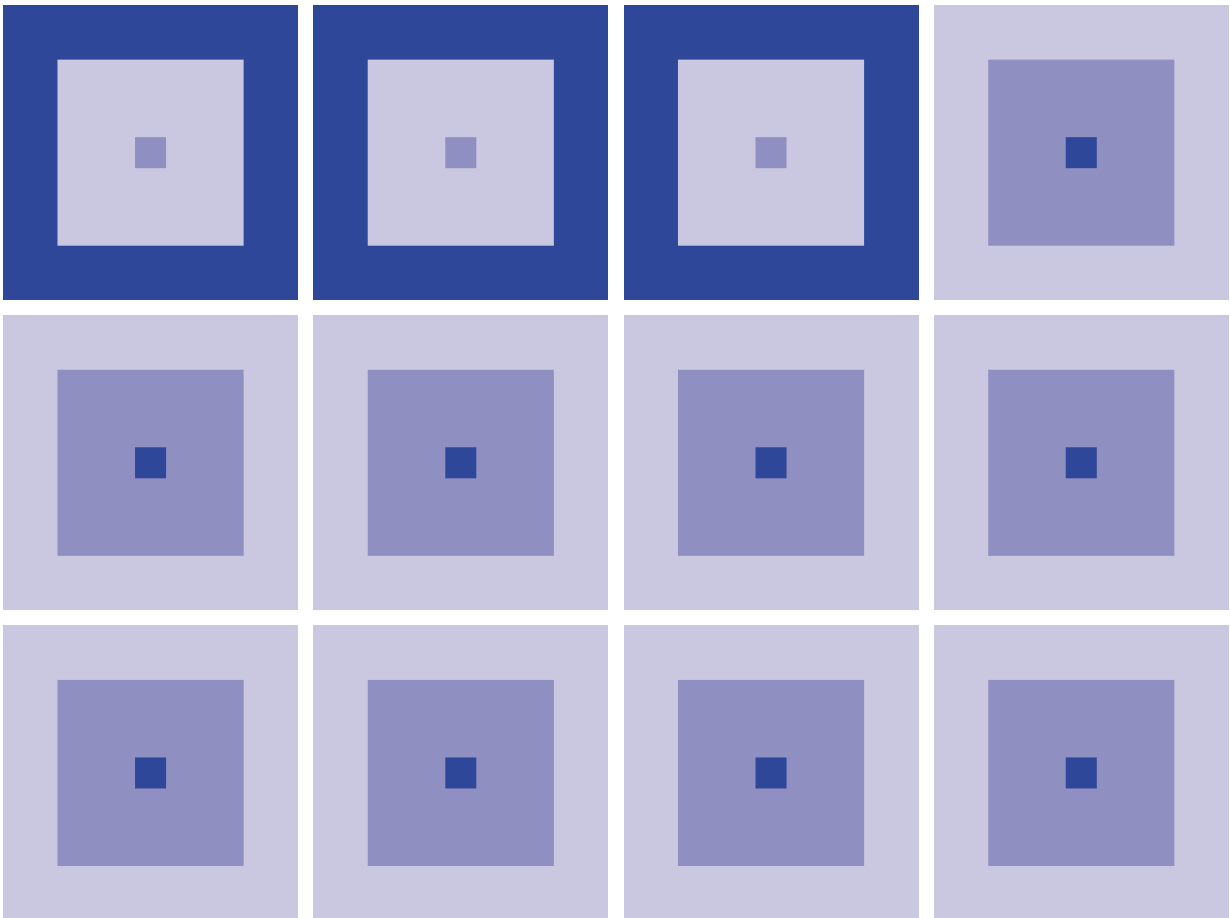
MF1020-02

EPSON

Dot Matrix Graphics LCD Controller

# S1D13503 Series
# Technical Manual



ENERGY SAVING EPSON

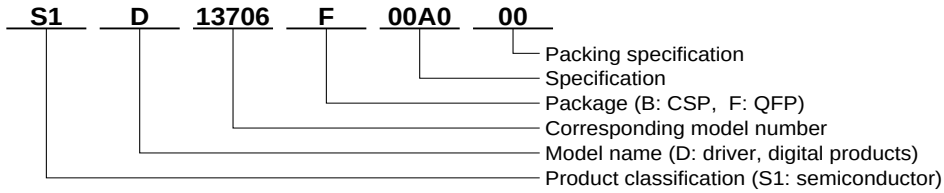
SEIKO EPSON CORPORATION

## The information of the product number change

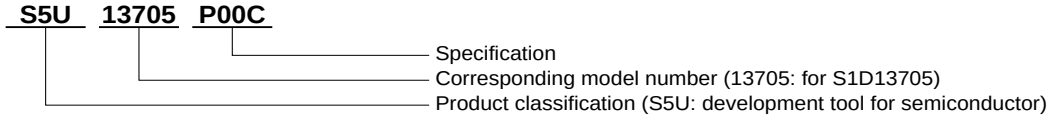
Starting April 1, 2001, the product number will be changed as listed below. To order from April 1, 2001 please use the new product number. For further information, please contact Epson sales representative.

## Configuration of product number

● Devices




● Evaluation Board




## Comparison table between new and previous number

• S1D13305 Series

| Previous No. | New No. |
|---|---|
| SED1335 Series | S1D13305 Series |
| SED1335D0A | S1D13305D00A |
| SED1335F0A | S1D13305F00A |
| SED1335F0B | S1D13305F00B |

• S1D1350x Series

| Previous No. | New No. |
|---|---|
| SED135x Series | S1D1350x Series |
| SED1353D0A | S1D13503D00A |
| SED1353F0A | S1D13503F00A |
| SED1353F1A | S1D13503F01A |
| SED1354F0A | S1D13504F00A |
| SED1354F1A | S1D13504F01A |
| SED1354F2A | S1D13504F02A |
| SED1355F0A | S1D13505F00A |
| SED1356F0A | S1D13506F00A |

• S1D1370x Series

| Previous No. | New No. |
|---|---|
| SED137x Series | S1D1370x Series |
| SED1374F0A | S1D13704F00A |
| SED1375F0A | S1D13705F00A |
| SED1376B0A | S1D13706B00A |
| SED1376F0A | S1D13706F00A |
| SED1378 Series | S1D13708 Series |

• S1D13A0x Series

| Previous No. | New No. |
|---|---|
| SED13Ax Series | S1D13A0x Series |
| SED13A3F0A | S1D13A03F00A |
| SED13A3B0B | S1D13A03B00B |
| SED13A4B0B | S1D13A04B00B |

• S1D1380x Series

| Previous No. | New No. |
|---|---|
| SED138x Series | S1D1380x Series |
| SED1386F0A | S1D13806F00A |

## Comparison table between new and previous number of Evaluation Boards

• S1D1350x Series

| Previous No. | New No. |
|---|---|
| SDU1353#0C | S5U13503P00C |
| SDU1354#0C | S5U13504P00C |
| SDU1355#0C | S5U13505P00C |
| SDU1356#0C | S5U13506P00C |

• S1D1370x Series

| Previous No. | New No. |
|---|---|
| SDU1374#0C | S5U13704P00C |
| SDU1375#0C | S5U13705P00C |
| SDU1376#0C | S5U13706P00C |
| SDU1376BVR | S5U13706B32R |
| SDU1378#0C | S5U13708P00C |

• S1D13A0x Series

| Previous No. | New No. |
|---|---|
| SDU13A3#0C | S5U13A03P00C |
| SDU13A4#0C | S5U13A04P00C |

• S1D1380x Series

| Previous No. | New No. |
|---|---|
| SDU1386#0C | S5U13806P00C |

# S1D13503F00A Technical Manual

## HARDWARE FUNCTIONAL SPECIFICATION

## PROGRAMMING NOTES AND EXAMPLES

## UTILITIES

## S5U13503P00C PCI BUS EVALUATION BOARD USER'S MANUAL

## APPLICATION NOTES

# S1D13503 Series
## Dot Matrix Graphic LCD Controller

## Hardware Functional Specification

# HARDWARE FUNCTIONAL SPECIFICATION

## Table of Contents

## List of Figures

## List of Tables

# 1 INTRODUCTION

## 1.1 Scope

This is the Functional Specification for the S1D13503 Dot Matrix Graphic Color LCD Controller. Included in this document are timing diagrams, AC and DC characteristics, register descriptions, and power management descriptions. This document is intended for two audiences, Video Subsystem Designers and Software Developers.

## 1.2 Overview Description

This device is designed for products where low cost, low power consumption, and low component count are the major design considerations. This chip operates from 2.7 Volts to 5.5 Volts and up to 25 MHz to suit different power consumption, speed and cost requirements. The S1D13503 offers a flexible microprocessor interface and is pin compatible with the SED1352.

The S1D13503 is capable of displaying a maximum of 16 levels of gray shade or 256 simultaneous colors. In gray shade modes, a $16 \times 4$ Look-Up Table is provided to allow remapping of the 16 possible gray shades displayed on the LCD panel. In color modes, three $16 \times 4$ Look-Up Tables are provided to allow remapping of the 4096 possible colors displayed on the LCD panel. The S1D13503 can interface to an MC68000 family microprocessor or an 8/16-bit MPU/Bus with minimum external “glue” logic. This device can directly control up to 128K bytes of static RAM with a 16-bit data path, or up to 64K bytes with an 8-bit data path.

# 2 FEATURES

## 2.1 Technology

- low power CMOS
- 2.7 to 5.5 volt operation
- S1D13503F00A is 100 pin QFP5-S2 surface mount package
- S1D13503F01A is 100 pin QFP15-STD surface mount package
- S1D13503D00A is Die form

## 2.2 System

- maximum 25 MHz input clock (or pixel clock)
- 2-terminal crystal input for internal oscillator or direct connection to external clock source
- maximum 16 MHz, 16-bit MC68000 MPU interface
- 8-bit or 16-bit MPU/Bus interface with memory accesses controlled by a READY (or WAIT#) signal
- option to use built-in index register or direct-mapping to access one of sixteen internal registers
- 8-bit or 16-bit SRAM data bus interface configurations
- display memory configurations:
  - 128K bytes using one 64K × 16 SRAM
  - 128K bytes using two 64K × 8 SRAMs
  - 64K bytes using two 32K × 8 SRAMs
  - 40K bytes using one 8K × 8 and one 32K × 8 SRAM
  - 32K bytes using one 32K × 8 SRAM
  - 16K bytes using two 8K × 8 SRAMs
  - 8K bytes using one 8K × 8 SRAM

## 2.3 Display Modes

- 1 bit-per-pixel, black-and-white display mode
- 2/4 bits-per-pixel, 4/16 level gray shade display modes
- 2/4/8 bits-per-pixel, 4/16/256 level color display modes
- one 16 × 4 Look-Up Table provided for gray shade display modes
- three 16 × 4 Look-Up Tables provided for color display modes
- maximum 16 shades of gray
- maximum 256 simultaneous colors from a possible 4096 colors
- split screen display mode (see AUX[0A])
- virtual display mode (see AUX[0D])

**Note:** 256 color display mode support requires a 16-bit display memory interface.

## 2.4 Display Support

- example resolutions:
  - 1024 × 768 black-and-white
  - 640 × 480 with 4 colors/grays
  - 640 × 400 with 16 colors/grays
  - 320 × 240 with 256 colors
- passive monochrome LCD panels:
  - 4-bit single (4-bit data transfer)
  - 8-bit single (8-bit data transfer)
  - 8-bit dual (4-bit data transfer for each half panel)
- passive color LCD panels:
  - 4-bit single (4-bit data transfer)
  - 8-bit single (8-bit data transfer)
  - 8-bit dual (4-bit data transfer for each half panel)
  - 16-bit single (8-bit data transfer with external circuit)
  - 16-bit dual (8-bit data transfer with external circuit)

See Section 9.5 on page 58 for complete details.

