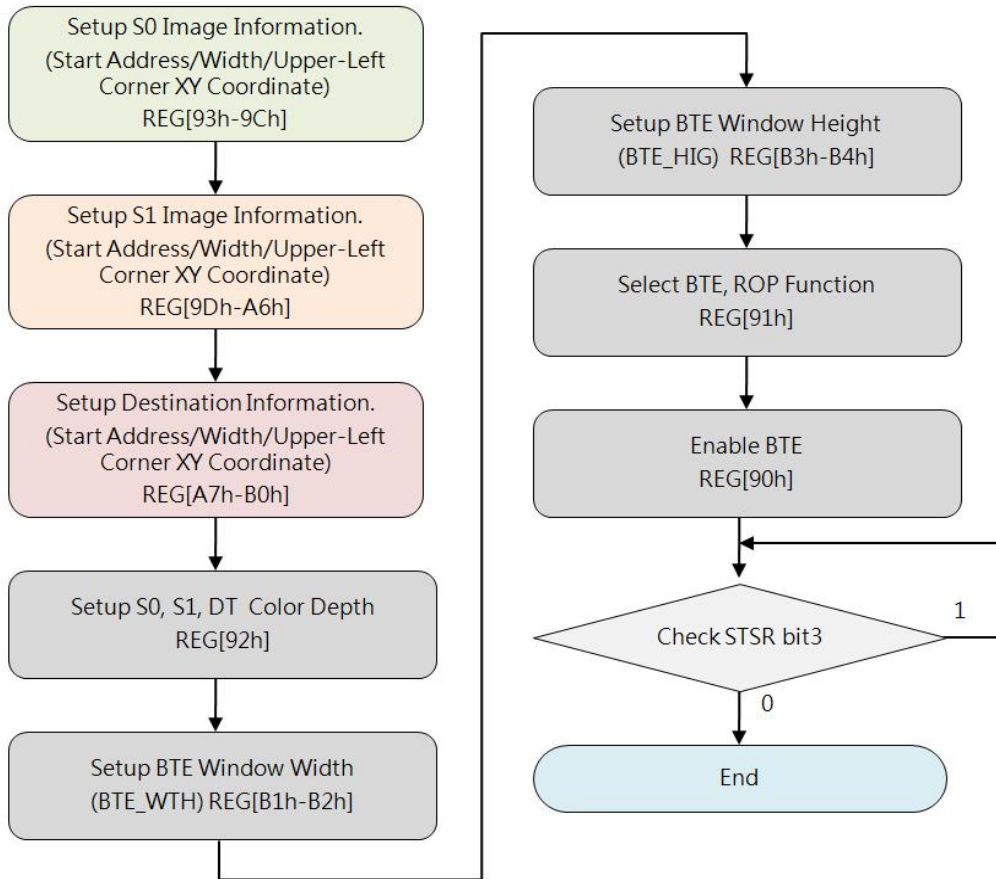


Figure 7-27: Flow Chart of Memory Copy with Opacity -Pixel Mode

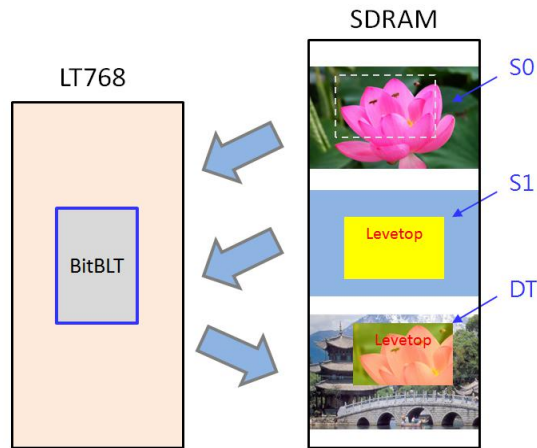


Figure 7-28: Example of Memory Copy with Opacity - Picture Mode

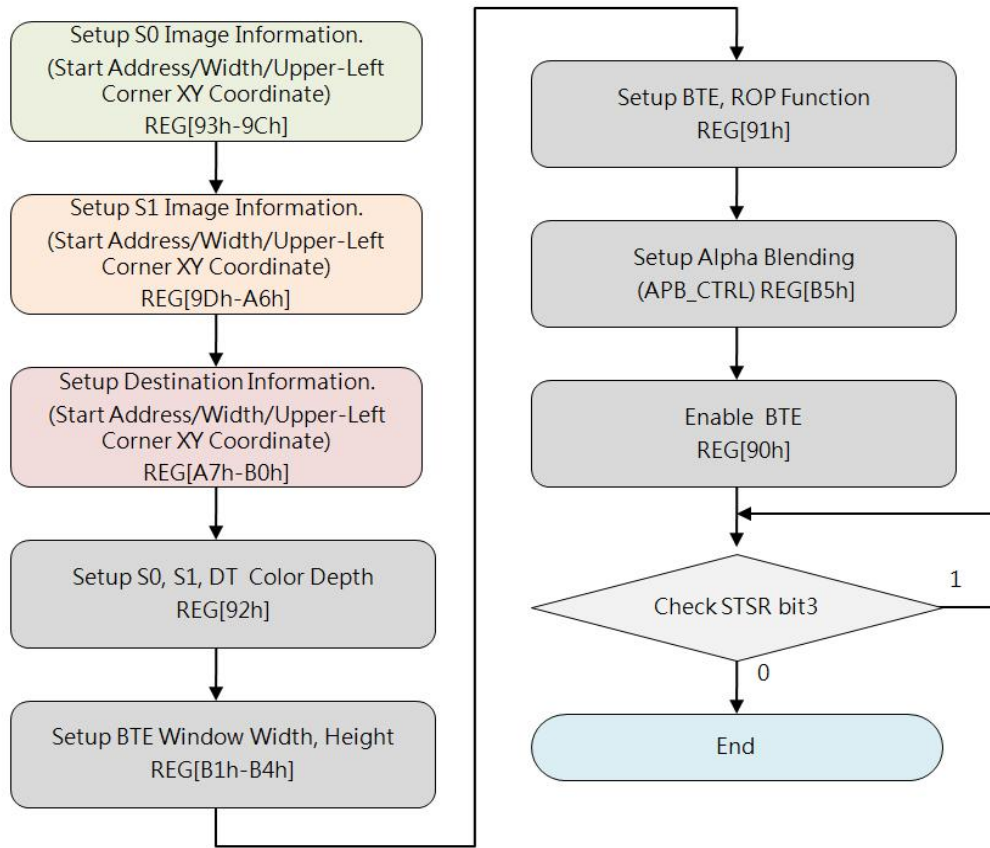


Figure 7-29: Flow Chart of Memory Copy with Opacity - Picture Mode

7.6.10 MCU Write with Opacity

This Operation is similar with Memory Copy with Opacity, but the difference is that S0 is from MCU Write, and has same Picture Mode and Pixel Mode, for more descriptions please refer to Section 7.6.9.

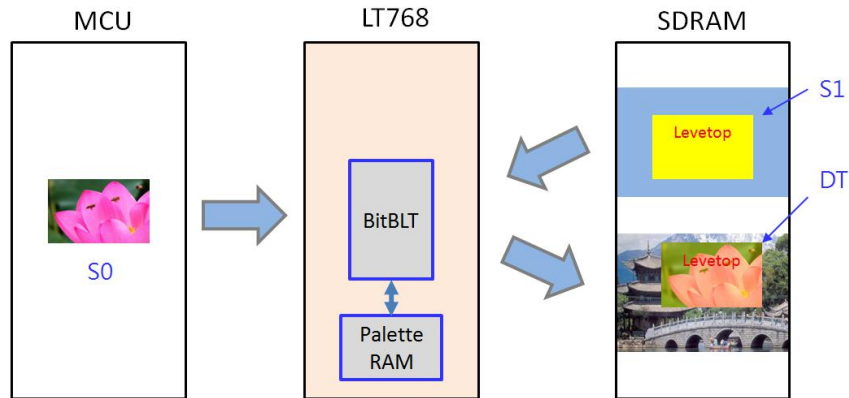


Figure 7-30: Example of MCU Write with Opacity

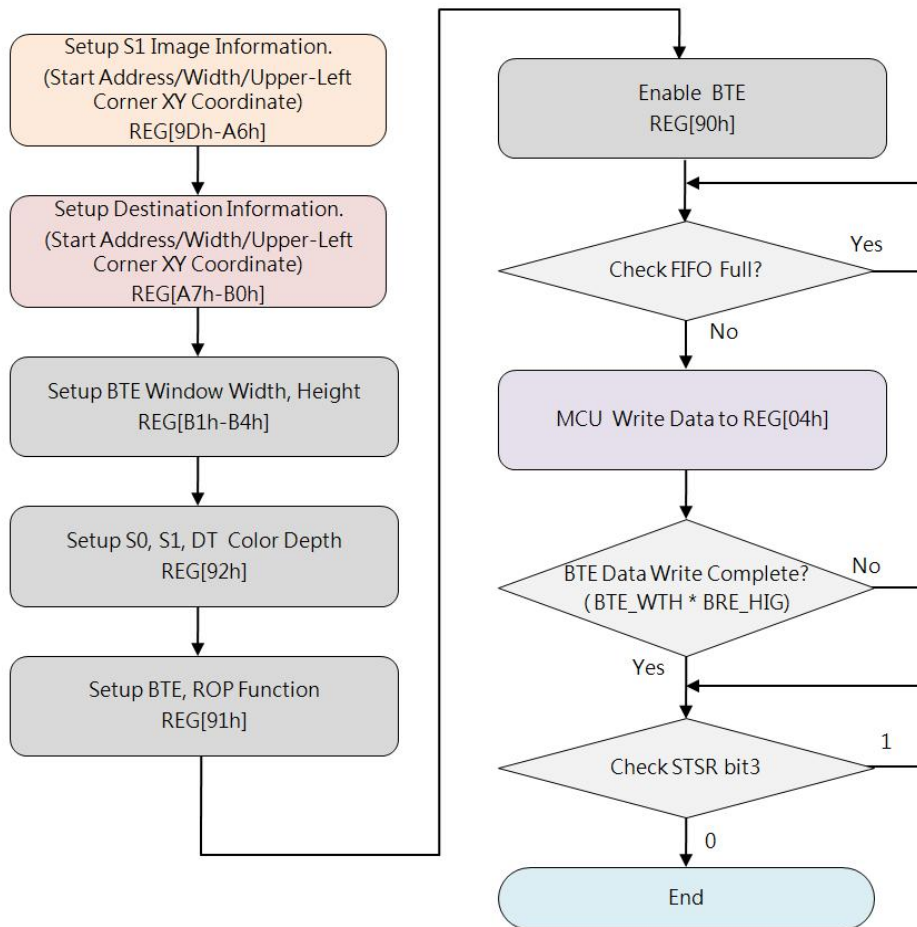


Figure 7-31: Flow Chart of MCU Write with Opacity

7.6.11 Memory Copy with Color Expansion

This Operation performs Color Expansion for the S0 data (from Memory), it 's 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]. User should set needed Start Bit to REG[91h] Bit[7:4] against 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) against the ROW Line of source image (from left to right), and sequently expands bit by bit until the BTE Width reached ending as well. 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:

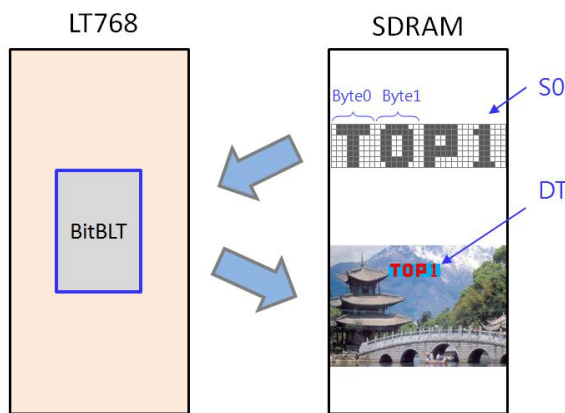


Figure 7-32: Example of Memory Copy with Color Expansion

While: S0 Color Depth = 8, Foreground Color = RED, Background Color = KHAKI, BTE window Width = 23, please refer to Figure 7-33 (ROP = 7) and Figure 7-34 (ROP = 4) as examples.

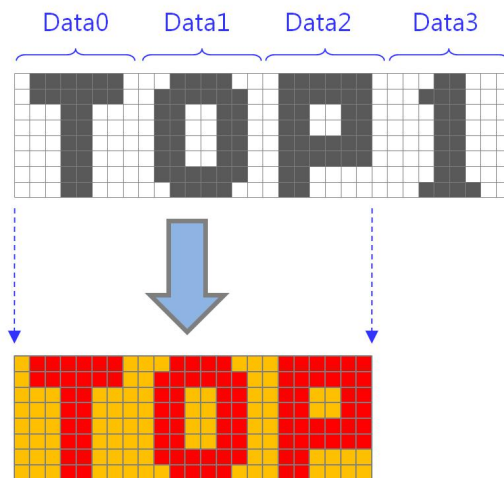


Figure 7-33: Example 1 of Memory Copy with Color Expansion (ROP=7)

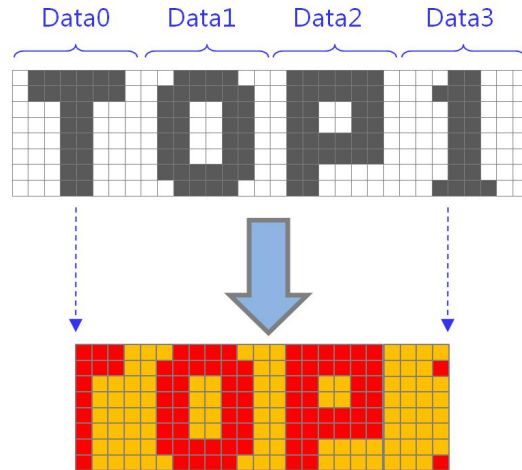


Figure 7-34: Example 2 of Memory Copy with Color Expansion (ROP=4)

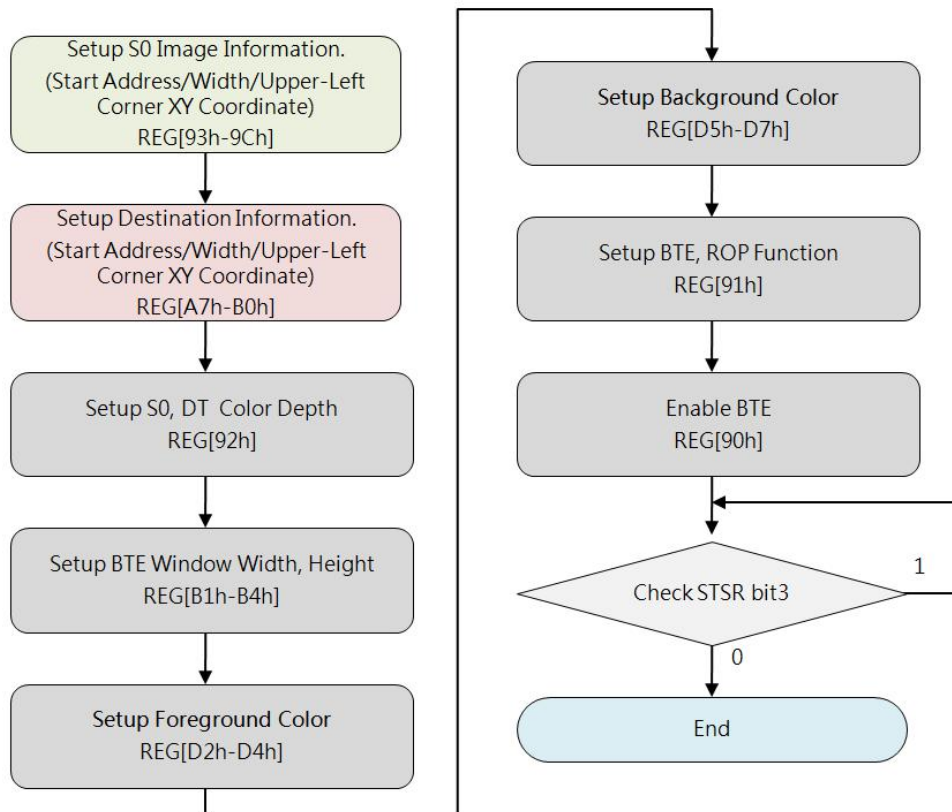


Figure 7-35: Flow Chart of Memory Copy with Color Expansion

7.6.12 Memory Copy with Color Expansion and Chroma Key

This Operation is similar with 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 expanding data_1b to RED (Foreground Color) and data_0b discarded, so the result to DT is RED "TOP" with TRANSPARENT background:

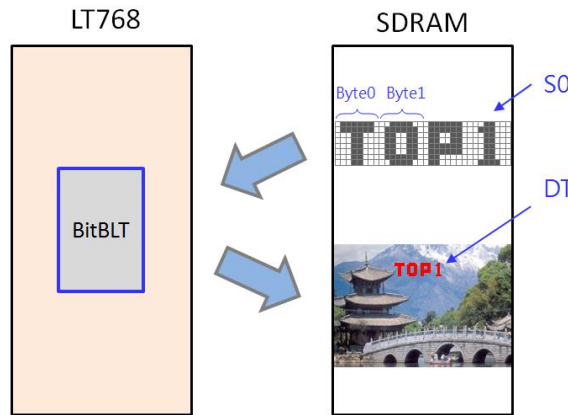


Figure 7-36: Example of Memory Copy with Color Expansion and Chroma Key

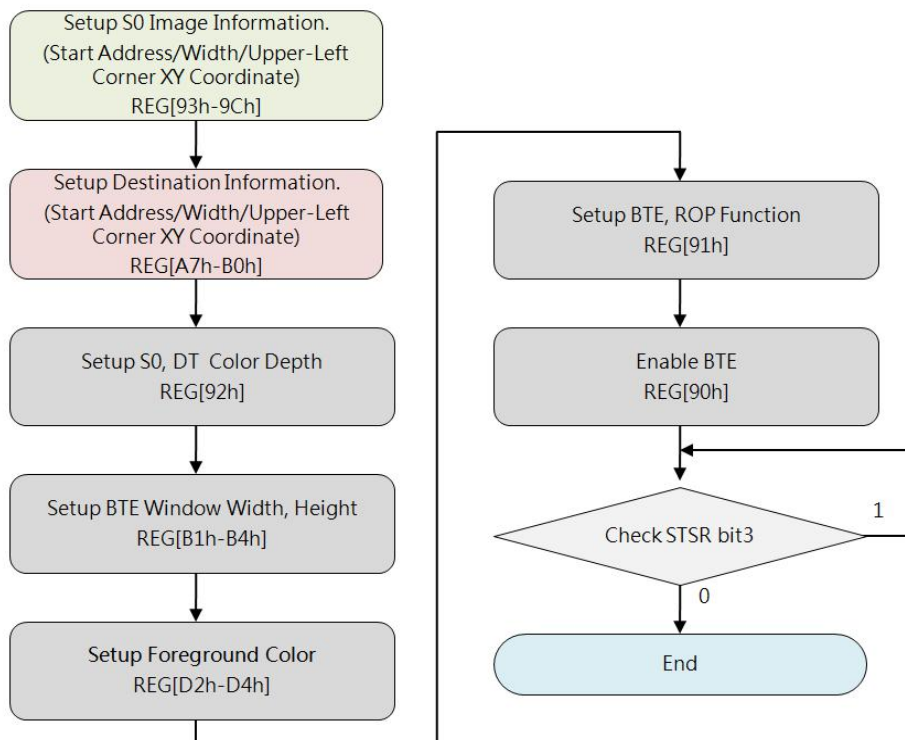


Figure 7-37: Flow Chart of Memory Copy with Color Expansion and Chroma Key

7.6.13 Solid Fill

This operation fills a specified Rectangle Area (BTE Window) of DT, with a Solid Color the data defined in the Foreground Color Register.

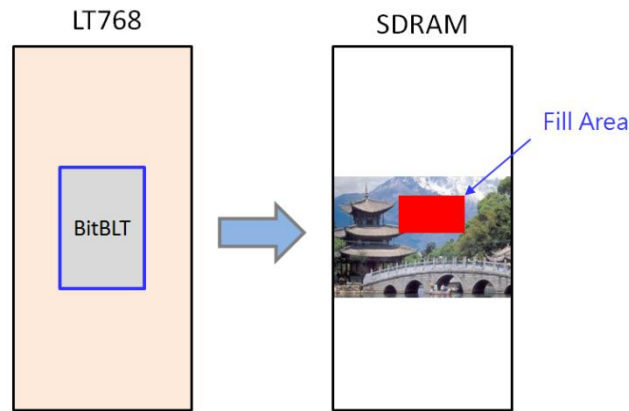


Figure 7-38: Example of Solid Fill

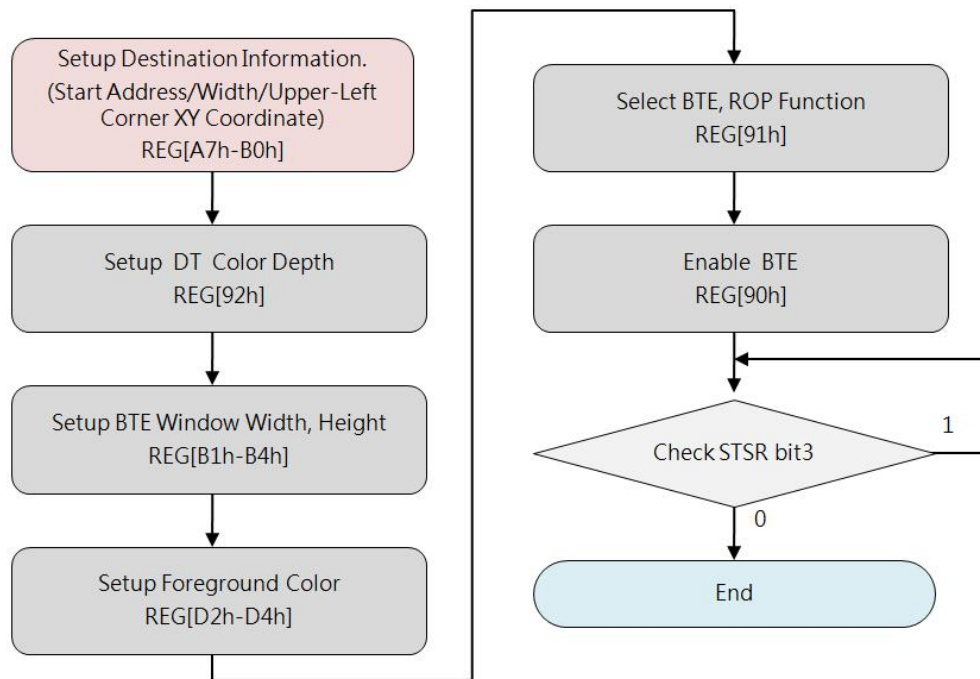


Figure 7-39: Flow Chart of Solid Fill

8. Display Text

LT7381 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)

LT7381 internal CGROM supports four sets of embedded Characters and Symbols of ASCII code, and internal CGRAM supports user to create own Characters and Symbols sets when needed. The registers REG[CCh] ~REG[DEh] are for 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.

8.1 Internal CGROM

LT7381 Internal CGROM provides user a convenient way to get and display characters by inputting ASCII code, the embedded Character sets and its Coding Standards fully meet with ISO/IEC 8859-1/2/4/5, it supports different resolution of dots matrix, including: 8*16, 12*24, 16*32. Below is basic programming flow chart:

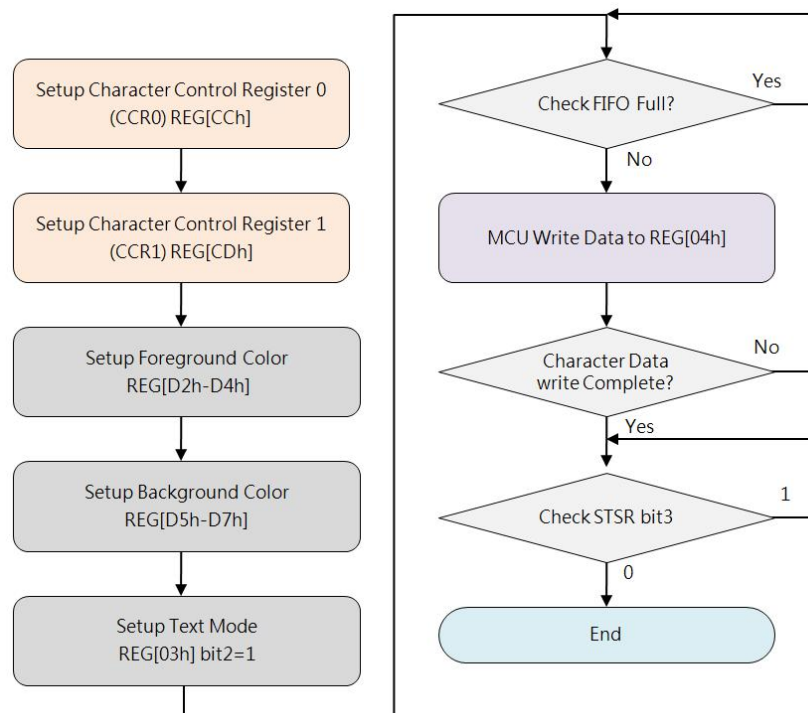


Figure 8-1: Flow Chart of CGROM Basic Programming

Table 8-1 shows the standard character encoding of ISO/IEC 8859-1. ISO means 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 consisting of 192 characters from the Latin script in range 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 also can use ISO/IEC 8859-1. In addition, it is also commonly used in many languages outside Europe, such as Swahili, Indonesian, Malaysian and Tagalong.

In the Table 8-1, character codes 0x80~0x9F are defined by Microsoft Windows, also called CP1252 (WinLatin1).

Table 8-1: ISO/IEC 8859-1

| L | H | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | ☺ | ☻ | ♥ | ♦ | ♣ | ♠ | ♣ | ♠ | + | ○ | ♀ | ♂ | ♂ | ♂ | ♂ | ♂ |
| 1 | | ▶ | ◀ | ↕ | !! | ¶ | § | ¶ | ↑ | ↓ | ↔ | → | ← | ↔ | ▲ | ▼ | |
| 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 | { | | } | ~ | |
| 8 | | € | , | f | „ | … | † | ‡ | ˆ | % | Š | < | Œ | Ž | | | |
| 9 | | • | ’ | “ | ” | • | - | - | ˜ | ™ | š | > | œ | ž | ÿ | | |
| A | | ı | ı | ı | ı | ı | ı | ı | ı | ı | ı | ı | ı | ı | ı | ı | ı |
| B | | ± | ² | ³ | µ | ¶ | · | ¸ | ¹ | º | » | ¼ | ½ | ¾ | ¿ | | |
| C | | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï |
| D | | Ð | Ñ | Ò | Ó | Ô | Õ | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ | ß |
| E | | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï |
| F | | ð | ñ | ò | ó | ô | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ | ÿ |

Table 8-2: ISO/IEC 8859-2

| L | H | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | ☺ | ☻ | ♥ | ♦ | ♣ | ♠ | ♣ | ♠ | + | ○ | ♀ | ♂ | ♂ | ♂ | ♂ | ♂ |
| 1 | | ▶ | ◀ | ↕ | !! | ¶ | § | ¶ | ↑ | ↓ | ↔ | → | ← | ↔ | ▲ | ▼ | |
| 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 | { | | } | ~ | |
| 8 | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | |
| A | | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï |
| B | | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï |
| C | | Ā | Ă | Ą | Ć | Č | Ĉ | Ď | Ě | Ĝ | Ğ | Ĥ | Ž | Ż | Ž | Ž | Ž |
| D | | Ā | Ă | Ą | Ć | Č | Ĉ | Ď | Ě | Ĝ | Ğ | Ĥ | Ž | Ż | Ž | Ž | Ž |
| E | | ā | ă | ą | ć | č | ĉ | ď | ě | ğ | ğ | ĥ | ž | ż | ž | ž | ž |
| F | | đ | ñ | ň | ó | ô | õ | ö | ÷ | ř | ú | ů | ű | ű | ý | ť | ť |

Table 8-2 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 8-3 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 8-3: ISO/IEC 8859-4

| L/H | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|-----|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ☺ | ☹ | ♥ | ♦ | ♣ | ♠ | ♣ | ♠ | ♣ | ♠ | ♣ | ♠ | ♣ | ♠ | ♣ | ♠ |
| 1 | ▶ | ◀ | ↑ | !! | ¶ | § | ¶ | ↑ | ↓ | → | ← | ↔ | ▲ | ▼ | | |
| 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 | { | | } | ~ | |
| 8 | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | |
| A | À | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï |
| B | Ā | ā | ŗ | ī | ļ | š | ē | ģ | ţ | Đ | ž | ŋ | | | | |
| C | Ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā |
| D | Ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā |
| E | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā |
| F | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā | ā |

Table 8-4: ISO/IEC 8859-5

| L/H | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|-----|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ☺ | ☹ | ♥ | ♦ | ♣ | ♠ | ♣ | ♠ | ♣ | ♠ | ♣ | ♠ | ♣ | ♠ | ♣ | ♠ |
| 1 | ▶ | ◀ | ↑ | !! | ¶ | § | ¶ | ↑ | ↓ | → | ← | ↔ | ▲ | ▼ | | |
| 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 | { | | } | ~ | |
| 8 | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | |
| A | Ё | ѐ | Г | г | Є | є | І | і | Ј | ј | Њ | њ | Ќ | ќ | Ў | ў |
| B | А | Б | В | Г | Д | Е | Ж | З | И | Й | К | Л | М | Н | О | П |
| C | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я |
| D | а | б | в | г | д | е | ж | з | и | й | к | л | м | н | о | п |
| E | р | с | т | у | ф | х | ц | ч | ш | щ | ъ | ы | ь | э | ю | я |
| F | ѐ | ђ | ѓ | є | ѕ | і | ї | ј | њ | ћ | ќ | ѕ | ў | џ | ѡ | ѣ |

Table 8-4 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.

8.2 User-defined Character Graphic (UCG)

User can create and utilize UCG when needed. LT7381 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 someone UCG, just need MCU to write corresponding CODE of the UCG to LT7381, LT7381 can resolve the CODE (relative ADDRESS of CGRAM) to corresponding absolute ADDRESS of Memory where the UCG data is really saved, then transfer the graphic data to display Memory Buffer. Of course, user can define Foreground Color by setting the REG[D2h]~REG[D4h] and Background Color by setting the REG[D5h]~REG[D7h] in advance.

To create UCG and Initialize CGRAM, need to format data of dot-matrix graphic and allocate Memory Space at first, please refer to below sections.