## 2.5 Power Management

- two software power-save modes
- low power consumption
- panel power control switch (see AUX[01] bit 4)

# 3 TYPICAL SYSTEM BLOCK DIAGRAMS

The following figures show typical system implementations of the S1D13503. All of the following block diagrams are shown without SRAM or LCD display. Refer to the interface specific Application Notes for complete details.

## *16-Bit MC68000 MPU*

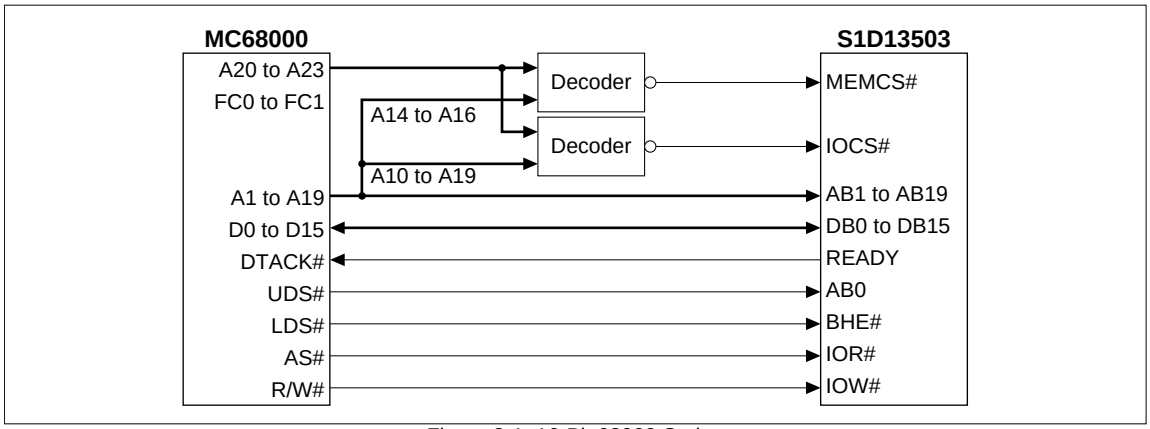



Figure 3-1 16-Bit 68000 Series
(example implementation only - actual may vary)

## *MPU with READY (or WAIT#) Signal*

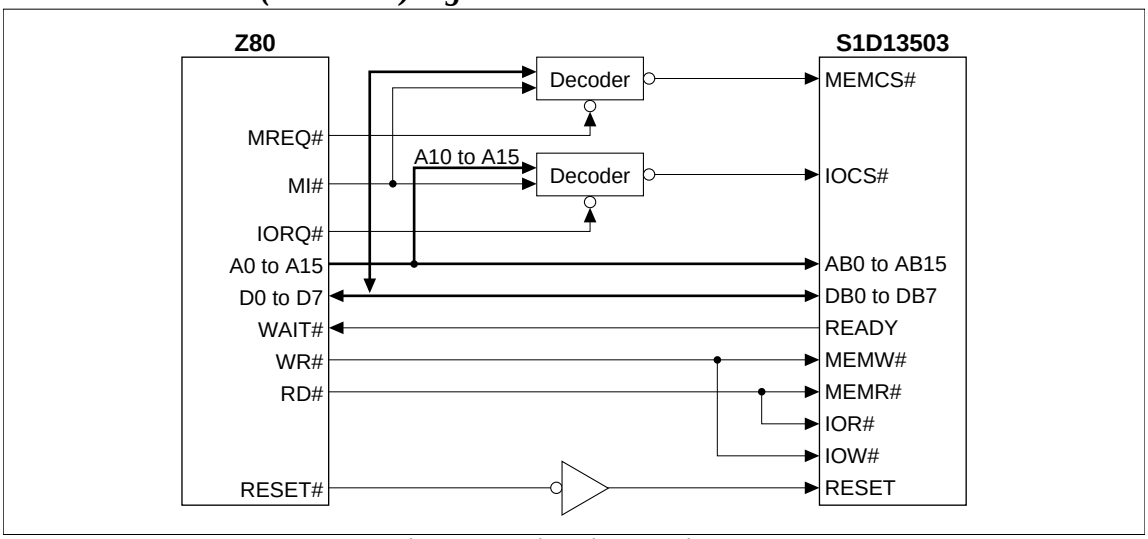



Figure 3-2 8-Bit Mode, Example: Z80
(example implementation only - actual may vary)

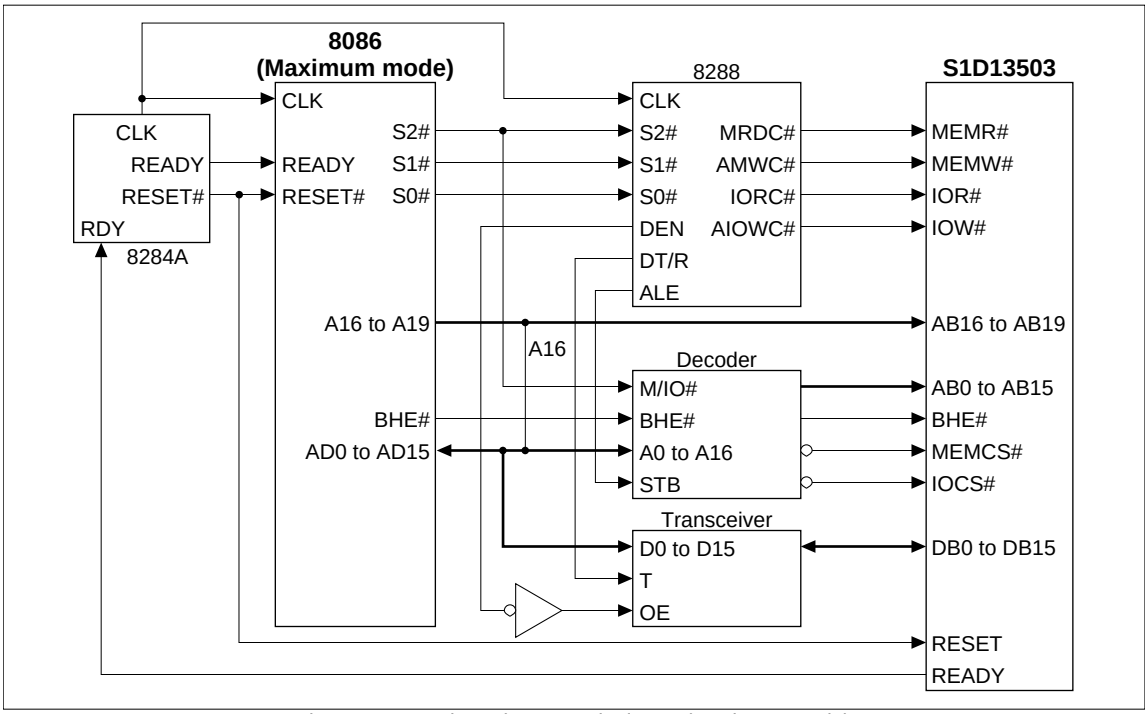



Figure 3-3 16-Bit Mode, Example: i8086 (maximum mode)
(example implementation only - actual may vary)

## *ISA Bus*

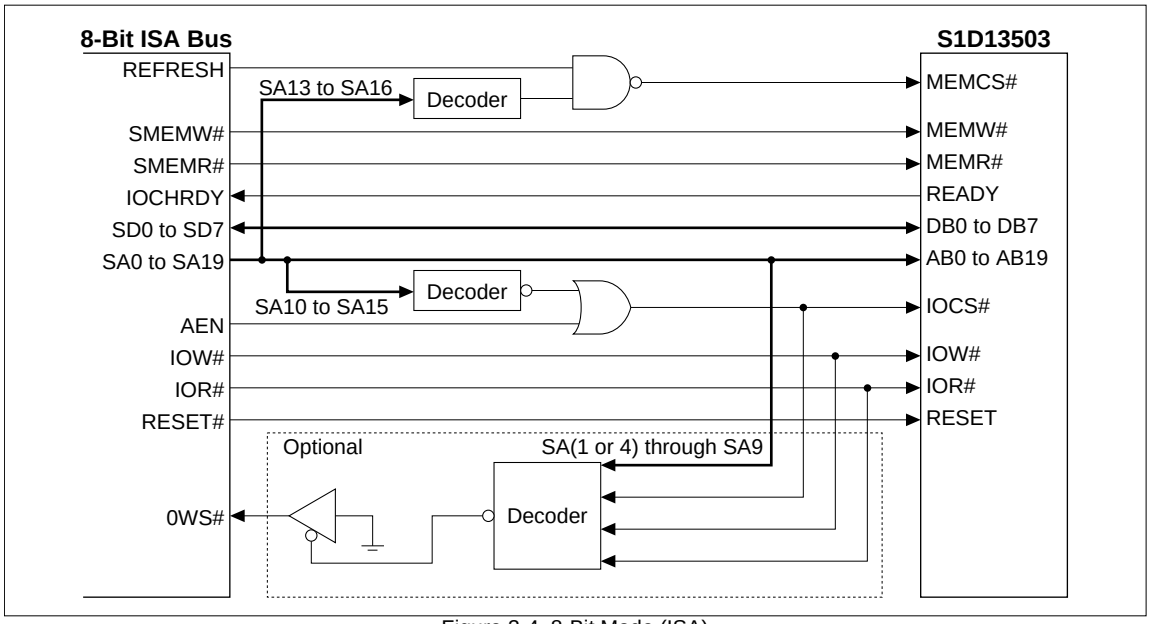



Figure 3-4 8-Bit Mode (ISA)
(example implementation only - actual may vary)

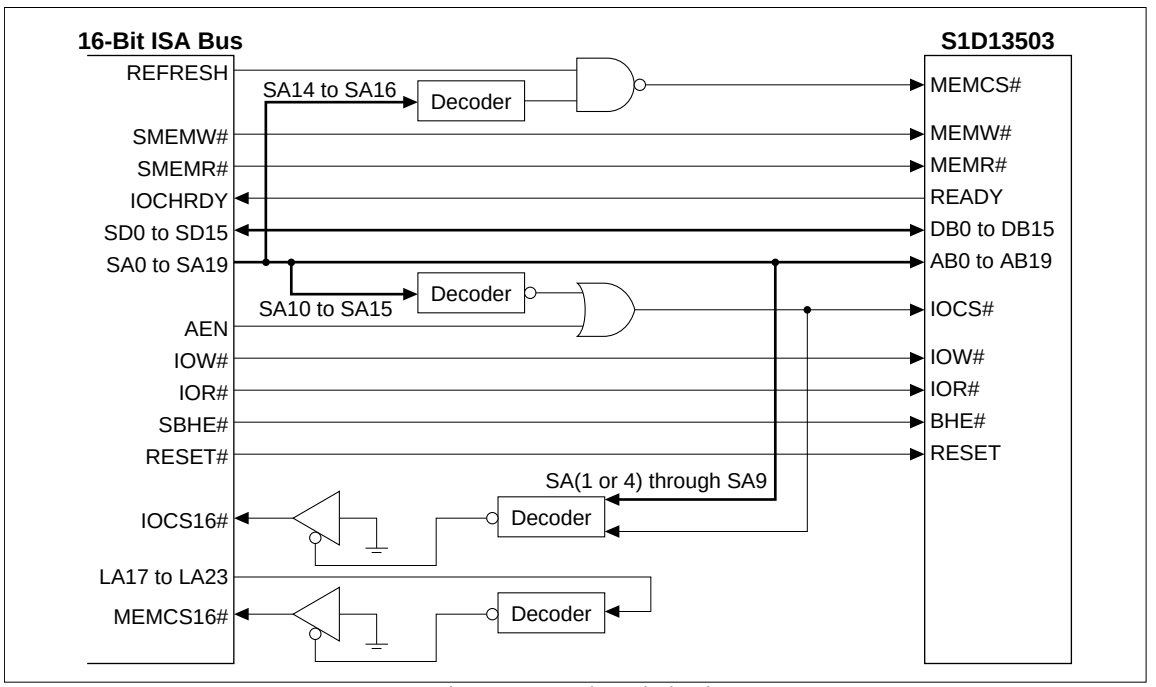



Figure 3-5 16-Bit Mode (ISA)
(example implementation only - actual may vary)

## 3.1 Internal Block Diagram

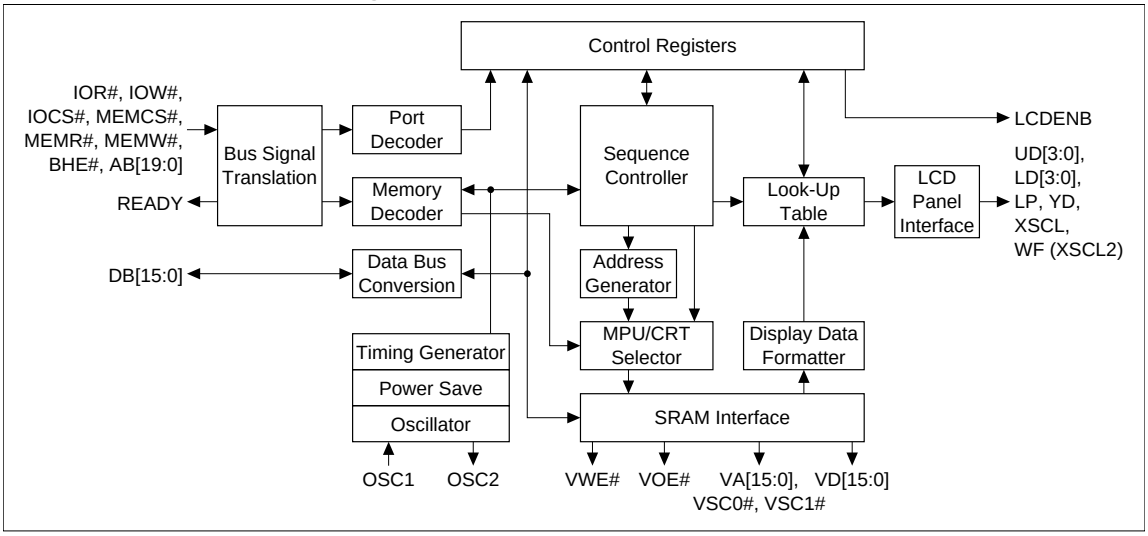



Figure 3-6 Internal Block Diagram

## 3.2 Functional Block Descriptions

### Bus Signal Translation

According to configuration setting VD2, the Bus Signal Translation block translates either MC68000 type MPU signals or MPU controlled by a READY type signals to internal bus interface signals.

### Control Registers

The register block contains the 16 internal control and configuration registers. These registers can be accessed by either direct-mapping or using the built-in internal index register.

### Sequence Controller

The Sequence Controller block generates horizontal and vertical display timings according to the configuration registers settings.

### LCD Panel Interface

The LCD Interface block performs frame rate modulation and output data pattern formatting for both passive monochrome and passive color LCD panels.

### Look-Up Table

The Look-Up Table block contains three $16 \times 4$-bit wide palettes. In gray shade modes, the “green” palette can be configured for the re-mapping of 16 possible shades of gray. In color modes, all three palettes can be configured for the re-mapping of 4096 possible colors. See Section 8.2, Look-Up Table Architecture for details.

### Port Decoder

According to configuration settings VD1, VD12–VD4, IOCS# and address lines AB9–1, the Port Decoder validates a given I/O cycle.

### Memory Decoder

According to configuration settings VD15–VD13, MEMCS# and address lines AB19–17, the Memory Decoder validates a given memory cycle.

### *Data Bus Conversion*

According to configuration setting VD0, the Data Bus Conversion Block maps the external data bus, either 8-bit or 16-bit, into the internal odd and even data bus.

### *Address Generator*

The Address Generator generates display refresh addresses to be used to access display memory.

### *MPU / CRT Selector*

This block grants access to the display memory from either the MPU or the display refresh circuitry.

### *Display Data Formatter*

The Display Data Formatter reads in the display data from the display memory and outputs the correct format for all supported gray shade and color selections.

### *Clock Inputs / Timing*

The Timing block generates the internal master clock (MCLK) according to gray-level/color selected and display memory interface. The master clock (MCLK) can be;
– MCLK = input clock
– MCLK = 1/2 input clock
– MCLK = 1/4 input clock
Refer to Section 9.2, *"SRAM Access Time"* for further details.
Pixel clock = input clock ($f_{OSC}$)

### *SRAM Interface*

This block generates the necessary signals to interface to the Display Memory (SRAM).

# 4 PINOUT DIAGRAM

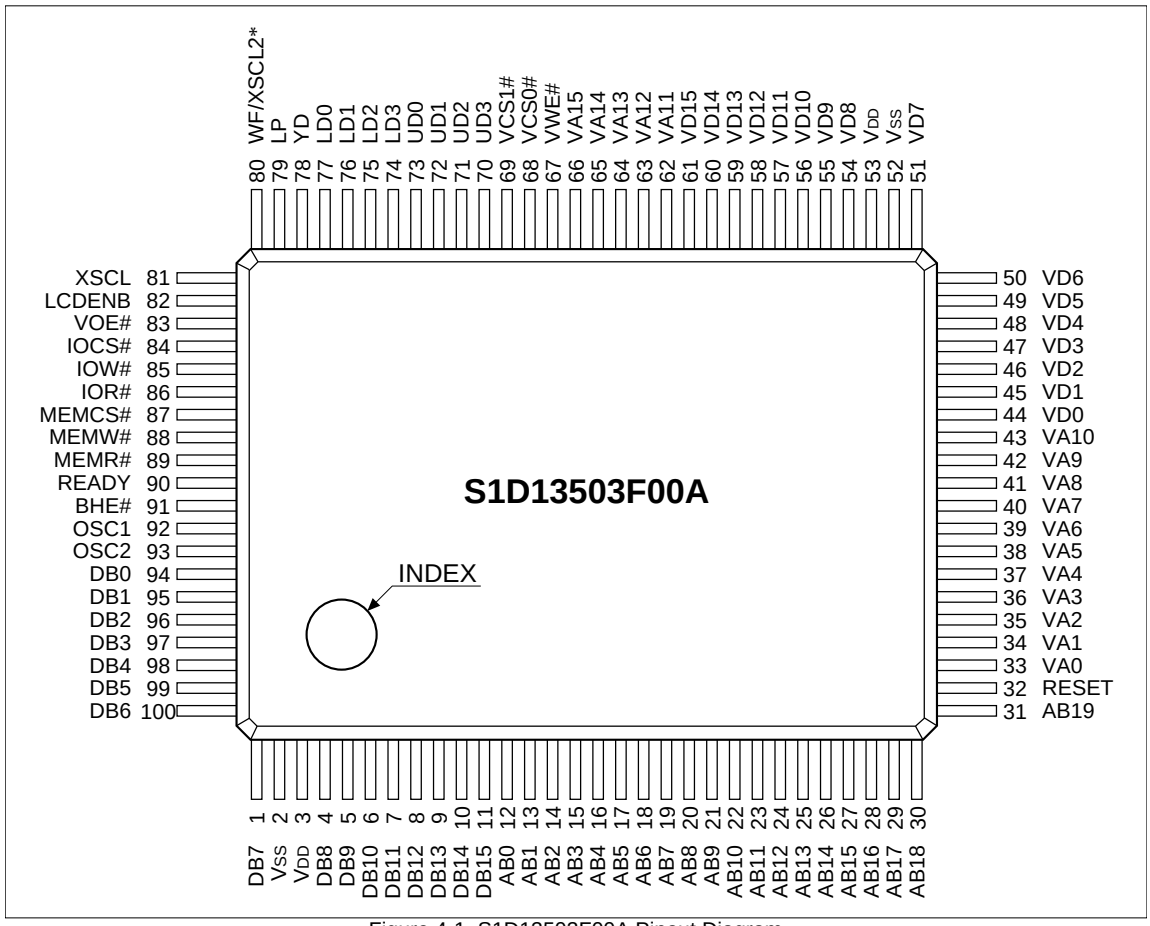



Figure 4-1 S1D13503F00A Pinout Diagram

**Note:** Package type: 100 pin surface mount QFP5-S2

∗ Pin 80 = WF in all display modes except format 1 for 8-bit single color panel.
∗ Pin 80 = XSCL2 in format 1 for 8-bit single color panel.