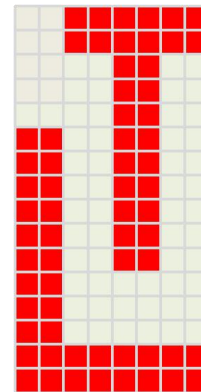
8.2.1 8*16 UCG Data Format

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 and table show the way to calculate 8*16 UCG Address in Memory, data Format and data byte Sequence:

$$\text{UCG_ADD} = \text{CGRAM_Start_ADD} + (\text{UCG_Code} * 16)$$

Table 8-5: Data Format and Byte Sequenc 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 |



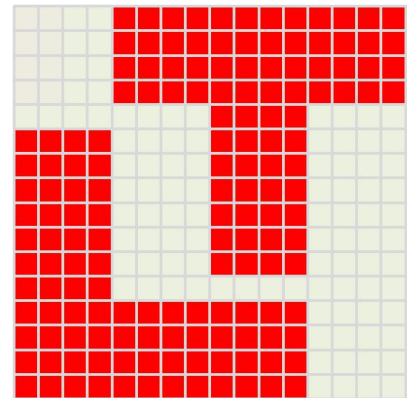
8.2.2 16*16 UCG Data Format

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 and table show the way to calculate 16*16 UCG Address in Memory, data Format and data byte Sequence:

$$UCG_ADD = CGRAM_Start_ADD + (UCG_Code * 32)$$

Table 8-6: 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 |



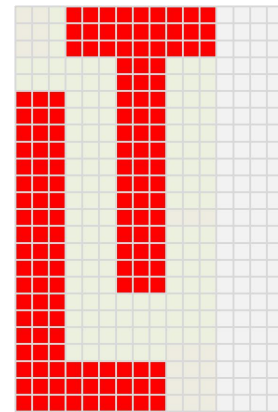
8.2.3 12*24 UCG Data Format

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 and table show the way to calculate 12*24 UCG Address in Memory, data Format and data byte Sequence:

$$\text{UCG_ADD} = \text{CGRAM_Start_ADD} + (\text{UCG_Code} * 48)$$

Table 8-7: Data Format and Byte Sequenc 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 |



8.2.4 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 and table show the way to calculate 24*24 UCG Address in Memory, data Format and data byte Sequence:

$$\text{UCG_ADD} = \text{CGRAM_Start_ADD} + (\text{UCG_Code} * 72)$$

Table 8-8: Data Format and Byte Sequenc 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 |

8.2.5 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 and table show the way to calculate 16*32 UCG Address in Memory, data Format and data byte Sequence:

$$\text{UCG_ADD} = \text{CGRAM_Start_ADD} + (\text{UCG_Code} * 64)$$

Table 8-9: Data Format and Byte Sequenc 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 |

8.2.6 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 and table show the way to calculate 32*32 UCG Address in Memory, data Format and data byte Sequence:

$$\text{UCG_ADD} = \text{CGRAM_Start_ADD} + (\text{UCG_Code} * 128)$$

Table 8-10: Data Format and Byte Sequenc 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 |

8.2.7 Initialize CGRAM from MCU

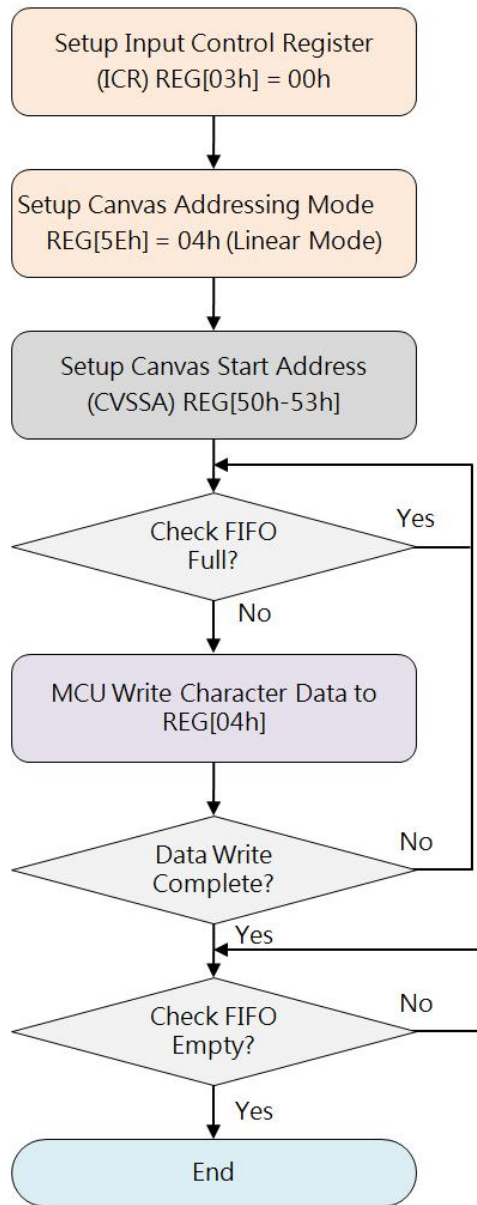


Figure 8-2: Flow Chart of CGRAM Initialization from MCU

8.2.8 Initialize CGRAM from Serial Flash

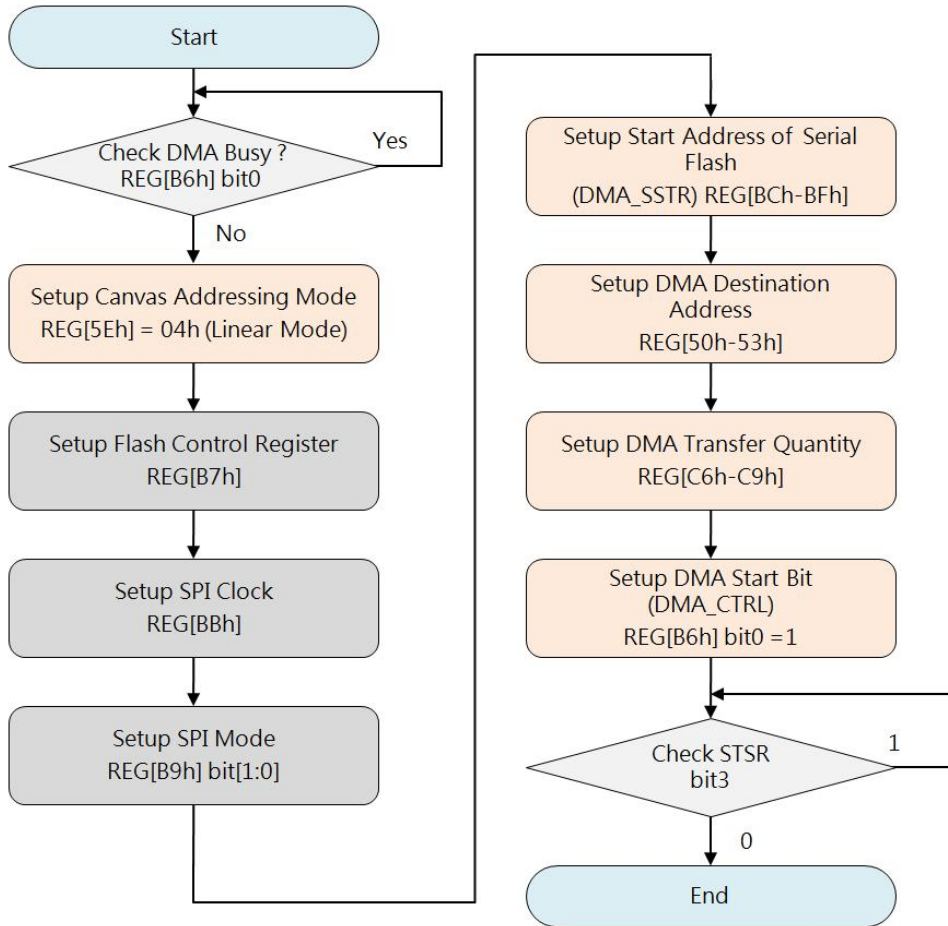


Figure 8-3: Flow Chart of CGRAM Initialization from Serial Flash

8.3 Character Rotation by 90 Degree

LT7381 supports to rotate character display 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 rotate counterclockwise 90 degree and flip in vertical, as well as text direction will change to from top to bottom then from left to right. But to see correct display result, need to further change Display Scan Direction (set VDIR REG[12h] bit3=1, but please note that Text Cursor and Graphic Cursor as well as PIP are disabled automatically under this setting). Below is an example of Character Rotation by 90 degree:

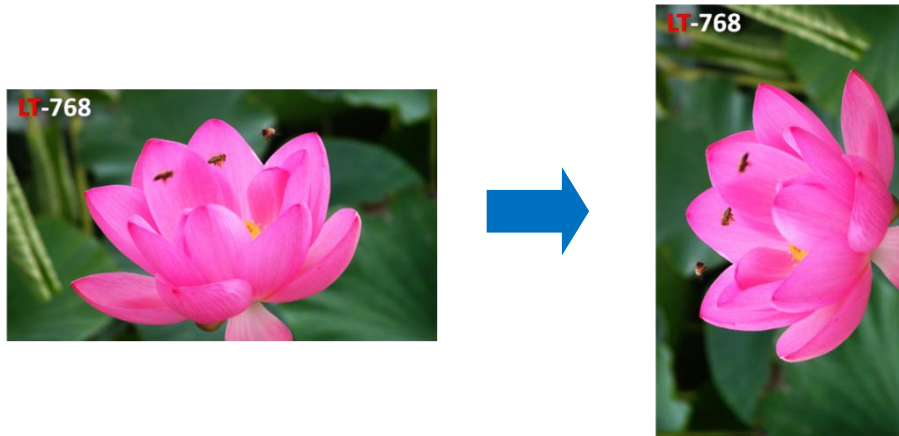


Figure 8-4: Example of Character Rotation

8.4 Size Enlargement

LT7381 supports linear *1, *2, *3, *4 character size enlargement for Height and/or Width, controlled by REG[CDh] bit[3:0].

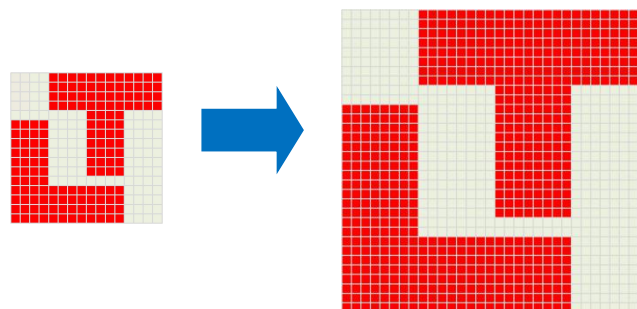


Figure 8-5: Example of Size Enlargement

8.5 Background Transparency

LT7381 supports character Background transparent, controlled by REG[CDh] bit6.

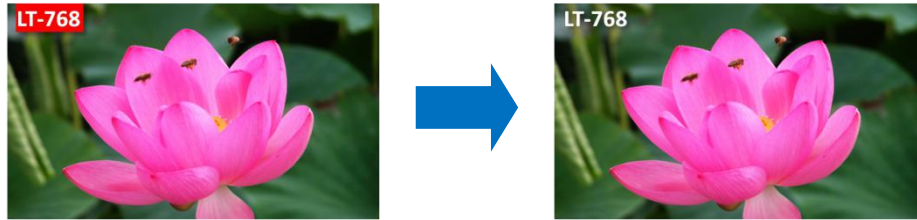


Figure 8-6: Example of Background Transparency

8.6 Character Full-Alignment

LT7381 supports character full-alignment that makes the character to align each other when input and display Half or Full size characters, set REG[CDh] bit7=1.

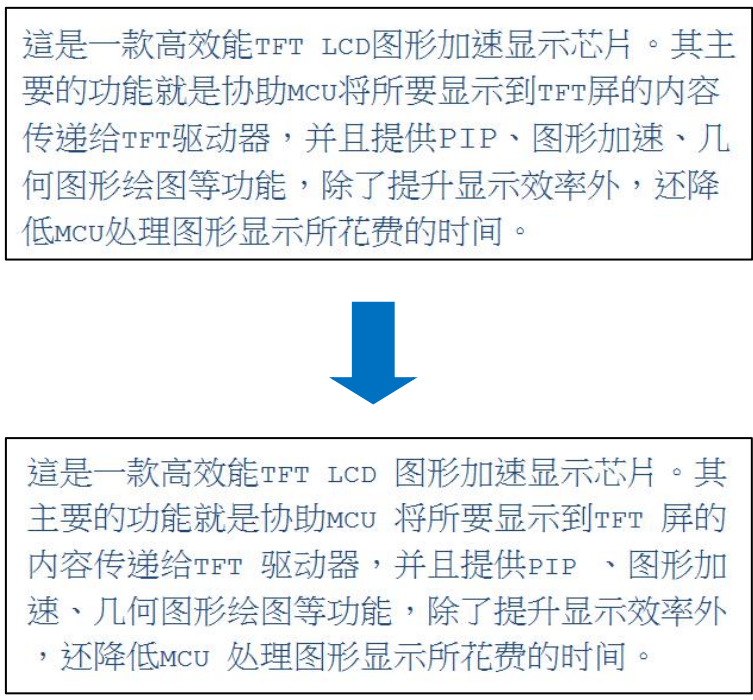


Figure 8-7: Example of Character Full-Alignment

8.7 Automatic Line Feed

LT7381 supports sequent text input and display, and can perform automatically Line Feed at active window boundary.

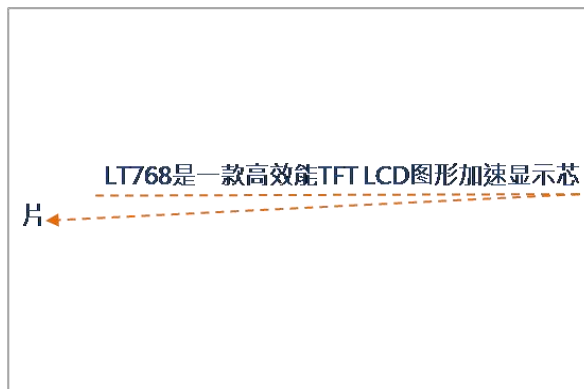


Figure 8-8: Example of Automatic Line Feed

8.8 Cursor

LT7381 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 maximum size, to point text input position. Note that when Vertical Scan Direction set to “From bottom to Top” (VDIR REG[12h] bit=1), Text Cursor and Graphic Cursor as well as PIP will be disabled automatically.

8.8.1 Text Cursor

Text Cursor has Auto-move, Blinking, and Enlargement functions, once enabled the Text Cursor appears in waiting input position, and can automatically move to next input position after current input completed. Auto-move also supports Auto Line Feed, but is dominated by Active Window, so Text Cursor must be positioned in active window and in Text Mode, moving distance and direction is same with Character input settings.

Table 8-11: Regiaters 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 X coordinate |
| REG[D0h] | FLDR | Text Line Gap Setting |

■ **Text Cursor Blinking**

By GTCCR (REG[3Ch]) to set Blinking enabled (bit[0]=1) or disabled (bit[0]=0), use below formula to calculate blinking interval:

■ **Blink_Time (sec) = BTCR[3Dh] * (1/Frame_Rate)**



Figure 8-9: 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 same enlargement settings.

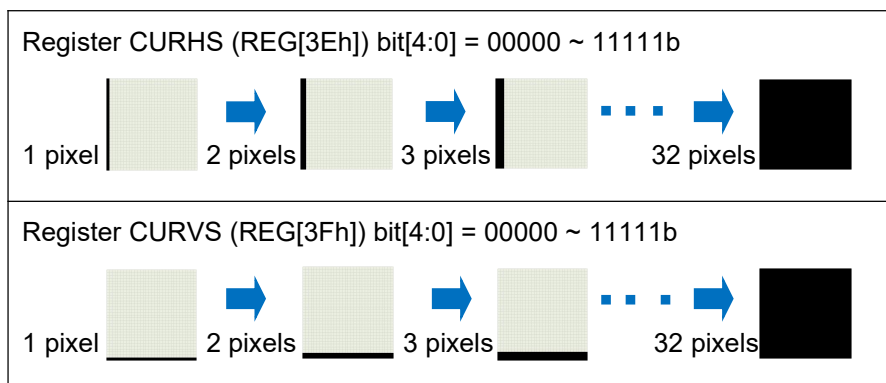


Figure 8-10: Example of Text Cursor Height and Width Settings

8.8.2 Graphic Cursor

LT7381 Graphic Cursor is 32*32 pixel graphic with 2-bits Color Index, display color data is redirected to register GCC0(REG[44h]), GCC1(REG[45h]), Background Color, Inversed Background color:

Table 8-12: Graphic Color Definition

| | |
|--------------|--------------------------------|
| 2'b00 | Color-0 (defined by REG[44h]) |
| 2'b01 | Color-1 (defined by REG[45h]) |
| 2'b10 | Background Color |
| 2'b11 | Inversed Background Color |

To create a Graphic Cursor needs 256 bytes (32x32x2/8), Figure 8-11 shows the data format and byte sequence. LT7381 supports 4 sets of graphic cursor for selection (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]).