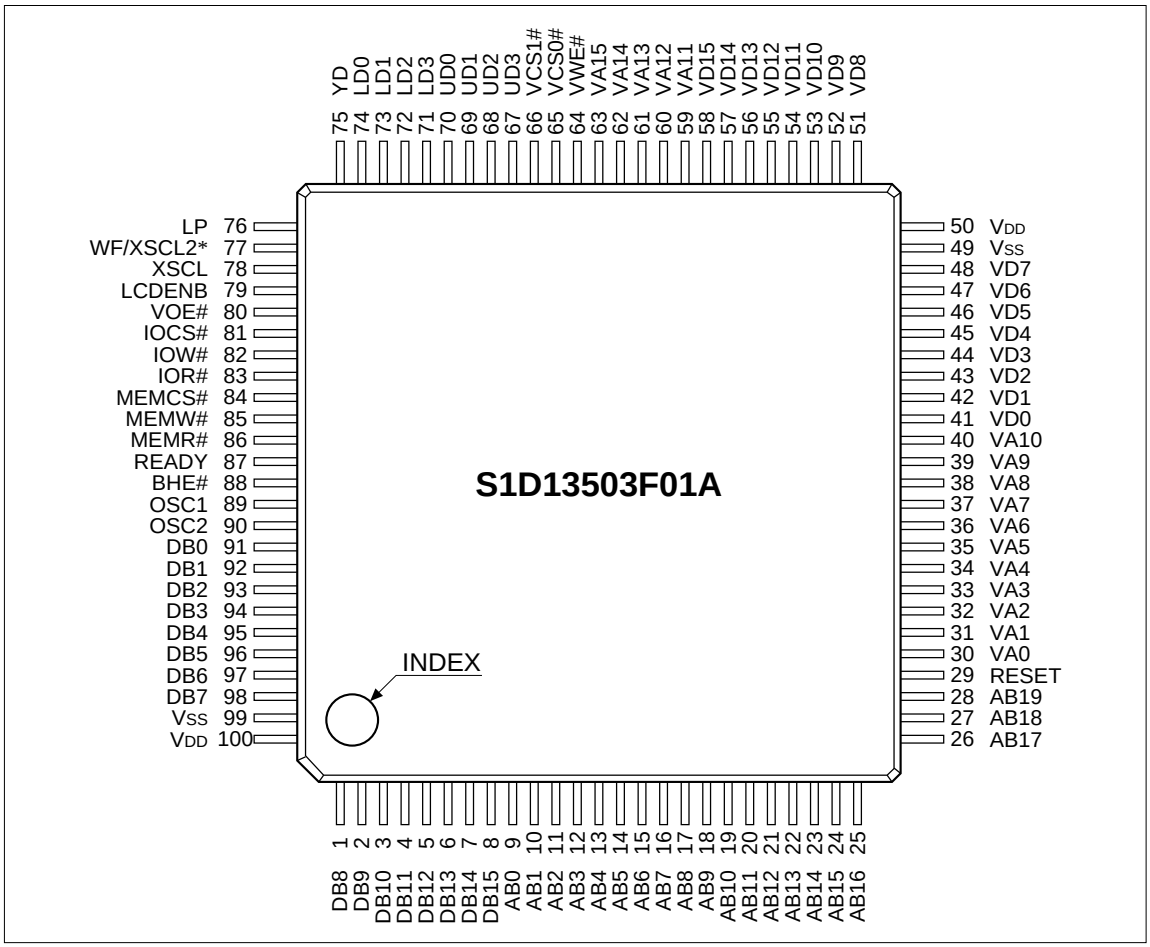
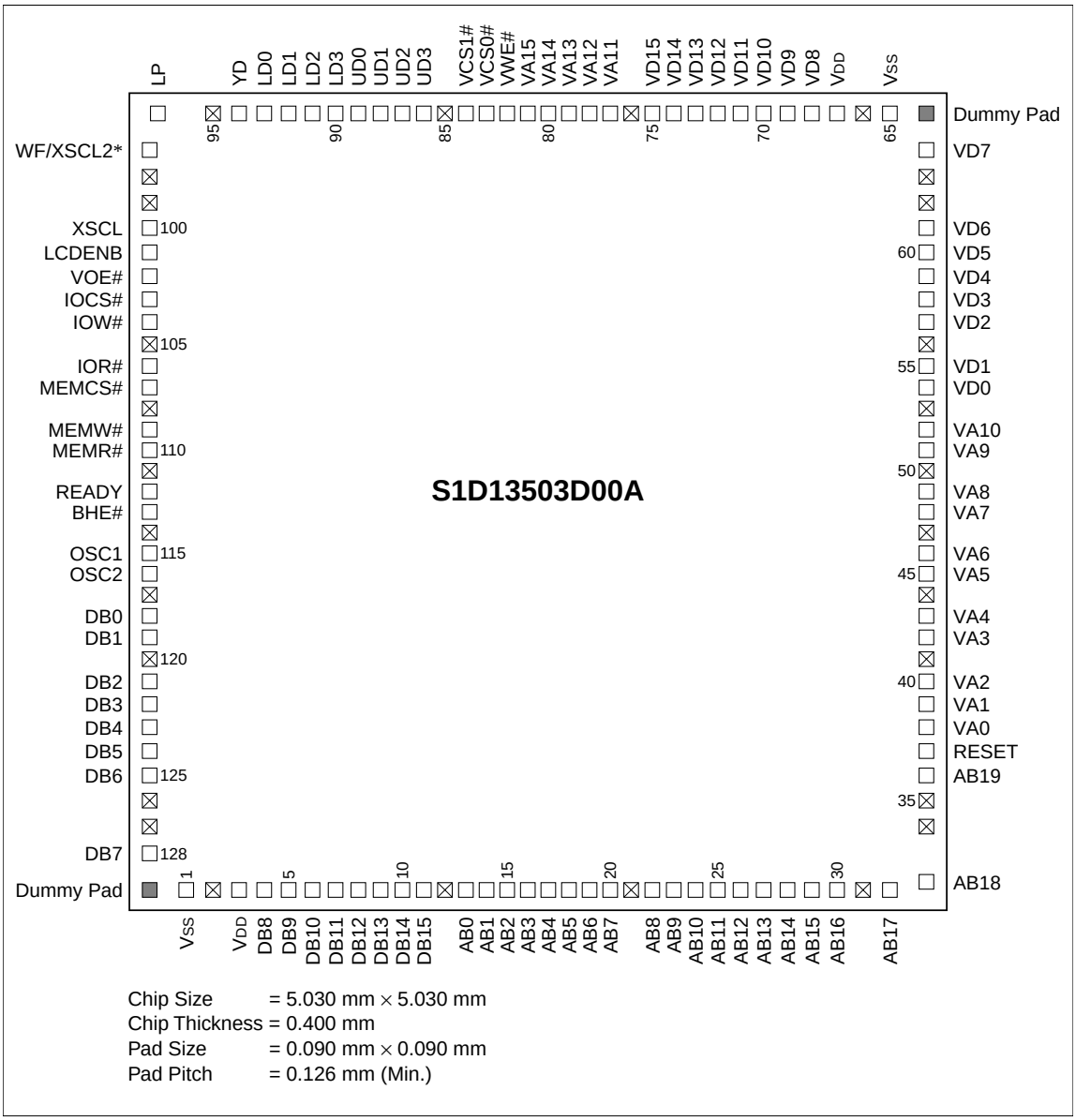



Figure 4-2 S1D13503F01A Pinout Diagram

**Note:** Package type: 100 pin surface mount QFP15-STD

∗ Pin 77 = WF in all display modes except format 1 for 8-bit single color panel.
∗ Pin 77 = XSCL2 in format 1 for 8-bit single color panel.




Chip Size = 5.030 mm × 5.030 mm
Chip Thickness = 0.400 mm
Pad Size = 0.090 mm × 0.090 mm
Pad Pitch = 0.126 mm (Min.)

Figure 4-3 S1D13503D00A Pad Diagram

**Note:** * Pad 97 = WF in all display modes except format 1 for 8-bit single color panel.
* Pad 97 = XSCL2 in format 1 for 8-bit single color panel.

## PAD Coordinates

(Unit: mm)

| Pad No. | Pad Name | Pad Center Coordinate | |
|---|---|---|---|
| | | X | Y |
| 1 | VSS | -2.165 | -2.390 |
| 2 | – | -2.000 | -2.390 |
| 3 | VDD | -1.840 | -2.390 |
| 4 | DB8 | -1.685 | -2.390 |
| 5 | DB9 | -1.535 | -2.390 |
| 6 | DB10 | -1.388 | -2.390 |
| 7 | DB11 | -1.246 | -2.390 |
| 8 | DB12 | -1.106 | -2.390 |
| 9 | DB13 | -0.969 | -2.390 |
| 10 | DB14 | -0.835 | -2.390 |
| 11 | DB15 | -0.703 | -2.390 |
| 12 | – | -0.573 | -2.390 |
| 13 | AB0 | -0.444 | -2.390 |
| 14 | AB1 | -0.317 | -2.390 |
| 15 | AB2 | -0.190 | -2.390 |
| 16 | AB3 | -0.063 | -2.390 |
| 17 | AB4 | 0.063 | -2.390 |
| 18 | AB5 | 0.190 | -2.390 |
| 19 | AB6 | 0.317 | -2.390 |
| 20 | AB7 | 0.444 | -2.390 |
| 21 | – | 0.573 | -2.390 |
| 22 | AB8 | 0.703 | -2.390 |
| 23 | AB9 | 0.835 | -2.390 |
| 24 | AB10 | 0.969 | -2.390 |
| 25 | AB11 | 1.106 | -2.390 |
| 26 | AB12 | 1.246 | -2.390 |
| 27 | AB13 | 1.388 | -2.390 |
| 28 | AB14 | 1.535 | -2.390 |
| 29 | AB15 | 1.685 | -2.390 |
| 30 | AB16 | 1.840 | -2.390 |
| 31 | – | 2.000 | -2.390 |
| 32 | AB17 | 2.165 | -2.390 |
| 33 | AB18 | 2.390 | -2.340 |
| 34 | – | 2.390 | -2.000 |
| 35 | – | 2.390 | -1.840 |
| 36 | AB19 | 2.390 | -1.685 |
| 37 | RESET | 2.390 | -1.535 |
| 38 | VA0 | 2.390 | -1.388 |
| 39 | VA1 | 2.390 | -1.246 |
| 40 | VA2 | 2.390 | -1.106 |
| 41 | – | 2.390 | -0.969 |
| 42 | VA3 | 2.390 | -0.835 |
| 43 | VA4 | 2.390 | -0.703 |
| 44 | – | 2.390 | -0.573 |
| 45 | VA5 | 2.390 | -0.444 |
| 46 | VA6 | 2.390 | -0.317 |
| 47 | – | 2.390 | -0.190 |
| 48 | VA7 | 2.390 | -0.063 |
| 49 | VA8 | 2.390 | 0.063 |
| 50 | – | 2.390 | 0.190 |
| 51 | VA9 | 2.390 | 0.317 |
| 52 | VA10 | 2.390 | 0.444 |
| 53 | – | 2.390 | 0.573 |
| 54 | VD0 | 2.390 | 0.703 |
| 55 | VD1 | 2.390 | 0.835 |
| 56 | – | 2.390 | 0.969 |
| 57 | VD2 | 2.390 | 1.106 |
| 58 | VD3 | 2.390 | 1.246 |
| 59 | VD4 | 2.390 | 1.388 |
| 60 | VD5 | 2.390 | 1.535 |
| 61 | VD6 | 2.390 | 1.685 |
| 62 | – | 2.390 | 1.840 |
| 63 | – | 2.390 | 2.000 |
| 64 | VD7 | 2.390 | 2.165 |
| 65 | VSS | 2.165 | 2.390 |

| Pad No. | Pad Name | Pad Center Coordinate | |
|---|---|---|---|
| | | X | Y |
| 66 | – | 2.000 | 2.390 |
| 67 | VDD | 1.840 | 2.390 |
| 68 | VD8 | 1.685 | 2.390 |
| 69 | VD9 | 1.535 | 2.390 |
| 70 | VD10 | 1.388 | 2.390 |
| 71 | VD11 | 1.246 | 2.390 |
| 72 | VD12 | 1.106 | 2.390 |
| 73 | VD13 | 0.969 | 2.390 |
| 74 | VD14 | 0.835 | 2.390 |
| 75 | VD15 | 0.703 | 2.390 |
| 76 | – | 0.573 | 2.390 |
| 77 | VA11 | 0.444 | 2.390 |
| 78 | VA12 | 0.317 | 2.390 |
| 79 | VA13 | 0.190 | 2.390 |
| 80 | VA14 | 0.063 | 2.390 |
| 81 | VA15 | -0.063 | 2.390 |
| 82 | VWE# | -0.190 | 2.390 |
| 83 | VCS0# | -0.317 | 2.390 |
| 84 | VCS1# | -0.444 | 2.390 |
| 85 | – | -0.573 | 2.390 |
| 86 | UD3 | -0.703 | 2.390 |
| 87 | UD2 | -0.835 | 2.390 |
| 88 | UD1 | -0.969 | 2.390 |
| 89 | UD0 | -1.106 | 2.390 |
| 90 | LD3 | -1.246 | 2.390 |
| 91 | LD2 | -1.388 | 2.390 |
| 92 | LD1 | -1.535 | 2.390 |
| 93 | LD0 | -1.685 | 2.390 |
| 94 | YD | -1.840 | 2.390 |
| 95 | – | -2.000 | 2.390 |
| 96 | LP | -2.340 | 2.390 |
| 97 | WF/XSCL2 | -2.390 | 2.165 |
| 98 | – | -2.390 | 2.000 |
| 99 | – | -2.390 | 1.840 |
| 100 | XSCL | -2.390 | 1.685 |
| 101 | LCDENB | -2.390 | 1.535 |
| 102 | VOE# | -2.390 | 1.388 |
| 103 | IOCS# | -2.390 | 1.246 |
| 104 | IOW# | -2.390 | 1.106 |
| 105 | – | -2.390 | 0.969 |
| 106 | IOR# | -2.390 | 0.835 |
| 107 | MEMCS# | -2.390 | 0.703 |
| 108 | – | -2.390 | 0.573 |
| 109 | MEMW# | -2.390 | 0.444 |
| 110 | MEMR# | -2.390 | 0.317 |
| 111 | – | -2.390 | 0.190 |
| 112 | READY | -2.390 | 0.063 |
| 113 | BHE# | -2.390 | -0.063 |
| 114 | – | -2.390 | -0.190 |
| 115 | OSC1 | -2.390 | -0.317 |
| 116 | OSC2 | -2.390 | -0.444 |
| 117 | – | -2.390 | -0.573 |
| 118 | DB0 | -2.390 | -0.703 |
| 119 | DB1 | -2.390 | -0.835 |
| 120 | – | -2.390 | -0.969 |
| 121 | DB2 | -2.390 | -1.106 |
| 122 | DB3 | -2.390 | -1.246 |
| 123 | DB4 | -2.390 | -1.388 |
| 124 | DB5 | -2.390 | -1.535 |
| 125 | DB6 | -2.390 | -1.685 |
| 126 | – | -2.390 | -1.840 |
| 127 | – | -2.390 | -2.000 |
| 128 | DB7 | -2.390 | -2.165 |
| 129 | Dummy Pad | 2.390 | 2.390 |
| 130 | Dummy Pad | -2.390 | -2.390 |