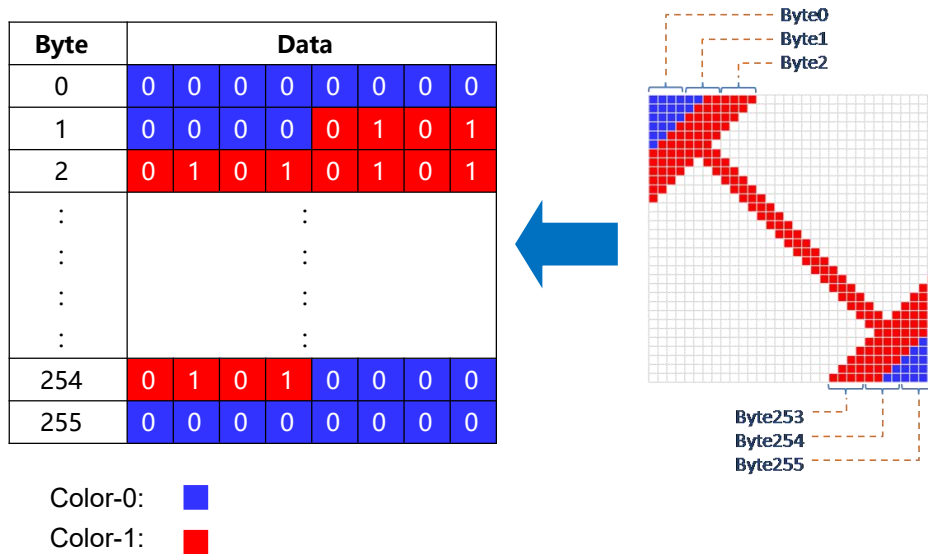


Figure 8-11: Graphic Cursor Data Format and Example

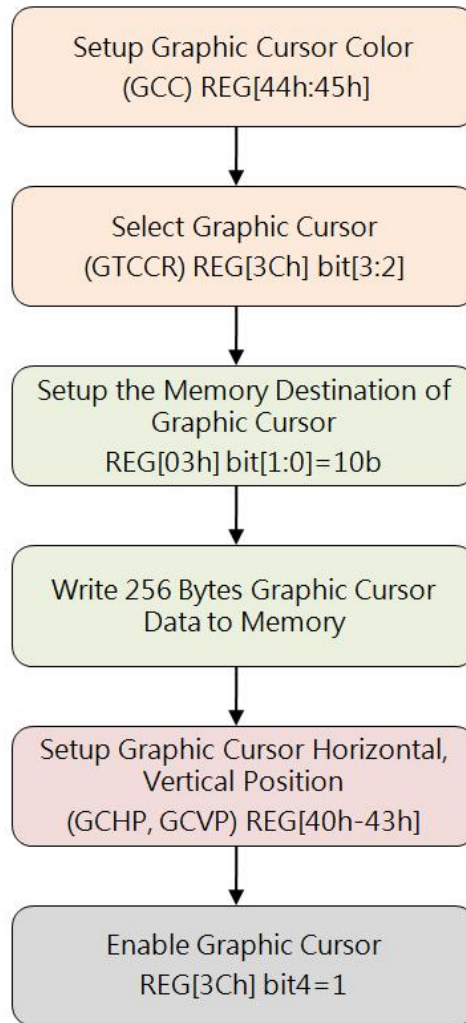


Figure 8-12: Flow Chart of Graphic Cursor Creation

9. Pulse Width Modulation (PWM)

LT7381 built-in PWM function, and provide two PWM signals output: PWM0 and PWM1. The LT7381 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 use 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 active. That's mean 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 to 0 then an interrupt will generated. The PWM0's state will back to the original 0, and also automatically reload TCNTB0 value. The above procedure is represent a complete PWM cycle.

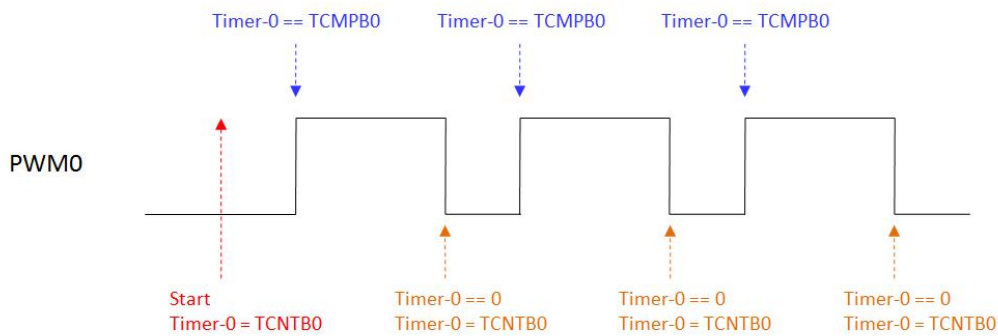


Figure 9-1: PWM0 Output Wave Form

Base on the above waveform and description, actually the duty cycle of PWM0 is determined by comparison registers (TCMPB0, REG[88h-89h]). For example, you want to generate a voltage of approximate DC potential by PWM0. When the PWM0 is initially set to 0, if TCMPB0 value is set to be large then we can get higher equivalent voltage on PWM0. Conversely, when the PWM0 is initially set to 1, then TCMPB0 value has to set smaller for getting higher equivalent voltage on PWM0.

Note: The automatic reload feature of PWM0 (REG[86h] bit1) must be turned on, 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.

9.1 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]):

$$PWM_CLK = CCLK / (Prescaler + 1)$$

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 next chapter for REG[84H] and REG[85h].

9.2 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 overload feature (REG[86h] bit1 = 0), and stop Timer-0 count (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 shared output pins that can be used for other purposes, please refer to the following description of Register REG[85h] bit[3:0].

Table 9-1: REG[85h] Description

| Bit | Description |
|-----|--|
| 3-2 | PWM[1] Function Control 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 | PWM[0] Function Control 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 is followed by the setting and control of PWM0, except that it is a PWM0 reverse output state:

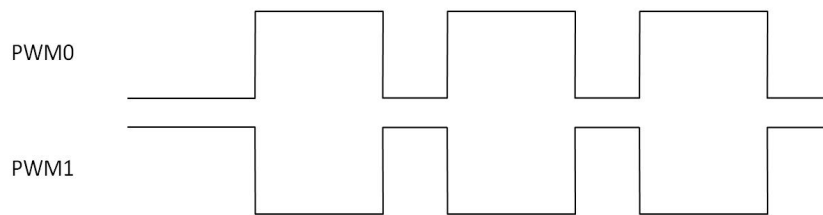


Figure 9-2: The Complementary Outputs of PWM0 and PWM1

In some applications of using Complementary Outputs of PWM0 and PWM1, PWM0 and PWM1 change state at the same time will cause excessive current. The LT7381 provides a dead-zone timing control to stagger the output state of PWM0 and PWM1 transfer at same time. The dead-zone length of PWM is set by register REG[87h] (DZ_LENGTH):

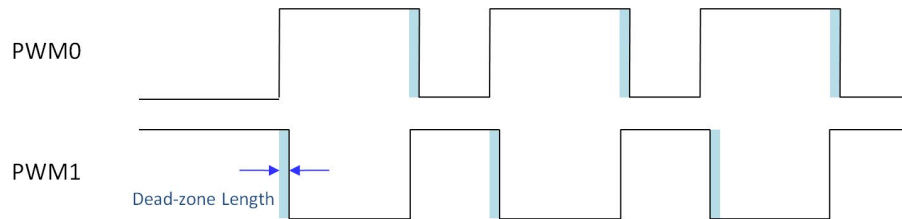


Figure 9-3: dead-zone of PWM0 and PWM1

10. I2C Master

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

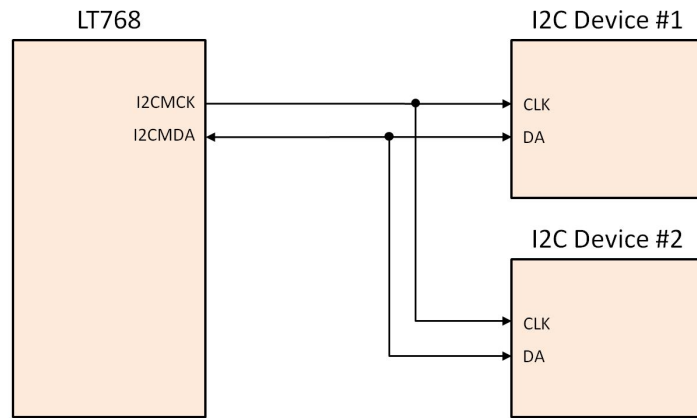


Figure 10-1: Application Circuit of LT7381 I2C Master

LT7381 support 100Kbps and 400Kbps transfer rate for I2C bus. The formula of I2C Master clock(I2CMCK) is as the following:

$$I2CMCK = CCLK / [5 * (Prescaler + 2)]$$

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 synchronously 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 processing during the high period of I2CMCK so that the I2CMDA could be change 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

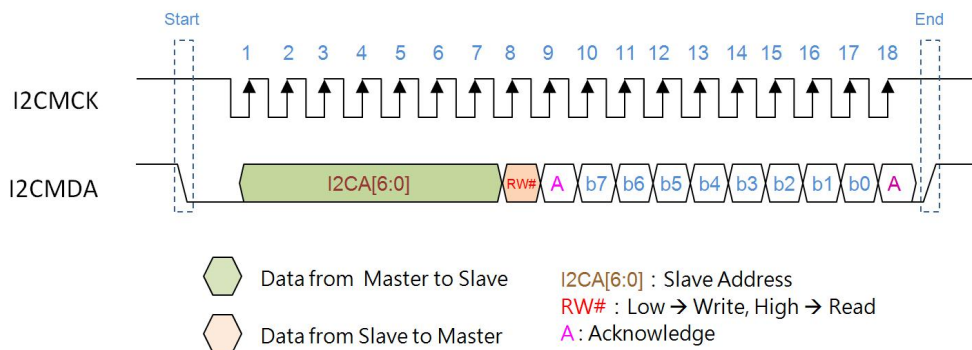


Figure 10-2: I2C Communication Protocol

Example 1. Write 1 Byte data to Slave Device:

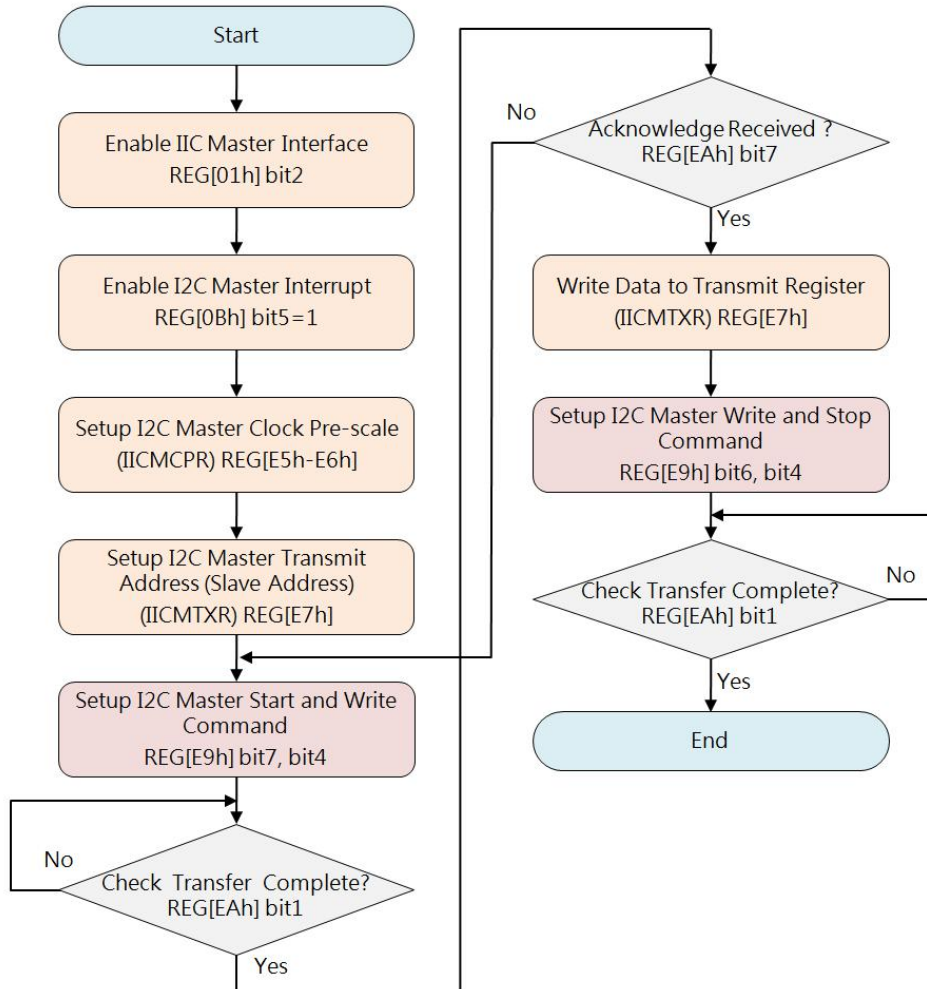


Figure 10-3: Write Data to Slave

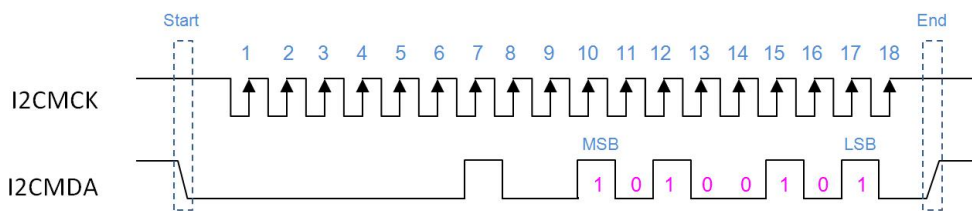


Figure 10-4: Write Data "A5" to Slave (Address=0x01)

Example 2. Read 1 Byte Data from Slave Device:.

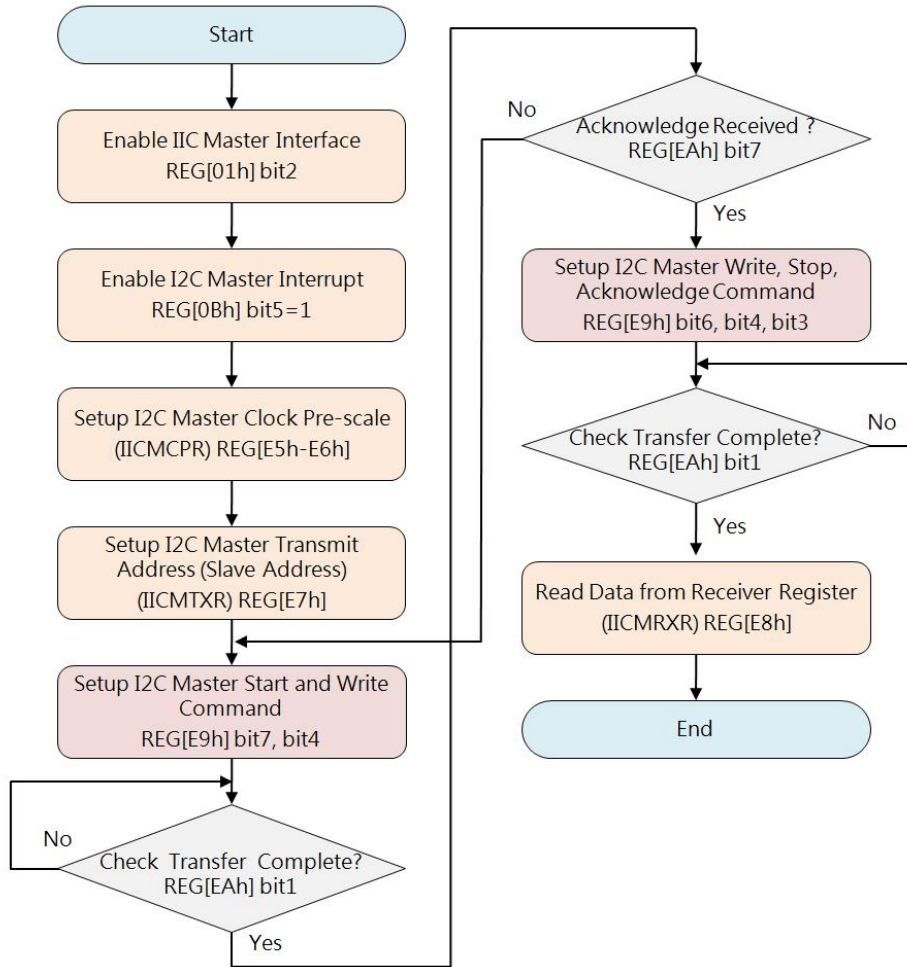


Figure 10-5: Read Data from Slave

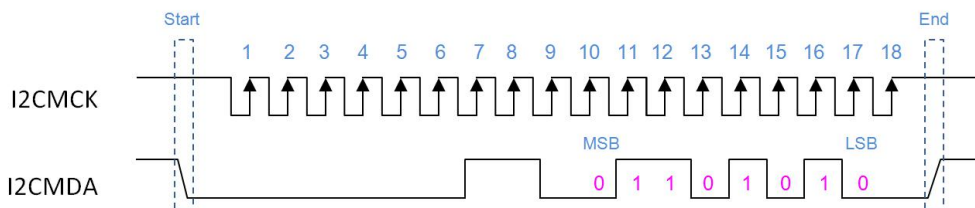


Figure 10-6: Read Data "6A" from Slave (Address=0x01)

Note: The I2CMCK and I2CMDA are multiplex pins that share with KI[0] and KO[0].

11. Keypad-Scan

LT7381 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. As long as the setting of several registers, Host can make MCU easy to get the keyboard data. The following diagram is the basic keyboard application circuit.

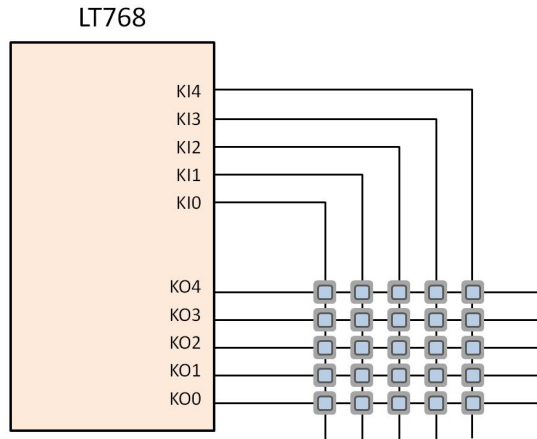


Figure 11-1: Application Circuit of Keypad-scan

11.1 Keypad-scan Operation

The embedded Keypad-scan features of LT7381 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 pressed. The Register KSCR2 bit[1:0] record the number of keystrokes pressed, then Host can get the corresponding key-code of pressed keys by reading register KSDR.

Table 11-1: 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 11-1 and Table 11-2 are 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 user press the Keypad button for a long time, then LT7381 will corresponding different key-code which saved in Registers (KSDR0 ~ KSDR2).

Table 11-2: Key-code of Long-press Key

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

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

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

KSDR0 = 0x00

KSDR1 = 0x22

KSDR2 = 0x34

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

Table 11-3: 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 check the Keypad-scan Status Register
- Host check the Interrupt Signal

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

In addition, LT7381 supports "key-press wakeup" in power-saving mode. After the register are setup complete, any keystroke triggers can awaken the LT7381 from sleep mode. In order to recognize the Wake-up event, Host can use software to polling LT7381 interrupt status bit. As follows Figure 11-2 flowchart shown.

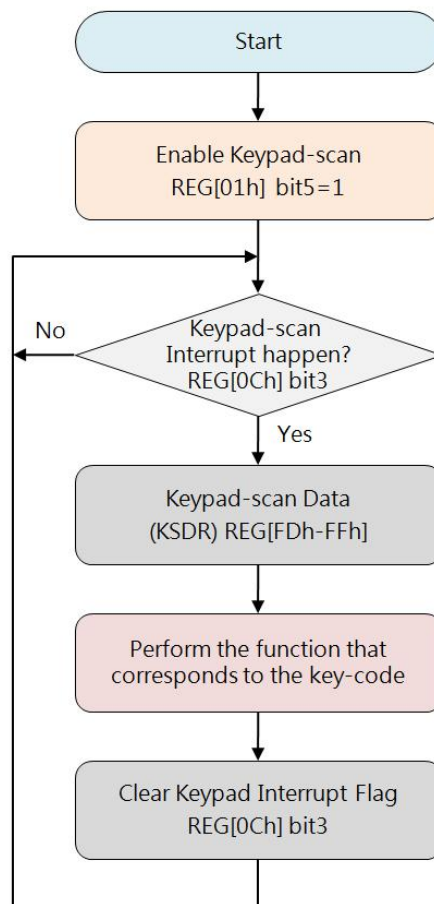


Figure 11-2: Flowchart to check Key-pad Interrupt (S/W Polling Mode)

The Host can also be notified by hardware interrupt signal INT#, as shown in the Figure 11-3 flowchart.

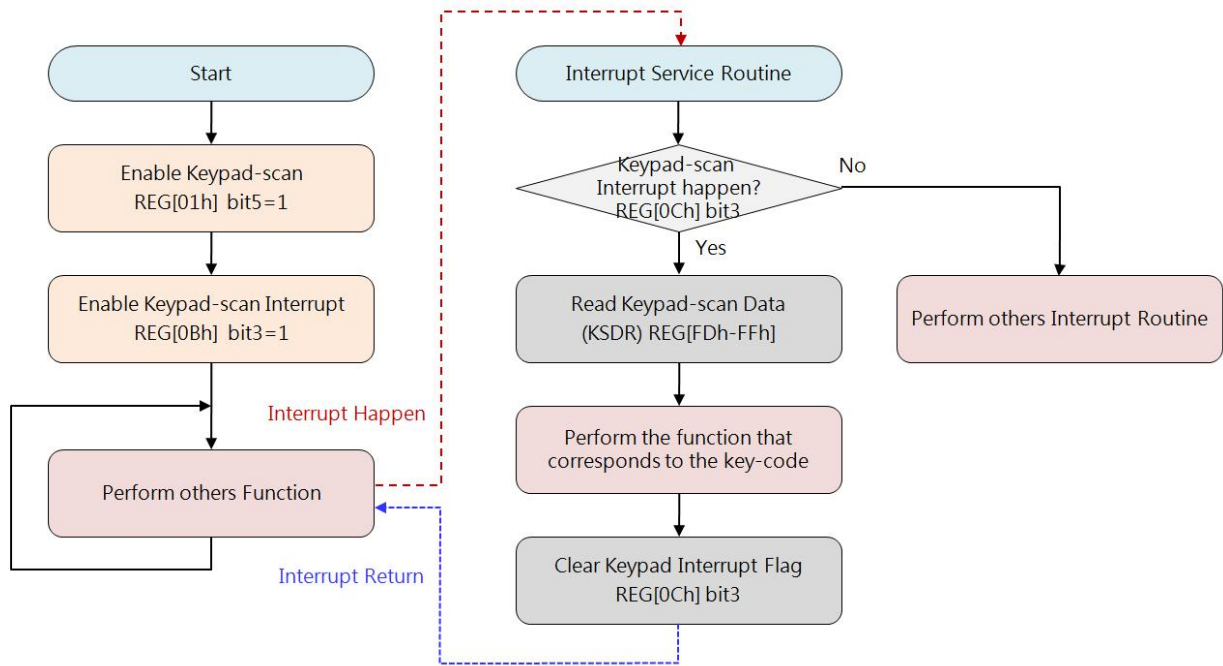


Figure 11-3: Flowchart to check Key-pad Interrupt (H/W Interrupt Mode)

12. GPIO Interface

LT7381 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 12-1 as below. And note these GPIO signals are available only when their mapping control signals were disabled.

Table 12-1: 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] |
| IOD[7:0] | PD[18, 2, 17, 16, 9, 8, 1, 0] |

Table 12-2 is a supported GPIO signals list of LT7381. These GPIO signals are set to output or input, as well as to output data or read input data, are controlled by REG[F0h-F6h]. Please refer to Chapter 14 - Register Description.

Table 12-2: The Supported GPIO Port for LT7381

| GPIO Number | GPIO Port |
|-------------|---|
| 23 | GPIOA[7:0] GPIB[4], GPOB[4], GPIB[3:0] GPIOC[7], GPIOD[7:0] |

13. Power Management

LT7381 has a total of four power modes of operation, based on the power consumption from high to Low: Normal mode, Standby mode, Suspend mode and sleep mode. These four working modes are set by register REG[DFh]. The following are four working modes of clock action comparison tables:

Table 13-1: 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 LT7381 enters the power saving mode, the LCD interface will not output the signal. Therefore, before entering the power-saving mode, Host need to do the LCD module display off or power off to avoid LCD polarization damage.
- The "OSC" means external X'tal oscillator.

13.1 Normal Mode

In this mode, three of the internal PLL functions are working normally. That is, Host setup Registers to control three PLL respectively produce CCLK (Core Clock), MCLK (Display RAM Clock) and PCLK (LCD scan Clock). Because the PLL need some time to work stable, therefore, Host must first through the register 01h bit7 to know whether the PLL frequency is in a stable state.

13.2 Standby Mode

If setup REG[DFh] bit[1:0] to 01b, LT7381 will enter Standby mode. The System Clock (CCLK) and LCD-Scan Clock (PCLK) will stop. The Display RAM Clock is active and provides 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 this bit become to 1 to make sure LT7381 enter Standby mode.

Back to Normal Mode:

- ◆ Setup REG[DFh] bit7 to 0 (Quit from Standby Mode)
- ◆ The Host may check the bit1 of Status Register (STSR), and wait this bit become to 0 to make sure LT7381 back to Normal Mode.

13.3 Suspend Mode

If setup REG[DFh] bit[1:0] to 10b, LT7381 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 provides by OSC Clock..

Enter Suspend Mode:

- ◆ According to OSC frequency to setup appropriate Display RAM(SDRAM) Refresh Clock
- ◆ Select Suspend Mode(REG[DFh] bit[1:0] = 10b)
- ◆ Setup REG[DFh] bit7 to 1 (Enter Suspend Mode)
- ◆ The Host may check the bit1 of Status Register (STSR), and wait this bit become to 1 to make sure LT7381 enter Suspend Mode.

Back to Normal Mode:

- ◆ Setup REG[DFh] bit7 to 0 (Quit from Suspend Mode)
- ◆ The Host may check the bit1 of Status Register (STSR), and wait this bit become to 0 to make sure LT7381 back to Normal Mode.

13.4 Sleep Mode

If setup REG[DFh] bit[1:0] to 11b, LT7381 will enter Sleep Mode. And all of Clocks and PLL will be stop operate.

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 that before LT7381 enter Sleep mode. If REG[E0h] bit7 is 1, the Display RAM will enter Refresh mode.
- ◆ If Host interface is Parallel Mode, then LT7381 will stop OSC. If Host interface is Serial Mode, then OSC will not be stop.
- ◆ The Host may check the bit1 of Status Register (STSR), and wait this bit become to 1 to make sure LT7381 enter Sleep Mode.

Back to Normal Mode:

- ◆ Setup REG[DFh] bit7 to 0 (Quit from Sleep Mode)
- ◆ If OSC was be stopped in Sleep mode, the Host have to enable OSC.
- ◆ The Host may check the bit1 of Status Register (STSR), and wait this bit become to 0 to make sure LT7381 back to Normal Mode.

14. Register Description

LT7381 provides a compatible 4 forms of Host interface, and the internal registers are read and written through these Host interface cycles to complete. LT7381 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. "Command Write" specifies the address of the register, and then the "Data Write" period can be written to the specified register. And when the 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 the set register address, "Data Read" is to read register data.

Table 14-1: 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 LT7381 are describe as below. Each register contains 8bits data. In the register table include the detail description, default value and access attribute (RO: Read Only, WO: Write Only, RW: Readable and Writeable).

14.1 Status Register

Table 14-2: Status Register (STSR)

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 5 | <p>Memory Read FIFO Full Indicate</p> <p>0: The FIFO of Memory Read not Full 1: The FIFO of Memory Read was Full</p> <p>When the Memory Read FIFO was Full, Host can continuous to read 8bpp data with 64 pixels, 16bpp data with 32 pixels or 24bpp data with 16 pixels.</p> | 0 | RO |
| 4 | <p>Memory Read FIFO Empty Indicate</p> <p>0: The FIFO of Memory Read not Empty 1: The FIFO of Memory Read was Empty</p> <p>When the Memory Read FIFO not Empty, Host can continuous read pixel data from Memory.</p> | 1 | RO |
| 3 | <p>Core Task is Busy, Fontwr_Busy</p> <p>This bit represents whether the action is completed for LT7381 BTE, Geometry engine, DMA, text writing, or graphic writing.</p> <p>0: The action is complete or idle. 1: The action is not completed, in a busy state.</p> <p>When the Host program switches text and graphics mode, or changes the base diagram related settings, the program must first verify that the LT7381 is idle.</p> <p>Note: In text mode, if program changes text rotation, line spacing, character spacing, foreground color, background color, and text/graphics settings, Host must confirm this bit is 0.</p> | 0 | RO |
| 2 | <p>Display RAM Ready for Access</p> <p>0: Display RAM is not ready for access 1: Display RAM is ready for access</p> <p>Before check this bit, Host has to setup REG[E4h] bit0 "SDR_INIT" as 1.</p> | 0 | RO |
| 1 | <p>Operation Mode Status</p> <p>0: Normal operation state 1: Inhibit operation state</p> <p>Inhibit operation state means internal reset event keep running or initial display still running or chip enter power saving state. In power saving state, this bit becomes 1 until PLL clock stop.</p> | 0 | RO |
| 0 | <p>Interrupt Pin State</p> <p>0: without interrupt active 1: interrupt active</p> | 0 | RO |

14.2 Configuration Registers

REG[00h] Software Reset Register (SRR)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7 | Reconfigure PLL frequency Write "1" to this bit will reconfigure PLL frequency. Note: 1. When user change PLL relative parameters, PLL clock won't change immediately, user must set this bit as "1" again. 2. User may read (check) this bit to know whether system already switch to PLL clock or not yet. | 0 | RW |
| 6-1 | Reserved | 0 | RO |
| 0 | Software Reset (Reconfigure PLL Frequency) 0: Normal Operation. 1: Software Reset. The bit will auto clear after reset. Software Reset only reset internal state machine. Configuration Registers value won't be reset. So all read-only flag in the register will return to its initial value. User should have proper action to make sure flag is in desired state. | 0 | WO |
| 0 | Warning Condition Flag 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 | Check PLL Ready Host may read (check) this bit to know whether system already switch to PLL clock or not yet. Read "1" means PLL clock ready and switch successfully. | 0 | RW |
| 6 | Mask WAIT# on CS# De-assert 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# keep low, Host program should poll WAIT# and wait it goes high then start next access. 1: Mask, WAIT# de-assert when CS# de-assert. Use in Host cycle can be extended by WAIT# automatically. | 1 | RW |
| 5 | Keypad-scan Enable/Disable 0: Disable 1: Enable | 0 | RW |

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 4-3 | TFT Panel I/F Setting 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 | I2C Master Interface Enable/Disable 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 enable simultaneously then KI[0] and KO[0] will become I2C function & other KI/KO pins still keep Keypad-scan function. | 0 | RW |
| 1 | Serial Flash or SPI Interface Enable/Disable 0: Disable (GPIO Function) 1: Enable (SPI Master Function) | 0 | RW |
| 0 | Host Data Bus Width Selection 0: 8bits Data Bus 1: 16bits Data Bus If Serial Host I/F selected or in initial display operation period, LT7381 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 | Host Read/Write Image Data Format 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 | Host Read Memory Direction (Only for Graphic Mode) 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 | 0 | RO |
| 2-1 | Host Write Memory Direction (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°) 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 | Interrupt Pin Active Level 0: Low Active 1: High Active | 0 | RW |
| 6 | External Interrupt Signal - PSM[0] Pin De-bounce 0: Without De-bounce 1: Enable De-bounce (1,024 OSC Clock) | 0 | RW |
| 5-4 | External Interrupt Signal - PSM[0] Pin Trigger Type 00b: low level trigger 01b: falling edge trigger 10b: high level trigger 11b: rising edge trigger | 00b | RW |
| 3 | NA | 0 | RW |
| 2 | Text Mode Enable 0: Graphic mode 1: Text mode Before toggle this bit user must make sure core task busy bit in status register is done or idle. This bit always 0 (in graphic mode) if canvas' address mode is linear mode. | 0 | RW |
| 1-0 | Memory Port Read/Write Destination Selection 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. User need specify desired gamma table and continuous write 256 bytes. 10b: Graphic Cursor RAM (only accept low 8-bits MPU data, similar normal register data r/w.), not support Graphic Cursor RAM read. It contains 4 graphic cursor sets. Each set has 128x16 bits. User need specify target graphic cursor set and continue write 256 bytes. 11b: Color palette RAM. It is 64x12 bits SRAM, so even address' data only low 4 bits are valid. Not support Color palette RAM read. User need continue write 128 bytes. | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | <p>Write Function: Memory Write Data</p> <p>Data to write in memory corresponding to the setting of REG[03h][1:0]. Continuous data write cycle can be accepted in bulk data write case.</p> <p>Note:</p> <ul style="list-style-type: none"> A. Image data in Display RAM: according MPU I/F bit width setting (8/16-bits), Host R/W image data format, canvas color depth and set canvas in block mode. B. Pattern data for BTE operation in Display RAM: according MPU I/F bit width setting (8/16-bits), Host R/W image data format, canvas color depth and set canvas in block mode. Active window's width and height should set as 8x8 or 16x16 depend on user required. C. User-characters in Display RAM: according MPU I/F bit width setting (8/16-bits), Host R/W image data format and set canvas in linear mode. D. Character code: only accept low 8-bits MPU data, similar to normal register R/W. For two bytes character code, input high byte first. To user defined Character, code < 8000h is half size, code >= 8000h is full size. E. Gamma table data: only accept low 8-bits MPU data. User must set "Select Gamma table sets([3Ch] Bit6-5)" to clear internal Gamma table's address counter then start to write data. User should program 256 bytes data to memory data port. F. Graphic Cursor RAM data: only accept low 8-bits MPU data. User must set "Select Graphic Cursor sets" bits to clear internal Graphic Cursor RAM address counter then start to write data. G. Color palette RAM data: only accept low 8-bits MPU data. User must program full Color palette RAM in a continuous 128 byte data write to memory data port and cannot change register address. <p>Read Function: Memory Read Data</p> <p>Data to read from memory corresponding to the setting of REG[03h][1:0]. Continuous data read cycle can be accepted in bulk data read case.</p> <p>Note1: if you set this port address from different port address, must issue a dummy read, the first data read cycle is dummy read and data should be ignored. Graphic Cursor RAM & Color palette RAM data are not support data read function.</p> <p>Note2: read memory data is 4 bytes alignment no matter color depth setting.</p> <p>Note3: If user write data to Display RAM user must make sure write FIFO is empty before he change register number or core task busy status bit becomes idle.</p> | -- | RW |