# 5 PIN DESCRIPTION

## 5.1 Description

**Key:**

**I** = Input
**O** = Output
**I/O** = Bidirectional (Input/Output)
**P** = Power pin
**COx** = CMOS level output driver, x denotes driver type (see Table 6-4, “Output Specifications,” on page 17)
**COxS** = CMOS level output driver with slew rate control for noise reduction, x denotes driver type (see Table 6-4, “Output Specifications,” on page 17)
**TSx** = Tri-state CMOS level output driver, x denotes driver type (see Table 6-4, “Output Specifications,” on page 17)
**TSxD2** = Tri-state CMOS level output driver with pull down resistor (typical values of 100 kΩ/200 kΩ at 5 V/3.0 V respectively), x denotes driver type (see Table 6-4, “Output Specifications,” on page 17)
**TTL** = TTL level input
**TTLS** = TTL level input with hysteresis

Table 5-1 Bus Interface

| Pin Name | Type | F00A Pin No. | F01A Pin No. | D00A Pad No. | Driver | Description |
|---|---|---|---|---|---|---|
| DB0–DB15 | I/O | 94–100, 1, 4 –11 | 91–98, 1–8 | 118–119, 121–125, 128, 4–11 | TS2 | These pins are connected to the system data bus. In 8-bit bus mode, DB8–DB15 must be tied to VDD. |
| AB0 | I | 12 | 9 | 13 | TTLS | In MC68000 MPU interface, this pin is connected to the Upper Data Strobe (UDS#) pin of MC68000. In other MPU/Bus interfaces, this pin is connected to the system address bus. |
| AB1–AB19 | I | 13–31 | 10–28 | 14–20, 22–30, 32–33, 36 | TTL | These pins are connected to the system address bus. |
| BHE# | I | 91 | 88 | 113 | TTLS | In MC68000 MPU interface, this pin is connected to the Lower Data Strobe (LDS#) pin of MC68000. In other MPU/Bus interfaces, this pin is the Byte High Enable input for use with 16-bit system. In 8-bit bus mode tie the BHE# input to VDD. |
| IOCS# | I | 84 | 81 | 103 | TTLS | Active low input to select one of sixteen internal registers. |
| IOW# | I | 85 | 82 | 104 | TTLS | In MC68000 MPU interface, this pin is connected to the R/W# pin of MC68000. This input pin defines whether the data transfer is a read (active high) or write (active low) cycle. In other MPU/Bus interfaces, this is the active low input to write data into an internal register. |
| IOR# | I | 86 | 83 | 106 | TTLS | In MC68000 MPU interface, this pin is connected to the AS# pin of MC68000. This input pin indicates a valid address is available on the address bus. In other MPU/Bus interfaces, this is the active low input to read data from an internal register. |
| MEMCS# | I | 87 | 84 | 107 | TTLS | Active low input to indicate a memory cycle. |
| MEMW# | I | 88 | 85 | 109 | TTLS | Active low input to indicate a memory write cycle. This pin should be tied to VDD in an MC68000 MPU interface. |
| MEMR# | I | 89 | 86 | 110 | TTLS | Active low input to indicate a memory read cycle. This pin should be tied to VDD in an MC68000 MPU interface. |
| READY | O | 90 | 87 | 112 | TS3 | For MC68000 MPU interface, this pin is connected to the DTACK# pin of MC68000 and is driven low when the data transfer is complete. In other MPU/Bus interfaces, this output is driven low to force the system to insert wait states when needed.<br>READY is placed in a high impedance (Hi-Z) state after the transfer is completed. |
| RESET | I | 32 | 29 | 37 | TTLS | Active high input to force all signals to their inactive states. |

Table 5-2 Display Memory Interface

| Pin Name | Type | F00A Pin No. | F01A Pin No. | D00A Pad No. | Driver | Description |
|---|---|---|---|---|---|---|
| VD0–VD15 | I/O | 44–51, 54–61 | 41–48, 51–58 | 54–55, 57–61, 64, 68–75 | TS1D2 | These pins are connected to the display memory data bus. For 16-bit interface, VD0–VD7 are connected to the display memory data bus of even byte addresses and VD8–VD15 are connected to the display memory data bus of odd byte addresses. The output drivers of these pins are placed in a high impedance state when RESET is high. On the falling edge of RESET, the values of VD0–VD15 are latched into the chip to configure various hardware options (see Table 5-6 on page 15). |
| VA0–VA15 | O | 33–43, 62–66 | 30–40, 59–63 | 38–40, 42–43, 45–46, 48–49, 51–52, 77–81 | CO1 | These pins are connected to the display memory address bus. |
| VCS1# | O | 69 | 66 | 84 | CO1 | Active low chip-select output to the second or odd byte address SRAM. See Display Memory Interface section for details. |
| VCS0# | O | 68 | 65 | 83 | CO1 | Active low chip-select output to the first or even byte address SRAM. See Display Memory Interface section for details. |
| VWE# | O | 67 | 64 | 82 | CO1 | Active low output used for writing data to the display memory. This pin is connected to the WE# input of the SRAMs. |
| VOE# | O | 83 | 80 | 102 | CO1 | Active low output to enable reading of data from the display memory. This pin is connected to the OE# input of the SRAMs. |

Table 5-3 LCD Interface

| Pin Name | FPDI-1™ Pin Name#1 | Type | F00A Pin No. | F01A Pin No. | D00A Pad No. | Driver | Description |
|---|---|---|---|---|---|---|---|
| UD3–UD0<br>LD3–LD0 | UD3–UD0<br>LD3–LD0 | O | 70–73, 74–77 | 67–70, 71–74 | 86–89, 90–93 | CO3S | Panel display data bus. The data format depends on the specific panel connected. For 4-bit single panels, LD3–LD0 are driven low (0 state). |
| XSCL | FPSHIFT | O | 81 | 78 | 100 | CO3 | Display data shift clock. Data is shifted into the LCD X-drivers on the falling edge of this signal. |
| LP | FPLINE | O | 79 | 76 | 96 | CO3 | Display data latch clock. The falling edge of this signal is used to latch a row of display data in the LCD X-drivers and to turn on the Y driver (row driver). |
| WF/<br>XSCL2 | MOD<br>FPSHIFT2 | O | 80 | 77 | 97 | CO3 | For format 1 of 8-bit single color panels this is the second shift clock.<br>For all other modes, this is the LCD backplane BIAS signal. This output toggles once every frame, or as programmed in AUX[05] bits 7–2. |
| YD | FPFRAME | O | 78 | 75 | 94 | CO3 | Vertical scanning start pulse. A logic '1' on this signal, sampled by the LCD module on the falling edge of LP, is used by the panel Y driver (row driver) to indicate the start of the vertical frame. |
| LCDENB | | O | 82 | 79 | 101 | CO2 | LCD enable signal output. It can be used externally to turn off the panel supply voltage and backlight. |

#1 VESA Flat Panel Display Interface Standard (FPDI-1™)

Table 5-4 Clock Inputs

| Pin Name | Type | F00A Pin No. | F01A Pin No. | D00A Pad No. | Driver | Description |
|---|---|---|---|---|---|---|
| OSC1 | I | 92 | 92 | 115 | * | This pin, along with OSC2, is the 2-terminal crystal interface when using a 2-terminal crystal as the clock input. If an external oscillator is used as a clock source, then this pin is the clock input. |
| OSC2 | O | 93 | 93 | 116 | * | This pin, along with OSC1, is the 2-terminal crystal interface when using a 2-terminal crystal as the clock input. If an external oscillator is used as a clock source this pin should be left unconnected. |

Table 5-5 Power Supply

| Pin Name | Type | F00A Pin No. | F01A Pin No. | D00A Pad No. | Driver | Description |
|---|---|---|---|---|---|---|
| $V_{DD}$ | P | 3, 53 | 50, 100 | 3, 67 | P | Voltage supply. |
| $V_{SS}$ | P | 2, 52 | 49, 99 | 1, 65 | P | Voltage ground. |

## 5.2 Summary of Configuration Options

The S1D13503 requires some configuration information on power-up. This information is provided through the SRAM data lines VD[0...15]. The state of these pins are read on the falling edge of RESET and used to configure the following options:

Table 5-6 Summary of Power On / Reset Options

| Pin Name | Value on this pin at falling edge of RESET is used to configure: (1/0) | |
|---|---|---|
| | 1 | 0 |
| VD0 | 16-bit host bus interface | 8-bit host bus interface |
| VD1 | Use direct-mapping for I/O accesses | Use internal index register for I/O accesses |
| VD2 | MC68000 MPU interface | MPU / Bus interface with memory accesses controlled by a READY (WAIT#) signal |
| VD3 | Swap of high and low data bytes in 16-bit bus interface | No byte swap of high and low data bytes in 16-bit bus interface |
| VD12–VD4 | Select I/O mapping address bits [9:1].<br>These nine bits are latched on power-up and are compared to the MPU address bits [9–1]. A valid I/O cycle combined with a valid address will enable the internal I/O decoder. Therefore, both types of I/O mapping are limited to even address boundaries to determine either the absolute or indexed I/O address of the first register. Note that a "valid I/O cycle" includes IOCS# being toggled low. | |
| VD15–VD13 | Select memory mapping address bits [3:1]<br>These three bits are latched on power-up and are compared to the MPU address bits [19–17]. A valid memory cycle combined with a valid address will enable the internal memory decoder. As only the three most significant bits of the address are compared, the maximum amount of memory supported is 128K bytes. Note that a "valid memory cycle" includes MEMCS# being toggled low.<br>When using 128K-byte memory it must be mapped at an even address such that all 128K bytes is available without a change in state on A17, as this would invalidate the internal compare logic. | |

**Note:** The S1D13503 has internal pull down resistors on these pins and therefore will be pulled down and read on a logic "0" after RESET. If pull up resistors are required refer to Table 6-3, "Input Specifications," on page 16 for pull down resistor values.