14.3 PLL Setting Register

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-6 | PCLK Output Divider Ratio, OD[1:0] 00b: Divided by 1. 01b: Divided by 2. 10b: Divided by 3. 11b: Divided by 4. | 0 | RW |
| 5-1 | PCLK Input Divider Ratio, R[4:0] The value should be 2~31. | 10 | RW |
| 0 | PCLK Feedback Divider Ratio of Loop, N[8] 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 | PCLK PLLDIVN[7:0] 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 | MCLK output divider Ratio, OD[1:0] 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 | MCLK PLLDIVN[7:0] 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 (CPLL1)

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

REG[0Ah] CCLK PLL Control Register 2 (CPLL2)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | CCLK PLLDIVN[7:0] 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, $IN = 10\text{MHz}$, $R[4:0] = 01010$, $N[8:0] = 100000000$, $OD[1:0] = 11$, then

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

14.4 Interrupt Control Register

REG[0Bh] Interrupt Enable Register (INTEN)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7 | Wakeup/Resume Interrupt Enable 0: Disable 1: Enable | 0 | RW |
| 6 | External Interrupt Input - PSM[0] Enable) 0: Disable 1: Enable | 0 | RW |
| 5 | I2C Master Interrupt Enable 0: Disable 1: Enable | 0 | RW |
| 4 | VSYNC Time Base Interrupt Enable 0: Disable Interrupt 1: Enable Interrupt This interrupt event may provide the host processor with Vsync signal information for tearing effect. | 0 | RW |
| 3 | Keypad-scan Interrupt Enable 0: Disable Interrupt 1: Enable Interrupt | 0 | RW |
| 2 | Serial Flash DMA Complete / Draw Task Finished / BTE Process Complete etc. Interrupt Enable 0: Disable Interrupt 1: Enable Interrupt | 0 | RW |
| 1 | PWM Timer-1 Interrupt Enable 0: Disable Interrupt 1: Enable Interrupt | 0 | RW |
| 0 | PWM Timer-0 Interrupt Enable 0: Disable Interrupt 1: Enable Interrupt | 0 | RW |

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

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

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

Note: If the Host received interruption, but the flag of this register show not interrupted happen, then Host should be to confirm SPI Master State Register Interrupt Flag of REG[BAh].

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

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

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

Note: If the Host masking interrupt flag, then the ILT7381 will not send interruption to Host. If only use some not to be masked interrupt flag, Host can check interrupt flag to know whether there is interruption.

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

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

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

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

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

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

14.5 LCD Display Control Registers

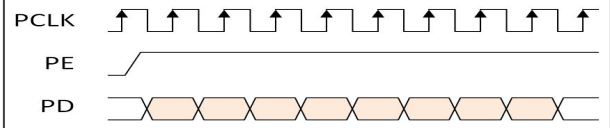
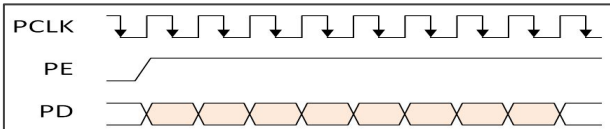
REG[10h] Main/PIP Window Control Register (MPWCTR)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7 | PIP-1 Window Enable/Disable 0: PIP-1 Window Disable 1: PIP-1 Window Enable PIP-1 window always on top of PIP-2 window. | 0 | RW |
| 6 | PIP-2 Window Enable/Disable 0: PIP-2 Window Disable 1: PIP-2 Window Enable PIP-1 window always on top of PIP-2 window | 0 | RW |
| 5 | NA | 0 | RO |
| 4 | Select Configure PIP 1 or 2 Window's parameters 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 | Main Image Color Depth Setting 00b: 8bpp Generic TFT (256 color) 01b: 16bpp Generic TFT (65K color) 1xb: 24bpp Generic TFT (1.67M color) | 1 | RW |
| 1 | NA | 0 | RW |
| 0 | To Control Panel's Synchronous Signals 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 | PIP-1 Window Color Depth Setting 00b: 8bpp Generic TFT (256 color) 01b: 16bpp Generic TFT (65K color) 1xb: 24bpp Generic TFT (1.67M color) | 1 | RW |
| 1-0 | PIP-2 Window Color Depth Setting 00b: 8bpp Generic TFT (256 color) 01b: 16bpp Generic TFT (65K color) 1xb: 24bpp Generic TFT (1.67M color) | 1 | RW |

REG[12h] Display Configuration Register (DPCR)

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

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

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7 | HSYNC Polarity 0: Low Active 1: High Active | 0 | RW |
| 6 | VSYNC Polarity 0: Low Active 1: High Active | 0 | RW |
| 5 | PDE Polarity 0: High Active 1: Low Active | 0 | RW |
| 4 | PDE Idle State 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 | PCLK Idle State 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 | PD Idle State 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 | HSYNC Idle State 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 | VSYNC Idle State 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 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 4-1 TFT-LCD interface Timing diagram.

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | <p>Horizontal Display Width Setting</p> <p>This register is set to a horizontal display width. Its specified LCD screen resolution is 8 pixels in one unit resolution.</p> <p style="text-align: center;">Horizontal Display Width (pixels) $= (\text{HDWR} + 1) * 8 + \text{HDFTR}$</p> <p>HDFTR (REG[15h]) is the fine-tuning value for Horizontal Display Width. Each fine-tuning resolution is 1 pixels, and the maximum horizontal width is 2,048 pixels.</p> | 4Fh | RW |

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

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

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-5 | NA | 0 | RO |
| 4-0 | <p>Horizontal Non-Display Period</p> <p>This register assign the Period of Horizontal Non-Display. It's also called "Back Porch".</p> <p style="text-align: center;">Horizontal Non-Display Period (Pixels) $= (\text{HNDR} + 1) * 8 + \text{HDFTR}$</p> <p>HDFTR (REG[17h]) is the fine-tuning value for Horizontal Non-display Period. Each fine-tuning resolution is 1 pixels, and the maximum horizontal width is 2,048 pixels.</p> | 03h | RW |

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-4 | NA | 0 | RO |
| 3-0 | <p>Horizontal Non-Display Period Fine Tuning</p> <p>This Register is the fine-tuning value for Horizontal Non-display Period, and each fine-tuning resolution is 1 pixels. 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 | HSYNC Start Position 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 8pixel. 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 | 0 | RO |
| 4-0 | HSYNC Pulse Width $\text{HSYNC Pulse Width (Pixels)} = (\text{HPW} + 1) * 8$ | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | Vertical Display Height 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 line, and the formula is as follows: $\text{Vertical Display Height (Line)} = \text{VDHR} + 1$ | DFh | RW |

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Vertical Non-Display Period 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 follows: $\text{Vertical Non-Display Period (Line)} = (\text{VNDR} + 1)$ | 15h | RW |

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | VSYNC Start Position The starting position from the end of display area to the beginning of VSYNC. $VSYNC\ Start\ Position\ (Line) = (VSTR + 1)$ | 0Bh | RW |

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-6 | NA | 0 | RO |
| 5-0 | VSYNC Pulse Width $VSYNC\ Pulse\ Width\ (Line) = (VPWR + 1)$ | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Main Image Start Address 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 | Main Image Width 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 | Main Window Upper-Left Corner X-Coordinates 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 | Main Window Upper-Left Corner Y-Coordinates 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. 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 | PIP Window Display Upper-Left Corner X-Coordinates 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. Y-axis coordinates should less than vertical display height. According to bit of Select Configure PIP 1 or 2 Window's parameters(REG[10h]), value will be configured for relative PIP window. | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | PIP Window Display Upper-Left Corner Y-Coordinates 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 less than vertical display height. According to bit of Select Configure PIP 1 or 2 Window's parameters(REG[10h]), value will be configured for relative PIP window. | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | PIP Image Start Address 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] According to bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window. | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | PIP Image Width REG[32h] mapping to PIW[7:0], bit[1:0] must be 0. REG[33h] bit[5:0] mapping to PIW[13:8], bit[7:6] are not used. Unit: Pixel. The value is physical width, and maximum value is 8192 pixels. This width should less than horizontal display width. According to bit of Select Configure PIP 1 or 2 Window's parameters, value will be configured for relative PIP window. | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | PIP Window 1 or 2 Image Upper-Left Corner X-Coordinates REG[34h] mapping to PWIULX[7:0], bit[1:0] must be 0. REG[35h] bit[4:0] mapping to PWIULX[12:8], bit[7:5] are not used. Unit: Pixel. The sum of X-axis coordinates plus PIP image width must less or equal to 8,191. | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | PIP Windows Display Upper-Left Corner Y-Coordinates REG[36h] mapping to PWIULY[7:0], bit[1:0] must be 0. REG[37h] bit[4:0] mapping to PWIULY[12:8], bit[7:5] are not used. Unit: Pixel. The sum of Y-axis coordinates plus PIP window height should less or equal to 8,191. | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | PIP Window Width REG[38h] mapping to PWW[7:0], bit[1:0] must be 0. REG[39h] bit[5:0] mapping to PWW[13:8], bit[7:6] are not used. Unit: Pixel. Maximum value is 8,192 pixels. | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | PIP Window Height 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. | 0 | RW |

Note 1: Pip 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 in turn by LSB to MSB. 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 in turn from LSB (REG[20h]) to MSB (REG[23h]). When REG[23h] was written, LT7381 will write complete Main Image Start Address to the internal register in actually.

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7 | Gamma Correction Enable 0: Disable 1: Enable Gamma correction is the last output stage. | 0 | RW |
| 6-5 | Gamma Table Select for Host Write Gamma Data 00b: Gamma table for Blue 01b: Gamma table for Green 10b: Gamma table for Red 11b: NA | 0 | RW |
| 4 | Graphic Cursor Enable 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 | Graphic Cursor Selection 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 |
| 1 | Text Cursor Enable 0: Disable 1: Enable Note: Text cursor & Graphic cursor cannot enable simultaneously. Graphic cursor has higher priority than Text cursor if enabled simultaneously. | 0 | RW |

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 0 | Text Cursor Blinking Enable 0: Disable 1: Enable | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | Text Cursor Blink Time Setting 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 | Text Cursor Horizontal Size Setting 00000b: 1 Pixel 00001b: 2 Pixels : : 11111b: 32 Pixels Note: When character is enlarged, the cursor setting will multiply the same times as the character enlargement. | 07h | RW |

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-5 | NA | 0 | RO |
| 4-0 | Text Cursor Vertical Size Setting Unit: Pixel, Zero-based number. Value "0" means 1 pixel. Note: When character is enlarged, the cursor setting will multiply the same times as the character enlargement. | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Graphic Cursor Horizontal Position 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 | Graphic Cursor Vertical Position 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 | Graphic Cursor Color 0 with 256 Colors RGB Format [7:0] = RRRGGGBB | 0 | RW |

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Graphic Cursor Color 1 with 256 Colors RGB Format [7:0] = RRRGGGBB | 0 | RW |

14.6 Geometric Engine Control Registers

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | Start Address of Canvas 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 in linear Addressing mode. | 0 | RW |

REG[55h-54h] Canvas Image Width (CVS_IMWTH)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | Canvas Image Width 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 in linear Addressing mode. | 0 | RW |

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

REG[57h-56h] Active Window Upper-Left Corner X-Coordinates (AWUL_X)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | Active Window Upper-Left Corner X-Coordinates 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 plus Active Window width cannot large than 8,191. These Registers will be ignored if canvas in linear Addressing mode. | 0 | RW |

REG[59h-58h] Active Window Upper-Left Corner Y-Coordinates (AWUL_Y)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | <p>Active Window Upper-Left Corner Y-Coordinates</p> <p>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.</p> <p>Unit: Pixel. The sum of Y-axis coordinates plus Active Window height cannot large than 8,191.</p> <p>These Registers will be ignored if canvas in linear Addressing mode.</p> | 0 | RW |

REG[5Bh-5Ah] Active Window Width (AW_WTH)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | <p>Active Window Width [13:8]</p> <p>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.</p> <p>Unit: Pixel. The value is physical pixel width. Maximum pixel width is 8,192.</p> <p>These Registers will be ignored if canvas in linear Addressing mode.</p> | 0 | RW |

REG[5Dh-5Ch] Active Window Height (AW_HT)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | <p>Height of Active Window [13:8]</p> <p>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.</p> <p>Unit: Pixel. The value is physical pixel width. Maximum pixel width is 8,192.</p> <p>These Registers will be ignored if canvas in linear Addressing mode.</p> | 0 | RW |

REG[5Eh] Color Depth of Canvas & Active Window (AW_COLOR)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-4 | NA | 0 | RO |
| 3 | Select What will Read Back from Graphic Read/Write Position Register 0: Read back Graphic Write position 1: Read back Graphic Read position (Pre-fetch Address) | 0 | RW |
| 2 | Canvas Addressing Mode 0: Block mode (X-Y coordinates addressing) 1: Linear mode | 0 | RW |
| 1-0 | Canvas Image's Color Depth & Memory R/W Data Width <i>In Block Mode:</i> 00b: 8bpp 01b: 16bpp 1xb: 24bpp Note: Monochrome data can input with any one color depth depends on 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 5-2 Canvas and Active Windows.

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Write: Set Graphic Read/Write X-Coordinate position Read: Read Graphic Read/Write X-Coordinate position Read back is Read position or Write position depends on REG[5Eh] bit3, Select to read back Graphic Read/Write position. 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 configure this register.

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | <p>Write: Set Graphic Read/Write X-Coordinate position Read: Read Graphic Read/Write X-Coordinate position Read back is Read position or Write position depends on REG[5Eh] bit3, Select to read back Graphic Read/Write position. 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>When DPRAM In Linear Mode: Memory Read/Write address [31:16], Unit: Byte.</p> <p>When DPRAM In Block Mode: Graphic Read/Write Y-Coordinate [12:0], Unit: Pixel.</p> | 0 | RW |

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

REG[64h-63h] Text Write X-Coordinates Register (F_CURX)

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

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7 | Draw Line / Triangle Start Control <i>Write:</i> 0: Stop the drawing function. 1: Start the drawing function. <i>Read:</i> 0: Drawing function complete. 1: Drawing function is processing. | 0 | RW |
| 6 | NA | 0 | RO |
| 5 | Fill Function for Triangle 0: Non Fill 1: Fill | 0 | RW |
| 4-1 | Draw Triangle or Line Select 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 | Polyline Style 0: Open-end Polyline, 1: Closed Polyline (connect last point to start point) | 0 | RO |

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

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | Draw Line/Triangle Point 1 X-coordinates DLHSR[12:0] 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 draw a rectangle, start point & end point cannot locate at same point or at 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 | Draw Line/Triangle Point 1 Y-coordinates DLVSR[12:0] 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 | Draw Line/Triangle Point 2 X-coordinates DLHER[12:0] 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 | Draw Line/Triangle Point 2 Y-coordinates DLVER[12:0] 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 | Draw Line/Triangle Point 3 X-coordinates DTPH[12:0] 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 | Draw Line/Triangle Point 3 Y-coordinates DTPV[12:0] 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 triangle: If any two endpoints overlap will draw a line. And three endpoints overlap will draw a dot.