**Example:** If an ISA bus (no byte swap) with memory segment "A" and I/O location 300h are used, the corresponding settings of VD15–VD0 would be:

Table 5-7 I/O and Memory Addressing Example

| Pin Name | 8-Bit ISA Bus | | 16-Bit ISA Bus | |
|---|---|---|---|---|
| | Index Register | Direct Mapping | Index Register | Direct Mapping |
| VD0 | 0 | 0 | 1 | 1 |
| VD1 | 0 | 1 | 0 | 1 |
| VD2 | 0 | 0 | 0 | 0 |
| VD3 | 0 | 0 | 0 | 0 |
| VD12–VD4 | 11 0000 000 | 11 0000 xxx | 11 0000 000 | 11 0000 xxx |
| VD15–VD13 | 101 | 101 | 101 | 101 |

Where x = don't care; 1 = connected to pull-up resistor; 0 = no pull-up resistor

# 6 D.C. CHARACTERISTICS

Table 6-1 Absolute Maximum Ratings

| Symbol | Parameter | Rating | Units |
|---|---|---|---|
| $V_{DD}$ | Supply Voltage | -0.3 to +6.0 | V |
| $V_{IN}$ | Input Voltage | -0.3 to $V_{DD}$ + 0.5 | V |
| $V_{OUT}$ | Output Voltage | -0.3 to $V_{DD}$ + 0.5 | V |
| $T_{STG}$ | Storage Temperature | -65 to 150 | °C |
| $T_{SOL}$ | Solder Temperature/Time | 260 for 10 sec. Max. at lead | °C |

Table 6-2 Recommended Operating Conditions

| Symbol | Parameter | Condition | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|---|
| $V_{DD}$ | Supply Voltage | $V_{SS}$ = 0V | 2.7 | 3.0/3.3/5.0 | 5.5 | V |
| $V_{IN}$ | Input Voltage | – | $V_{SS}$ | – | $V_{DD}$ | V |
| $I_{OPR}$ | Operating Current | $f_{OSC}$ = 6MHz<br>256 colors | – | 4.5/5.0/11 | – | mA |
| $T_{OPR}$ | Operating Temperature | – | -40 | 25 | 85 | °C |

Table 6-3 Input Specifications

| Symbol | Parameter | Condition | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Low Level Input Voltage | $V_{DD}$ = 4.5V | – | – | 0.8 | V |
| | | $V_{DD}$ = 3.0V | – | – | 0.4 | |
| | | $V_{DD}$ = 2.7V | – | – | 0.3 | |
| $V_{IH}$ | High Level Input Voltage | $V_{DD}$ = 5.5V | 2.0 | – | – | V |
| | | $V_{DD}$ = 3.6V | 1.3 | – | – | |
| | | $V_{DD}$ = 3.3V | 1.2 | – | – | |
| $V_{T+}$ | Positive-going Threshold | $V_{DD}$ = 5.0V | – | – | 2.4 | V |
| | | $V_{DD}$ = 3.3V | – | – | 1.4 | |
| | | $V_{DD}$ = 3.0V | – | – | 1.3 | |
| $V_{T-}$ | Negative-going Threshold | $V_{DD}$ = 5.0V | 0.6 | – | – | V |
| | | $V_{DD}$ = 3.3V | 0.5 | – | – | |
| | | $V_{DD}$ = 3.0V | 0.4 | – | – | |
| $V_H$ | Hysteresis Voltage | $V_{DD}$ = 5.0V | 0.1 | – | – | V |
| | | $V_{DD}$ = 3.3V | 0.1 | – | – | |
| | | $V_{DD}$ = 3.0V | 0.1 | – | – | |
| $I_{IZ}$ | Input Leakage Current | – | -1 | – | 1 | µA |
| $C_{IN}$ | Input Pin Capacitance | f = 1MHz<br>$V_{DD}$ = 0V | – | – | 12 | pF |
| $R_{PD}$ | Pull Down Resistance | $V_{DD}$ = 5.0V<br>$V_I$ = $V_{DD}$ | 50 | 100 | 200 | kΩ |
| | | $V_{DD}$ = 3.3V<br>$V_I$ = $V_{DD}$ | 90 | 180 | 360 | |
| | | $V_{DD}$ = 3.0V<br>$V_I$ = $V_{DD}$ | 100 | 200 | 400 | |

Table 6-4 Output Specifications

| Symbol | Parameter | Condition | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|---|
| $V_{OL}$ (5.0V) | Low Level Output Voltage<br>Type 1 - TS1D2, CO1<br>Type 2 - TS2<br>Type 3 - TS3, CO3, CO3S | $V_{DD}$ = Min.<br>$I_{OL}$ = 4mA<br>$I_{OL}$ = 8mA<br>$I_{OL}$ = 12mA | – | – | 0.4 | V |
| $V_{OL}$ (3.3V) | Low Level Output Voltage<br>Type 1 - TS1D2, CO1<br>Type 2 - TS2<br>Type 3 - TS3, CO3, CO3S | $V_{DD}$ = Min.<br>$I_{OL}$ = 2mA<br>$I_{OL}$ = 4mA<br>$I_{OL}$ = 6mA | – | – | 0.3 | V |
| $V_{OL}$ (3.0V) | Low Level Output Voltage<br>Type 1 - TS1D2, CO1<br>Type 2 - TS2<br>Type 3 - TS3, CO3, CO3S | $V_{DD}$ = Min.<br>$I_{OL}$ = 1.8mA<br>$I_{OL}$ = 3.5mA<br>$I_{OL}$ = 5mA | – | – | 0.3 | V |
| $V_{OH}$ (5.0V) | High Level Output Voltage<br>Type 1 - TS1D2, CO1<br>Type 2 - TS2<br>Type 3 - TS3, CO3, CO3S | $V_{DD}$ = Min.<br>$I_{OH}$ = -4mA<br>$I_{OH}$ = -8mA<br>$I_{OH}$ = -12mA | $V_{DD}$-0.4 | – | – | V |
| $V_{OH}$ (3.3V) | High Level Output Voltage<br>Type 1 - TS1D2, CO1<br>Type 2 - TS2<br>Type 3 - TS3, CO3, CO3S | $V_{DD}$ = Min.<br>$I_{OH}$ = -2mA<br>$I_{OH}$ = -4mA<br>$I_{OH}$ = -6mA | $V_{DD}$-0.3 | – | – | V |
| $V_{OH}$ (3.0V) | High Level Output Voltage<br>Type 1 - TS1D2, CO1<br>Type 2 - TS2<br>Type 3 - TS3, CO3, CO3S | $V_{DD}$ = Min.<br>$I_{OH}$ = -1.8mA<br>$I_{OH}$ = -3.5mA<br>$I_{OH}$ = -5mA | $V_{DD}$-0.3 | – | – | V |
| $I_{OZ}$ | Output Leakage Current | – | -1 | – | 1 | µA |
| $C_{OUT}$ | Output Pin Capacitance | f = 1MHz<br>$V_{DD}$ = 0V | – | – | 12 | pF |
| $C_{BID}$ | Bidirectional Pin Capacitance | f = 1MHz<br>$V_{DD}$ = 0V | – | – | 12 | pF |

# 7 A.C. CHARACTERISTICS

Conditions: $V_{DD}$ = 3.0V ± 10%, $V_{DD}$ = 3.3V ± 10% or $V_{DD}$ = 5.0V ± 10%, $T_A$ = -40°C to 85°C
$t_{rise}$ and $t_{fall}$ for all inputs must be ≤ 5 nsec (10% ~ 90%)
$C_L$ = 80pF (Bus/MPU Interface)
$C_L$ = 100pF (LCD Panel Interface)
$C_L$ = 20pF (Display Memory Interface)

## 7.1 Bus Interface Timing

### MC68000 Interface Timing

**Note:** All input timing parameters are based on a maximum 16MHz MPU clock.

**IOW# Timing**

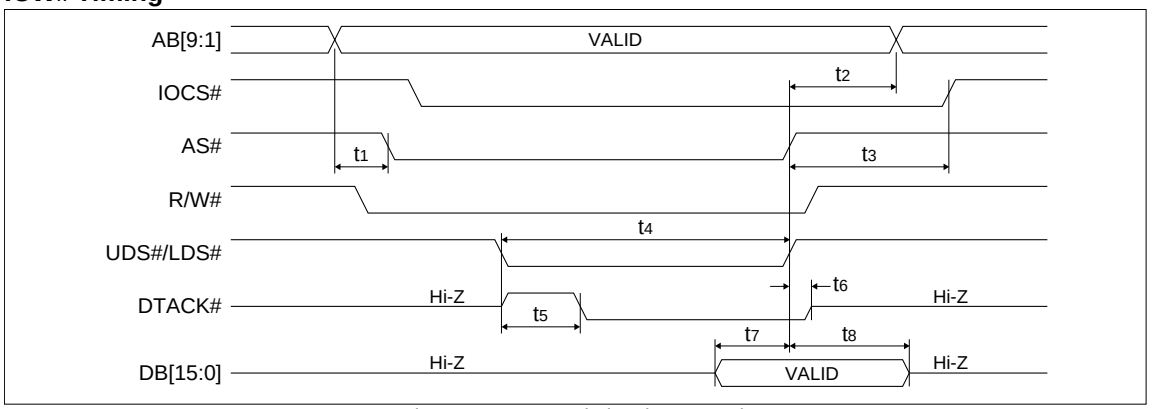



Figure 7-1 IOW# Timing (MC68000)

Table 7-1 IOW# Timing (MC68000)

| Symbol | Parameter | 3V/3.3V | | 5V | | Units |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| $t_1$ | AB[9:1] valid before AS# falling edge | 10 | – | 0 | – | ns |
| $t_2$ | AB[9:1] hold from AS# rising edge | 20 | – | 10 | – | ns |
| $t_3$ | IOCS# hold from AS# rising edge | 0 | – | 0 | – | ns |
| $t_4$ | UDS#/LDS# valid before AS# rising edge | 30 | – | 20 | – | ns |
| $t_5$ | UDS#/LDS# falling edge to DTACK# falling edge | – | 40 | – | 25 | ns |
| $t_6$ | AS# rising edge to DTACK# hi-z delay | – | 40 | – | 25 | ns |
| $t_7$ | DB[15:0] setup to AS# rising edge | 20 | – | 10 | – | ns |
| $t_8$ | DB[15:0] hold from AS# rising edge | 20 | – | 10 | – | ns |

## IOR# Timing

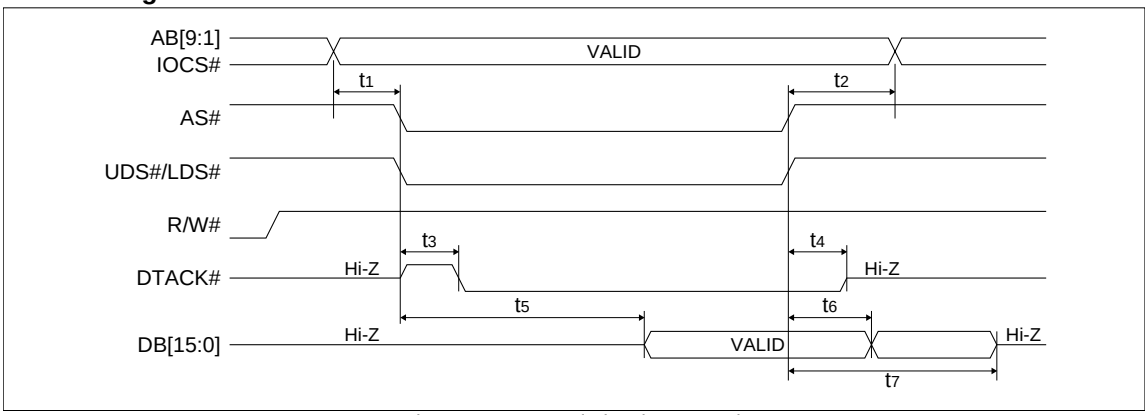



Figure 7-2 IOR# Timing (MC68000)