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

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7 | Draw Circle / Ellipse / Square /Circle Square Control Write: 0: Stop the drawing function. 1: Start the drawing function. Read: 0: Drawing function complete. 1: Drawing function is processing. | 0 | RW |
| 6 | Fill the Circle / Ellipse / Square / Rounded-rectangle Control 0: Non Fill 1: Fill | 0 | RW |
| 5-4 | Draw Circle / Ellipse / Square / Ellipse Curve / Rounded-rectangle Select 00b: Draw Circle / Ellipse 01b: Draw Arc 10b: Draw Rectangle 11b: Draw Rounded-Rectangle | 0 | RW |
| 3-2 | NA | 0 | RO |
| 1-0 | Draw Circle / Ellipse Curve Part Select (DECP) 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 | Draw Circle/Ellipse/Rounded-rectangle Major-Radius 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 draw a circle, then must set major radius equal to 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 | Draw Circle/Ellipse/Rounded-rectangle Minor-Radius 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 | Draw Circle/Ellipse/Rounded-rectangle Center X-coordinates 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 | Draw Circle/Ellipse/Rounded-rectangle Center Y-coordinates 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 | Foreground Color – Red (for Draw, Text or Color Expansion) 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 | Foreground Color – Green (for Draw, Text or Color Expansion) <ul style="list-style-type: none"> ■ 256 Colors: Green mapping to bit[7:5] ■ 65K Colors: Green mapping to bit[7:2] ■ 16.7M Colors: Green mapping to bit[7:0] | FFh | RW |

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

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

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

14.7 PWM Control Registers

REG[84h] PWM Prescaler Register (PSCLR)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | PWM Pre-scaler Register 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 | Setup PWM Timer-1 Divisor (Select 2 nd Clock Divider's MUX Input for PWM Timer-1) 00b = 1 01b = 1/2 10b = 1/4 11b = 1/8 | 0 | RW |
| 5-4 | Setup PWM Timer-0 Divisor (Select 2 nd Clock Divider's MUX Input for PWM Timer-0) 00b = 1 01b = 1/2 10b = 1/4 11b = 1/8 | 0 | RW |
| 3-2 | PWM[1] Function Control 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 | PWM[0] Function Control 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 | 0 | RO |
| 6 | PWM Timer-1 Output Inverter On/Off Determine the output inverter on/off for Timer 1. 0 = Inverter off 1 = Inverter on for PWM1 | 0 | RW |
| 5 | PWM Timer-1 Auto Reload On/Off Determine auto reload on/off for Timer 1. 0: One-Shot Mode 1: Interval Mode (Auto Reload) | 1 | RW |
| 4 | PWM Timer-1 Start/Stop 0: Stop 1: Start In Interval Mode, Host need program it as 0 to stop PWM timer. In One-shot Mode, this bit will auto clear. The Host may read this bit to know current PWMx is running or stopped. | 0 | RW |
| 3 | PWM Timer-0 Dead Zone Enable 0: Disable 1: Enable | 0 | RW |
| 2 | PWM Timer-0 Output Inverter On/Off Determine the output inverter on/off for Timer 0. 0 = Inverter off 1 = Inverter on for PWM0 | 0 | RW |
| 1 | PWM Timer-0 Auto Reload On/Off Determine auto reload on/off for Timer 0. 0: One-Shot Mode 1: Interval Mode (Auto Reload) | 1 | RW |
| 0 | PWM Timer-0 Start/Stop 0: Stop 1: Start In Interval Mode, Host need program it as 0 to stop PWM timer. In One-shot Mode, this bit will auto clear. The Host may read this bit to know current PWMx is running or stopped. | 0 | RW |

REG[87h] Timer-0 Dead Zone Length Register [DZ_LENGTH]

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Timer-0 Dead Zone Length Register 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 | Timer-0 compare Buffer Register REG[88h] mapping to TCMPB0 [7:0] REG[89h] mapping to TCMPB0 [15:8] The Timer-0 Compare Buffer Register 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 | Timer-0 Count Buffer Register [15:0] 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 down to 0 and the Reload_EN is enabled, the PWM will overloads 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 | Timer-1 Compare Buffer Register REG[8Ch] mapping to TCMPB1 [7:0] REG[8Dh] mapping to TCMPB1 [15:8] The Timer-1 Compare Buffer Register 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 | Timer-1 Count Buffer Register [15:0] 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 down to 0 and the Reload_EN is enabled, the PWM will overloads 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 |

14.8 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 | BTE Function Enable / Status <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 enable, Normal Host R/W memory through canvas[active window] doesn't allow. | 0 | RW |
| 3-1 | NA | 0 | RO |
| 0 | Pattern Format 0: 8*8 1: 16*16 | 0 | RW |

REG[91h] BTE Function Control Register1 (BLT_CTRL1)

| Bit | Description | Default | Access | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------|---|----------|-------------|-------|---------------|-------|---|-------|--------------------|-------|-----------|-------|--------------------|-------|-----------|-------|----------------|-------|---|-------|---------------|-------|-----------------------|-------|------|-------|----------------|-------|------|-------|----------------|-------|-----------|-------|---------------|---|----|
| 7-4 | <p>BTE ROP Code or Color Expansion Starting ROP is the acronym for Raster Operation. Some of BTE operation code has to collocate with ROP for the detailed function.</p> <p style="text-align: center;">Table 14-3: BTE ROP Code</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <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>$\sim S0 \cdot \sim S1$ or $\sim (S0+S1)$</td></tr> <tr><td>0010b</td><td>$\sim S0 \cdot S1$</td></tr> <tr><td>0011b</td><td>$\sim S0$</td></tr> <tr><td>0100b</td><td>$S0 \cdot \sim S1$</td></tr> <tr><td>0101b</td><td>$\sim S1$</td></tr> <tr><td>0110b</td><td>$S0 \wedge S1$</td></tr> <tr><td>0111b</td><td>$\sim S0 + \sim S1$ or $\sim (S0 \cdot S1)$</td></tr> <tr><td>1000b</td><td>$S0 \cdot S1$</td></tr> <tr><td>1001b</td><td>$\sim (S0 \wedge S1)$</td></tr> <tr><td>1010b</td><td>$S1$</td></tr> <tr><td>1011b</td><td>$\sim S0 + S1$</td></tr> <tr><td>1100b</td><td>$S0$</td></tr> <tr><td>1101b</td><td>$S0 + \sim S1$</td></tr> <tr><td>1110b</td><td>$S0 + S1$</td></tr> <tr><td>1111b</td><td>1 (Whiteness)</td></tr> </tbody> </table> <p>If BTE operation code function are 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, value should within 0 to 7. For 16-bits Host, value should within 0 to 15.</p> | Bit[7:4] | Description | 0000b | 0 (Blackness) | 0001b | $\sim S0 \cdot \sim S1$ or $\sim (S0+S1)$ | 0010b | $\sim S0 \cdot S1$ | 0011b | $\sim S0$ | 0100b | $S0 \cdot \sim S1$ | 0101b | $\sim S1$ | 0110b | $S0 \wedge S1$ | 0111b | $\sim S0 + \sim S1$ or $\sim (S0 \cdot S1)$ | 1000b | $S0 \cdot S1$ | 1001b | $\sim (S0 \wedge S1)$ | 1010b | $S1$ | 1011b | $\sim S0 + S1$ | 1100b | $S0$ | 1101b | $S0 + \sim S1$ | 1110b | $S0 + S1$ | 1111b | 1 (Whiteness) | 0 | RW |
| Bit[7:4] | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0000b | 0 (Blackness) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0001b | $\sim S0 \cdot \sim S1$ or $\sim (S0+S1)$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0010b | $\sim S0 \cdot S1$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0011b | $\sim S0$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0100b | $S0 \cdot \sim S1$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0101b | $\sim S1$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0110b | $S0 \wedge S1$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0111b | $\sim S0 + \sim S1$ or $\sim (S0 \cdot S1)$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1000b | $S0 \cdot S1$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1001b | $\sim (S0 \wedge S1)$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1010b | $S1$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1011b | $\sim S0 + S1$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1100b | $S0$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1101b | $S0 + \sim S1$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1110b | $S0 + S1$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1111b | 1 (Whiteness) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3-0 | <p>BTE Operation Code bit[3:0] LT7381 embedded a 2D BTE Engine, it can execute 13 BTE functions. Some of BTE Operation Code has to accommodate with the ROP code for the advance function.</p> | 0 | RW | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 14-4: BTE Operation Code

| REG[91h] Bit[3:0] | Description |
|----------------------|--|
| 0000b | MCU Write with ROP S0: Data comes from Host S1: Data comes from Memory D: According to ROP write to Memory |
| 0001b | Reserved |
| 0010b | Memory Copy with ROP S0: Data comes from Memory S1: Data comes from Memory D: According to ROP Write to memory |
| 0011b | Reserved |
| 0100b | MCU Write with Chroma Keying (without ROP) S0: Data comes from Host If the Host data is not the same color as the Chroma Key (Background Color Register), the data is written to the destination memory. |
| 0101b | Memory Copy (move) with Chroma keying (without ROP) S0: Data comes from Memory If the S0 data is not the same color as the chroma key Background Color Register), the data is written to the destination. |
| 0110b | Pattern Fill with ROP Pattern was specified by S0. |
| 0111b | Pattern Fill with Chroma Keying Pattern was specified by S0. If the data for S0 is not the same as the Chroma Key (Background Color) color, then it is written to the destination memory. |
| 1000b | MCU Write with Color Expansion S0: Data comes from Host BTE converts it to the specified color and color depth and writes to the destination memory. |
| 1001b | MCU Write with Color Expansion and Chroma Keying 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 in the destination memory setting. |
| 1010b | Memory Copy with Opacity S0, S1 & D: Data come from Memory |

Table 14-4: BTE Operation Code (Continued)

| REG[91h] Bit[3:0] | Description |
|----------------------|---|
| 1011b | MCU Write with Opacity S0: Data comes from Host S1: Data comes from Memory D: Refer to the alpha blending operation and write to the destination memory. |
| 1100b | Solid Fill The write value is a register setting value and the target is written to the destination memory. |
| 1101b | Reserved |
| 1110b | Memory Copy with Color Expansion S0 and D are in memory and S1 is not in use. S0 must be loaded with the microprocessor's write or DMA preload 8bpp or 16bpp of color depth into memory, so the S0 color depth should follow that color depth. |
| 1111b | Memory Copy with Color Expansion and Chroma Keying S0 and D are in memory and S1 is not in use. S0 must be loaded with the microprocessor's write or DMA 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 D is not written to any data. If the S0 data bit = 1, the Foreground Color data will be written to D. |

REG[92h] Source 0/1 & Destination Color Depth (BLT_COLR)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7 | NA | 0 | RO |
| 6-5 | S0 Color Depth 00b: 256 Color (8bpp) 01b: 64k Color (16bpp) 1xb: 16M Color (24bpp) | 0 | RW |
| 4-2 | S1 Color Depth 000b: 256 Color (8bpp) 001b: 64k Color (16bpp) 010b: 16M Color (24bpp) 011b: Constant color (S1 memory start address' setting definition change 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 | Destination Color Depth 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 | Source 0 Memory Start Address [31:2] 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] Note: REG[93h] bit[1:0] fix at 0. | 0 | RW |

REG[97h-98h] Source 0 Image Width (S0_WTH)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Source 0 Image Width [12:2] 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. Note: It must be divisible by 4, and REG[97h] bit[1:0] must fix at 0. Unit: Pixel. | 0 | RW |

REG[99h-9Ah] Source 0 Window Upper-Left Corner X-Coordinates (S0_X)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Source 0 Window Upper-Left Corner X-Coordinates [12:0] 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 | Source 0 Window Upper-Left Corner Y-Coordinates [12:0] 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 | Source 1 Memory Start Address [31:2] 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] Note1: REG[9Dh] bit[1:0] must fix at 0. Note2: If source 1(S0) set as Constant Color then S1 memory start address' setting definition change 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 for color definition. | 0 | RW |

REG[A1h-A2h] Source 1 Image Width (S1_WTH)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | Source 1 Image Width [12:2] 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. Note: It must be divisible by 4, and REG[A1h] bit[1:0] must fix at 0. Unit: Pixel. | 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 | Source 1 Window Upper-Left Corner Y-Coordinates [12:0] 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. | 0 | RW |

REG[A7h-AAh] Destination Memory Start Address (DT_STR)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-2 | Destination Memory Start Address [31:2] 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] Note: REG[A7h] bit[1:0] must fix at 0. | 0 | RW |

Note: The destination memory start address cannot be in the source 0 and Source 1 processing block, otherwise there will be an incorrect result output.

Start Address ~ S0/1's (Image_Width)* (Image_Height)* ([1|2|3]Color Depth)

REG[ABh-ACH] Destination Image Width (DT_WTH)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | Destination Image Width [12:2] 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. Note: It must be divisible by 4. And REG[ABh] bit[1:0] must fix 0. Unit: Pixel. | 0 | RW |

REG[ADh-AEh] Destination Window Upper-Left Corner X-Coordinates (DT_X)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | Destination Window Upper-Left Corner X-Coordinates [12:0] 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 | Destination Window Upper-Left Corner X-Coordinates [12:0] 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 | <p>BTE Window Width [12:0] 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.</p> <p>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.</p> | 0 | RW |

REG[B3h-B4h] BTE Window Height (BLT_HIG)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | <p>Destination Image Height [12:0] 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.</p> <p>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.</p> | 0 | RW |

REG[B5h] Alpha Blending (APB_CTRL)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-4 | NA | 0 | RO |
| 5-0 | <p>Window Alpha Blending Effect for S0 & S1 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.</p> <p>00h: 0 01h: 1/32 02h: 2/32 : : 1Eh: 30/32 1Fh: 31/32 2Xh: 1</p> <p>Output Effect $= [S0 \text{ image} * (1 - \text{Alpha Setting Value})] + (S1 \text{ Image} * \text{Alpha Setting Value})$</p> | 0 | RW |

REG[B6h-CBh] RESERVED

| Bit | 说明 | 默认值 | 存取模式 |
|-----|-----|-----|------|
| 7-0 | 未使用 | 0 | RO |

14.9 Text Engine Registers

REG[CCh] Character Control Register 0 (CCR0)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7:6 | Character Source Selection 00b: Select internal CGROM Character 01b: Select external CGROM Character 10b: Select user-defined Character. 11b: NA | 0 | RW |
| 5-4 | Character Height Setting for user-defined Character 00b: 16 dots, ex. 8*16 / 16*16 01b: 24 dots, ex. 12*24 / 24*24 10b: 32 dots, ex. 16*32 / 32*32 Note: 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 size 8*16 / 12*24 / 16*32. | 0 | RW |
| 3-2 | N | 0 | RO |
| 1-0 | Character Selection for Internal CGROM 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 | Full Alignment Selection 0: Full alignment disable 1: Full alignment enable. When Full alignment enable, displayed character width is equal to (Character Height)/2 if character width equal or small than (Character Height)/2, otherwise displayed font width is equal to Character Height. | 0 | RW |
| 6 | Chroma Keying Enable on Text Input 0: Character's background displayed with specified color. 1: Character's background displayed with original canvas' background. | 0 | RW |
| 5 | NA | 0 | RO |

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 4 | Character Rotation 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 (it should accommodate with set VDIR as 1). This attribute can be changed only when previous Text write finished (Core_Busy = 0) | 0 | RW |
| 3-2 | Character Width Enlargement Factor 00b: X1. 01b: X2. 10b: X3. 11b: X4. | 0 | RW |
| 1-0 | Character Height Enlargement Factor 00b: X1. 01b: X2. 10b: X3. 11b: X4. | 0 | RW |

REG[CEh-CFh] RESERVED

| Bit | Description | Default | Access |
|-----|-------------|---------|--------|
| 7-0 | NA | 0 | RO |

REG[D0h] Character Line gap Setting Register (FLDR)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-5 | NA | 0 | RO |
| 4-0 | Character Line Gap Setting Setting the character line gap when meet active window boundary. (Unit: pixel) Color of gap will fill-in background color. It won't be enlarged by character enlargement function. | 0 | RW |

REG[D1h] Character to Character Space Setting Register (F2FSSR)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-6 | NA | 0 | RW |
| 5-0 | Character to Character Space Setting 00h: 0 pixel 01h: 1 pixel 02h: 2 pixels : : 3Fh: 63 pixels Color of space will fill-in background color. It won't be enlarged by character enlargement function. | 0 | RW |

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

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

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

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

REG[D5h] Background Color Register - Red (BGCR)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | Background Color – Red (for Text or Color Expansion) <ul style="list-style-type: none"> ■ 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] Note: No matter background transparency is enabled or not, don't set same value with Foreground Color otherwise image or text will become a square with Foreground Color even BTE function. | 00h | RW |

REG[D6h] Background Color Register - Green (BGCG)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Background Color –Green (for Text or Color Expansion) <ul style="list-style-type: none"> ■ 256 Colors: Green mapping to bit[7:5] ■ 65K Colors: Green mapping to bit[7:2] ■ 16.7M Colors: Green mapping to bit[7:0] Note: No matter background transparency is enabled or not, don't set same value with Foreground Color otherwise image or text will become a square with Foreground Color even BTE function. | 00h | RW |

REG[D7h] Background Color Register - Blue (BGCB)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Background Color –Blue (for Text or Color Expansion) <ul style="list-style-type: none"> ■ 256 Colors: Blue mapping to bit[7:6] ■ 65K Colors: Blue mapping to bit[7:3] ■ 16.7M Colors: Blue mapping to bit[7:0] Note: No matter background transparency is enabled or not, don't set same value with Foreground Color otherwise image or text will become a square with Foreground Color even BTE function. | 00h | RW |

REG[D8h] – REG[DAh]: Reserved

| Bit | Description | Default | Access |
|-----|-------------|---------|--------|
| 7-0 | NA | 0 | RO |

REG[DBh] CGRAM Start Address 0 (CGRAM_STR0)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | <p>CGRAM Start ADDRESS [7:0] User-defined Characters space</p> <p>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]</p> <p>Host must use canvas image setting to organize CGRAM data and set CGRAM address to tell engine where to fetch CGRAM data.</p> | 0 | RW |

If user wants to change rotate attribute, character line gap, character-to-character space, foreground color, background color and Text/graphic mode setting, he must make sure Task_Busy (Status Register bit3) status bit is low.