Table 7-2 IOR# Timing (MC68000)

| Symbol | Parameter | 3V/3.3V | | 5V | | Units |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| t1 | AB[9:1] and IOCS# valid before AS# falling edge | 10 | – | 0 | – | ns |
| t2 | AB[9:1] and IOCS# hold from AS# rising edge | 20 | – | 10 | – | ns |
| t3 | AS# falling edge to DTACK# falling edge | – | 40 | – | 25 | ns |
| t4 | AS# rising edge to DTACK# hi-z delay | – | 40 | – | 25 | ns |
| t5 | AS# falling edge to DB[15:0] valid | – | 60 | – | 40 | ns |
| t6 | DB[15:0] hold from AS# rising edge | – | 20 | – | 15 | ns |
| t7 | AS# rising edge to DB[15:0] hi-z delay | – | 35 | – | 25 | ns |

## MEMW# Timing

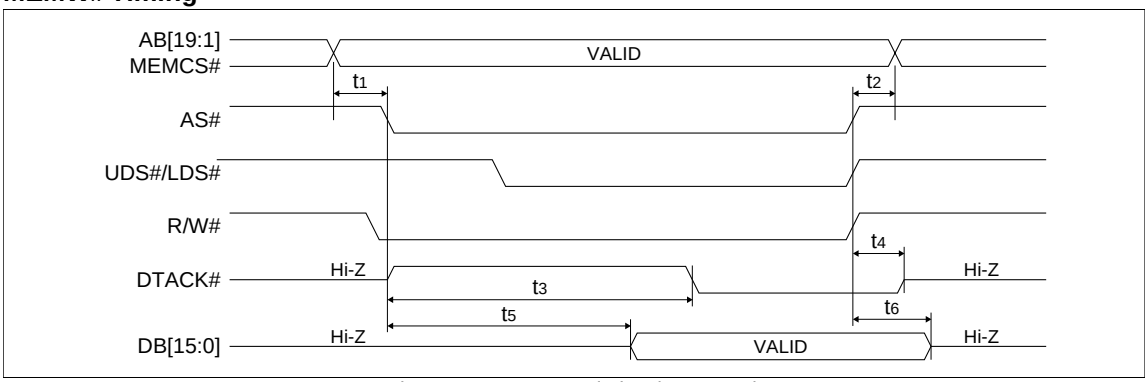



Figure 7-3 MEMW# Timing (MC68000)

Table 7-3 MEMW# Timing (MC68000)

| Symbol | Parameter | 3V/3.3V | | 5V | | Units |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| t1 | AB[19:1] and MEMCS# valid before AS# falling edge | 0 | – | 0 | – | ns |
| t2 | AB[19:1] and MEMCS# hold from AS# rising edge | 0 | – | 0 | – | ns |
| t3 | AS# falling edge to DTACK# falling edge | – | 3.5 * MCLK + 20 | – | 3.5 * MCLK + 10 | ns |
| t4 | AS# rising edge to DTACK hi-z delay | – | 40 | – | 25 | ns |
| t5 | AS# falling edge to DB[15:0] valid | – | MCLK - 40 | – | MCLK - 20 | ns |
| t6 | DB[15:0] hold from AS# rising edge | 0 | – | 0 | – | ns |

Where MCLK = 1/fOSC, or 2/fOSC, or 4/fOSC depending on which display mode the chip is in. (See Section 9.2 and 9.3.)

## MEMR# Timing

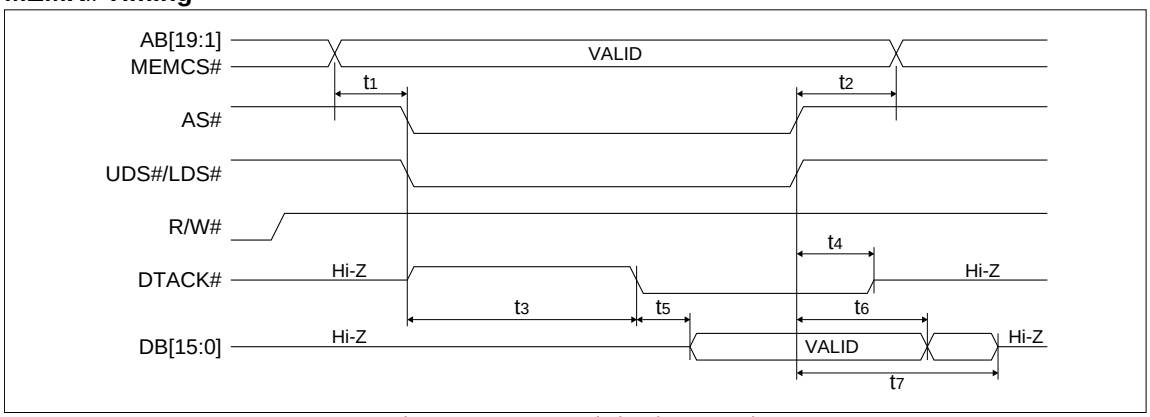



Figure 7-4 MEMR# Timing (MC68000)

Table 7-4 MEMR# Timing (MC68000)

| Symbol | Parameter | 3V/3.3V | | 5V | | Units |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| $t_1$ | AB[19:1] and MEMCS# valid before AS# falling edge | 0 | – | 0 | – | ns |
| $t_2$ | AB[19:1] and MEMCS# hold from AS# rising edge | 0 | – | 0 | – | ns |
| $t_3$ | AS# falling edge to DTACK# falling edge | – | 3.5 * MCLK + 20 | – | 3.5 * MCLK + 10 | ns |
| $t_4$ | AS# rising edge to DTACK# hi-z delay | – | 40 | – | 15 | ns |
| $t_5$ | DTACK# falling edge to DB[15:0] valid | – | 20 | – | 15 | ns |
| $t_6$ | DB[15:0] hold from AS# rising edge | – | 25 | – | 15 | ns |
| $t_7$ | AS# rising edge to DB[15:0] hi-z delay | – | 40 | – | 30 | ns |

Where MCLK = 1/fOSC, or 2/fOSC, or 4/fOSC depending on which display mode the chip is in. (See Section 9.2 and 9.3.)

## *Non-MC68000, MPU/Bus with READY (or WAIT#) Signal*

### IOW# Timing

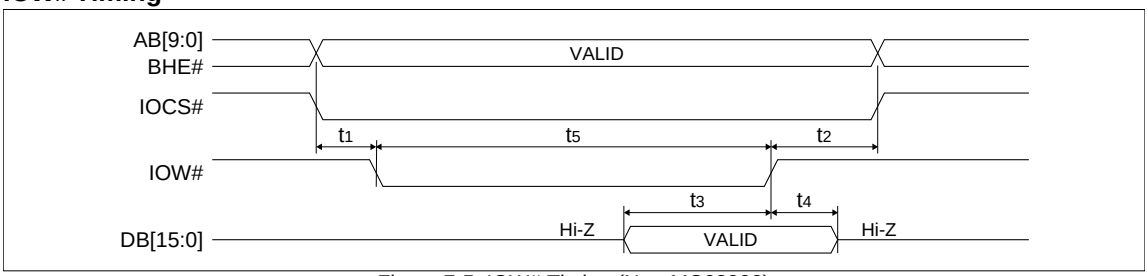



Figure 7-5 IOW# Timing (Non-MC68000)

Table 7-5 IOW# Timing (Non-MC68000)

| Symbol | Parameter | 3V/3.3V | | 5V | | Units |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| $t_1$ | AB[9:0], BHE# and IOCS# valid before IOW# falling edge | 10 | – | 0 | – | ns |
| $t_2$ | AB[9:0], BHE# and IOCS# hold from IOW# rising edge | 20 | – | 10 | – | ns |
| $t_3$ | DB[15:0] setup to IOW# rising edge | 20 | – | 10 | – | ns |
| $t_4$ | DB[15:0] hold from IOW# rising edge | 20 | – | 10 | – | ns |
| $t_5$ | Pulse width of IOW# | 30 | – | 20 | – | ns |

### IOR# Timing

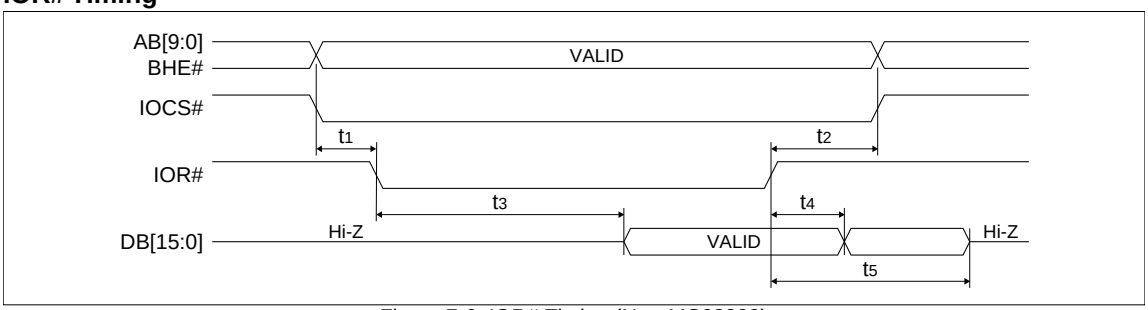



Figure 7-6 IOR# Timing (Non-MC68000)

Table 7-6 IOR# Timing (Non-MC68000)

| Symbol | Parameter | 3V/3.3V | | 5V | | Units |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| $t_1$ | AB[9:0], BHE# and IOCS# valid before IOR# falling edge | 10 | – | 0 | – | ns |
| $t_2$ | AB[9:0], BHE# and IOCS# hold from IOR# rising edge | 20 | – | 10 | – | ns |
| $t_3$ | IOR# falling edge to DB[15:0] valid | – | 60 | – | 40 | ns |
| $t_4$ | DB[15:0] hold from IOR# rising edge | – | 20 | – | 15 | ns |
| $t_5$ | IOR# rising edge to DB[15:0] hi-z delay | – | 35 | – | 25 | ns |

### MEMW# Timing

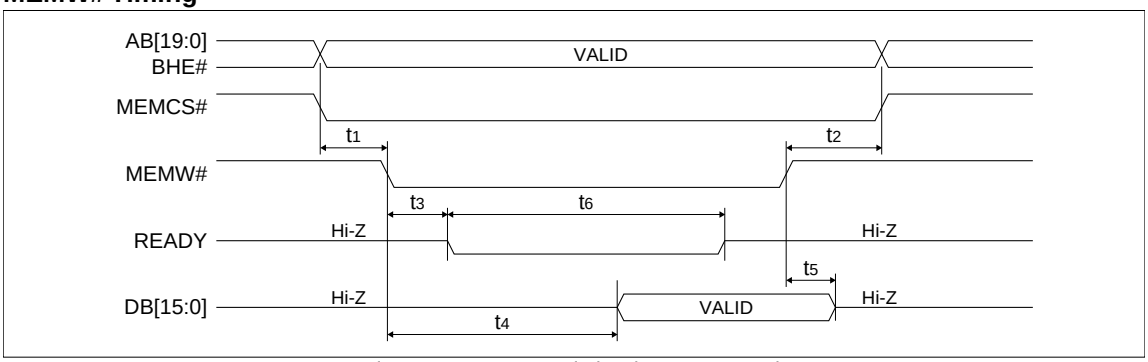



Figure 7-7 MEMW# Timing (Non-MC68000)

Table 7-7 MEMW# Timing (Non-MC68000)

| Symbol | Parameter | 3V/3.3V | | 5V | | Units |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| $t_1$ | AB[19:0], BHE# and MEMCS# valid before MEMW# falling edge | 0 | – | 0 | – | ns |
| $t_2$ | AB[19:0], BHE# and MEMCS# hold from MEMW# rising edge | 0 | – | 0 | – | ns |
| $t_3$ | MEMW# falling edge to READY falling edge | – | 30 | – | 20 | ns |
| $t_4$ | MEMW# falling edge to DB[15:0] valid | – | MCLK - 40 | – | MCLK - 20 | ns |
| $t_5$ | DB[15:0] hold from MEMW# rising edge | 0 | | 0 | – | ns |
| $t_6$ | READY negated pulse width | – | 3.5 * MCLK + 20 | – | 3.5 * MCLK + 10 | ns |

Where MCLK = 1/fOSC, or 2/fOSC, or 4/fOSC depending on which display mode the chip is in. (See Section 9.2 and 9.3.)