14.10 Power Management Control Register

REG[DFh]: Power Management Register (PMU)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7 | <p>Enter Power Saving State</p> <p>0: Normal state or wake up from power saving state 1: Enter power saving state.</p> <p>Note:</p> <p>There are 3 ways to wake up from power saving state: External interrupt event, Key Scan wakeup and Software wakeup.</p> <p>Write this bit to 0 will cause Software wakeup. It will be cleared until chip resume. MPU must wait until system quit from power saving state than allow to write other registers. User may check this bit or check status bit1 (power saving)</p> | 0 | RW |
| 6-2 | NA | 0 | RO |
| 1-0 | <p>Power Saving Mode Definition</p> <p>00b: NA 01b: Standby Mode, CCLK & PCLK will Stop. MCLK is provided by MPLL. 10b: Suspend Mode, CCLK & PCLK will Stop. MCLK is provided by OSC. 11b: Sleep Mode, All of Clocks and PLL will Stop.</p> | 3 | RW |

14.11 Display RAM Control Register

REG[E0h] SDRAM Attribute Register (SDRAR)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7 | SDRAM Power Saving 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 | SDRAM Bank Number, SDR_BANK 1: 4 Banks <i>Must set to 1 after reset.</i> | 1 | RW |
| 4-3 | SDRAM Row Addressing, SDR_ROW 00b: 2K (A0-A10) <i>Must set to 00 after reset.</i> | 1 | RW |
| 2-0 | SDRAM Column Addressing, SDR_COL 000b: 256 (A0-A7) <i>Must set to 000 after reset.</i> | 0 | RW |

Table 14-5: The initialize of REG[E0h]

| Embedded Display RAM Type | REG[E0h] | Description |
|---------------------------|-------------|---|
| 32Mb(4MB, 2M*16) | 0x20 | Bank no: 4, Row Size: 2048, Column Size: 256 |

Note: The value of register REG[E0h] must be set according to above table. Otherwise, the display of TFT panel will abnormal and the image is garbled.

REG[E1h] SDRAM Mode Register & Extended Mode Register (SDRMD)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-3 | Must keep 0. | 0 | RW |
| 2-0 | SDRAM CAS latency, SDR_CASLAT 010b: 2 SDRAM clock 011b: 3 SDRAM clock Others: Reserved Note: The suggest setting value of this register is 03h . This register was locked after SDR_INITDONE (REG[E4h] bit0) was set as 1. | 011b | RW |

REG[E2h-E3h] SDRAM Auto Refresh Interval (SDR_REF)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | SDRAM Auto Refresh Interval REG[E2h] mapping to SDR_REF [7:0]. REG[E3h] mapping to SDR_REF [15:8] The internal refresh time is determined according to the period specification of the SDRAM's refresh and the row size. For example, if the SDRAM frequency is 100MHz, SDRAM's refresh period T_{ref} is 64ms, and the row size is 4,096, then the internal refresh time should be less than $64 * 10^{-3} / 4096 * 100 * 10^6 \approx 1562 = 61Ah$.. Therefore the REG[E3h][E2h] is set 030Dh. Note: If this register is set to 0000h, SDRAM automatic refresh will be prohibited. | 00h | RW |

Table 14-6: The Reference Setting of REG[E3h-E2h]

| Model | REG[E3h] | REG[E2h] |
|--------|----------|----------|
| LT7381 | 06h | 1Ah |

REG[E4h] SDRAM Control Register (SDRCR)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-4 | Must keep 0. | 0 | RW |
| 3 | Report Warning Condition 0: Disable or Clear warning flag 1: Disable or Clear warning flag Warning condition are memory read cycle close to SDRAM maximum address boundary (may over maximum address minus 512 bytes) or out of range or SDRAM bandwidth insufficient to fulfill panel's frame rate, then this warning event will be latched, user could check this bit to do some judgments. That warning flag could be cleared by set this bit as 0. | 0 | RW |
| 2 | SDRAM Timing Parameter Register Enable, SDR_PARAMEN 0: Disable Display RAM timing parameter registers 1: Enable Display RAM timing parameter registers | 0 | RW |
| 1 | Enter Power Saving Mode, SDR_PSAVING 0 to 1 transition will enter power saving mode 1 to 0 transition will exit power saving mode | 0 | RW |
| 0 | Start SDRAM Initialization Procedure, SDR_INITDONE 0 to 1 transition will execute Display RAM initialization procedure. Read value '1' means Display RAM is initialized and ready for access. Once it was written as 1, it cannot be rewrite as 0. 1 to 0 transition without have any 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 | Time from Load Mode Command to Active/Refresh Command (T_{MRD}) 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 | Auto Refresh Period, T_{RFC} 0h – Fh: 1 ~ 16 SDRAM Clock (As REG[E0h] bit[3:0]) | 8 | RW |
| 3-0 | Time of Exit SELF Refresh-to-ACTIVE Command (T_{XSR}) 0h – Fh: 1 ~ 16 SDRAM Clock | 7 | RW |

REG[E2h] SDRAM Timing Parameter 3

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-4 | Time of Pre-charge Command Period (T_{RP}, 15/20ns) 0h – Fh: 1 ~ 16 SDRAM Clock | 2 | RW |
| 3-0 | Time of WRITE Recovery Time (T_{WR}) 0h – Fh: 1 ~ 16 SDRAM Clock | 0 | RW |

REG[E3h] SDRAM Timing Parameter 4

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-4 | Delay Time of Active-to-Read/Write (T_{RCD}) 0h – Fh: 1 ~ 16 SDRAM Clock | 2 | RW |
| 3-0 | Time of Active-to-Precharge (T_{RAS}) 0h – Fh: 1 ~ 16 SDRAM Clock | 6 | RW |

14.12 I2C Master Register

REG[E5h-E6h] I2C Master Clock Prescaler Register (I2CMCK)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | I2C Master Clock Pre-scaler [15:0] 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 | I2C Master Transmit [7:0] | 0 | RW |

REG[E8h] I2C Master Receiver Register (I2CMRXR)

| Bit | Description | Default | Access |
|-----|----------------------------------|---------|--------|
| 7-0 | I2C Master Receiver [7:0] | 0 | RW |

REG[E9h] I2C Master Command Register (I2CMCMD)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7 | Start Command Generate (repeated) start condition and be cleared by hardware Automatically Note: This bit is always read as 0. | 0 | RW |
| 6 | Stop Command Write 1: Generate stop condition and be cleared by hardware. Note: This bit is always read as 0. | 0 | RW |
| 5 | Read Command Write 1: Read form slave and be cleared by hardware automatically. Note: This bit is always read as 0. | 0 | RW |
| 4 | Write Command Write 1: Write to slave and be cleared by hardware automatically. Note: This bit is always read as 0. | 0 | RW |
| 3 | Acknowledge Command Write 0: Send ACK Write 1: Send NACK Note: This bit is always read as 0. | 0 | RW |
| 2-1 | NA | 0 | RO |

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 0 | Noise Filter 0: Disable 1: Enable | 0 | RW |

REG[EAh] I2C Master Status Register (I2CMST)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7 | Received Acknowledge from Slave 0: Acknowledge received. 1: No Acknowledge received. | 0 | RO |
| 6 | I2C Bus is Busy 0: Idle. '0' after STOP signal detected 1: Busy. '1' after START signal detected | 0 | RO |
| 5-2 | NA | 0 | RO |
| 1 | I2C Transfer in Progress 0: when transfer complete 1: when transferring data | 0 | RO |
| 0 | Arbitration Lost State When LT7381 lost Arbitration, this bit will be set 1. Note: When a Stop signal is detected, but is not required, then it means the condition of arbitration loss. The LT7381 I2C Master will drive SDA to 1, but the other Master will drive SDA to 0. | 0 | RO |

14.13 GPIO Register

REG[F0h] GPIO-A Direction (GPIOAD)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | PIOA In/Out Control 0: Output. 1: Input. | FFh | RW |

REG[F1h] GPIO-A (GPIOA)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | GPIOA Data 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]. And 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 | GPIOB[4] Data Write: Output Data to GPIOB[4] Read: Read Data from GPIOB[4] GPIOB[4] is shared with KO[0], GPIB[4] is shared with KI[0]. | NA | RW |
| 3-0 | GPIB[3:0] Data 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 | GPIOC In/Out Control 0: Output 1: Input. | FFh | RW |

REG[F4h] GPIO-C (GPIOC)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7 | GPIOC[7] Data 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 disabled. | NA | RW |
| 6-5 | NA | NA | RW |
| 4-0 | GPIOC[4:0] Data 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 disabled. | NA | RW |

REG[F5h] GPIO-D Direction (GPIODD)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7-0 | GPIOD In/Out Control 0: Output 1: Input | FFh | RW |

REG[F6h] GPIO-D (GPIOD)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | GPIOD Data 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.

14.14 Keypad-scan Control Register

REG[FBh] Keypad-scan Control Register 1 (KSCR1)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7 | Must set as 0. | 0 | 0 |
| 6 | Long Key Enable 1: Enable. Long key period is set by KSCR2 bit[4:2]. 0: Disable. | 0 | RW |
| 5-4 | Short Key de-bounce Times De-bounce times of keypad scan frequency. 00b: 4 01b: 8 10b: 16 11b: 32 | 0 | RW |
| 3 | Repeatable Key Enable 0: Disable Repeatable Key 1: Enable Repeatable Key i.e., if key is always pressed, then controller will repeat issue key interrupt in every short key de-bounce time (long key disable) or long key recognition time (long key enable) after user clear interrupt flag. | 0 | RW |
| 2-0 | Row Scan Time $T_{KEYCLK} = (1 / F_{SYSCLK}) * 2,048$ 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 = $T_{KEYCLK} * 128$ LT7381's Keypad-scan controller supports 5x5 keys. Total Key pads can time = Row Scan Time * 5. | 0 | RW |

REG[FCh] Keypad-scan Controller Register 2 (KSCR2)

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 7 | Keypad-scan Wakeup Enable 0: Keypad-scan Wakeup function is disabled. 1: Keypad-scan Wakeup function is enabled | 0 | R/W |
| 6 | Key Released Interrupt Enable 0: Without interrupt event when all key released 1: Generate an interrupt when all key released | 0 | RW |

| Bit | Description | Default | Access |
|-----|--|---------|--------|
| 5 | NA | 0 | RO |
| 4-2 | Long Key Recognition Factor It determines long key recognition time since short key was recognized. Value from 0 to 7. LongKey Recognition Time $= \text{RowScanTime} * 5 * (\text{Long Key Recognition Factor} + 1) * 1,024$ | 0 | RW |
| 1-0 | Numbers of Key Hit 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. It will auto return to 0 if without any keys are pressed for a de-bounce time. | 0 | RO |

REG[FDh] Keypad-scan Data Register (KSDR1)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Key Strobe Data1 The corresponding key code 1 that is pressed. It will auto return to FFh if without any keys are pressed for a de-bounce time. | TBD | RO |

REG[FEh] Keypad-scan Data Register (KSDR2)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Key Strobe Data2 The corresponding key code 2 that is pressed. It will auto return to FFh if without any keys are pressed for a de-bounce time. | TBD | RO |

REG[FFh] Keypad-scan Data Register (KSDR3)

| Bit | Description | Default | Access |
|-----|---|---------|--------|
| 7-0 | Key Strobe Data3 The corresponding key code 3 that is pressed. It will auto return to FFh if without any keys are pressed for a de-bounce time. | TBD | RO |

Package Information

■ **LT7381 (LQFP-128pin)**

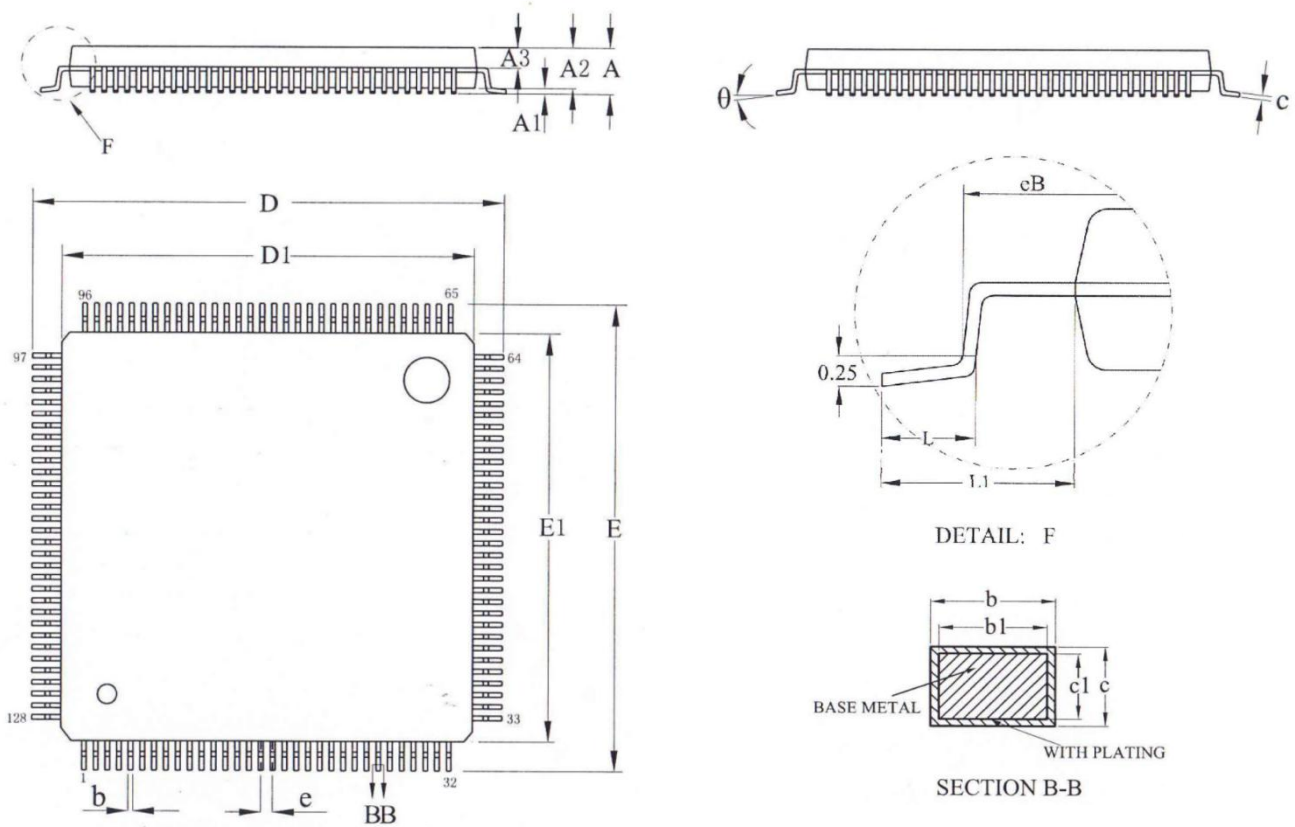


Figure B-1: 128Pin LQFP Outline

Table B-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 | | | | |

 **Revision****Table B-4: Revision**

| Version | Date | Description |
|----------------|-------------|----------------------------|
| V1.0 | 2019/1/20 | LT7381 Preliminary Release |

 **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 <http://www.levetop.cn>.