## MEMR# Timing

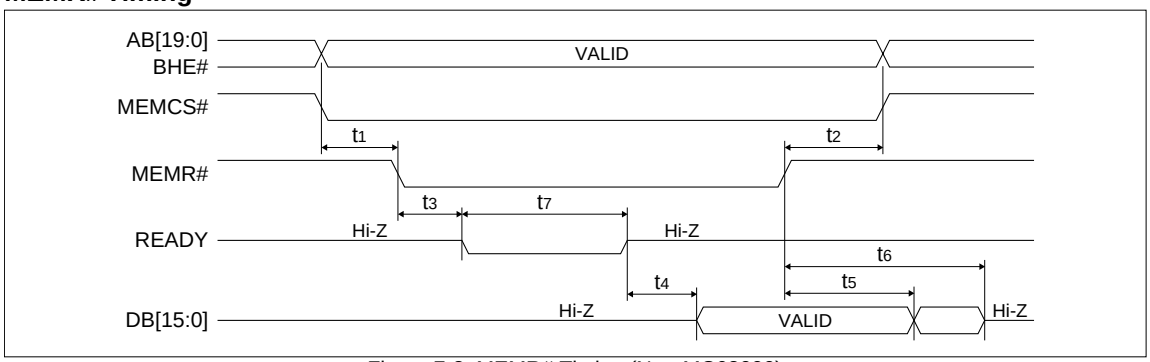



Figure 7-8 MEMR# Timing (Non-MC68000)

Table 7-8 MEMR# Timing (Non-MC68000)

| Symbol | Parameter | 3V/3.3V | | 5V | | |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | Units |
| $t_1$ | AB[19:0], BHE# and MEMCS# valid before MEMR# falling edge | 0 | – | 0 | – | ns |
| $t_2$ | AB[19:0], BHE# and MEMCS# hold from MEMR# rising edge | 0 | – | 0 | – | ns |
| $t_3$ | MEMR# falling edge to READY falling edge | – | 30 | – | 20 | ns |
| $t_4$ | READY rising edge to DB[15:0] valid | – | 15 | – | 10 | ns |
| $t_5$ | DB[15:0] hold from MEMR# rising edge | – | 20 | – | 10 | ns |
| $t_6$ | MEMR# rising edge to DB[15:0] hi-z delay | – | 30 | – | 20 | ns |
| $t_7$ | READY negated pulse width | – | 3.5 * MCLK + 20 | – | 3.5 * MCLK + 10 | ns |

Where MCLK = 1/fOSC, or 2/fOSC, or 4/fOSC depending on which display mode the chip is in. (See Section 9.2 and 9.3.)

## 7.2 Clock Input Requirements

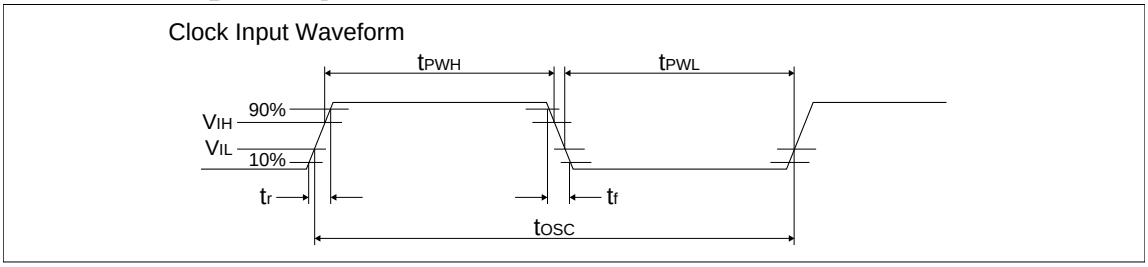



Figure 7-9 Clock Input Requirements

Table 7-9 Clock Input Requirements

| Symbol | Parameter | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|
| $t_{OSC}$ | Input clock period (CLKI) | 40 | – | – | ns |
| $t_{PWH}$ | Input clock pulse width high (CLKI) | 40% | – | 60% | $t_{OSC}$ |
| $t_{PWL}$ | Input clock pulse width low (CLKI) | 40% | – | 60% | $t_{OSC}$ |
| $t_f$ | Input clock fall time (10%–90%) | – | 5 | – | ns |
| $t_r$ | Input clock rise time (10%–90%) | – | 5 | – | ns |

### *Recommended Clock Input*

The nominal frequency must be calculated based on the formulas found in *"Frame Rate Calculation"* on page 57.

The crystal oscillator must be "fundamental mode" and have the following recommended RC load values:
$R_L = 2M\Omega \pm 5\%$
$C_L = 6.8pF$

The figure below demonstrates both a crystal interface and an oscillator interface to the S1D13503.

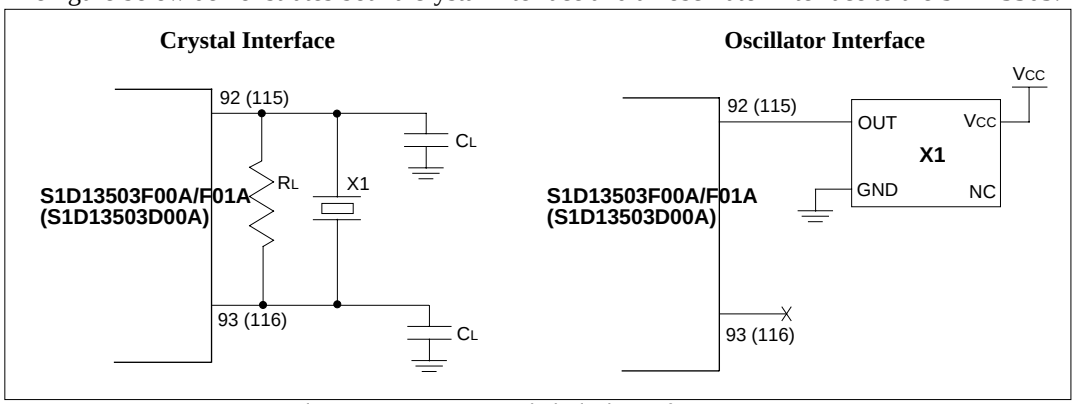



Figure 7-10 Recommended Clock Interface

# 7.3 Display Memory Interface Timing

## Write Data to Display Memory

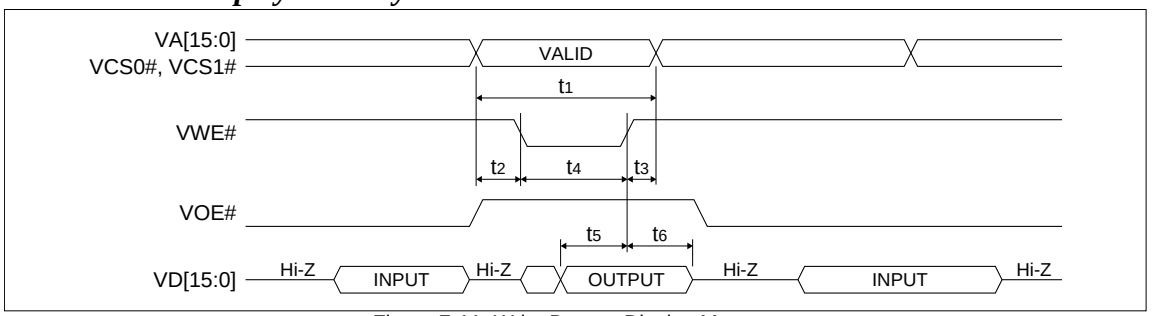



Figure 7-11 Write Data to Display Memory

Table 7-10 Write Data to Display Memory

| Symbol | Parameter | 3V/3.3V | | 5V | | Units |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| $t_1$ | Address cycle time | MCLK - 15 | – | MCLK - 10 | – | ns |
| $t_2$ | VA[15:0], VCS0# and VCS1# valid before VWE# falling edge | 0 | – | 0 | – | ns |
| $t_3$ | VA[15:0], VCS0# and VCS1# hold from VWE# rising edge | 0 | – | 0 | – | ns |
| $t_4$ | Pulse width of VWE# | MCLK - 15 | – | MCLK - 10 | – | ns |
| $t_5$ | VD[15:0] setup to VWE# rising edge | MCLK - 20 | – | MCLK - 15 | – | ns |
| $t_6$ | VD[15:0] hold from VWE# rising edge | 0 | – | 0 | – | ns |

Where MCLK = 1/fOSC, or 2/fOSC, or 4/fOSC depending on which display mode the chip is in. (See Section 9.2 and 9.3.)

## Read Data from Display Memory

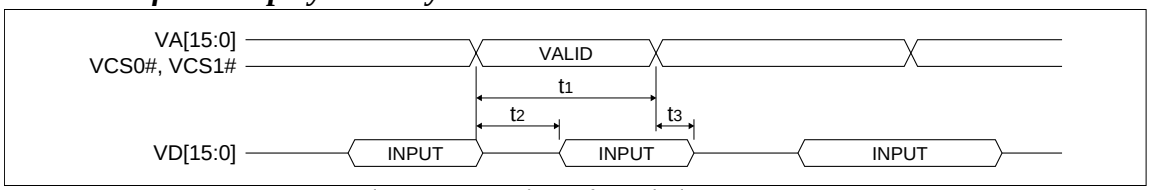



Figure 7-12 Read Data from Display Memory

Table 7-11 Read Data from Display Memory

| Symbol | Parameter | 3V/3.3V | | 5V | | Units |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| $t_1$ | Address cycle time | MCLK - 15 | – | MCLK - 10 | – | ns |
| $t_2$ | VA[15:0], VCS0# and VCS1# access time | – | MCLK - 40 | – | MCLK - 25 | ns |
| $t_3$ | VD[15:0] hold time | 0 | – | 0 | – | ns |

Where MCLK = 1/fOSC, or 2/fOSC, or 4/fOSC depending on which display mode the chip is in. (See Section 9.2 and 9.3.)

## 7.4 LCD Interface

### LCD Interface Timing - 4-Bit Single, 8-Bit Single/Dual Monochrome Panels

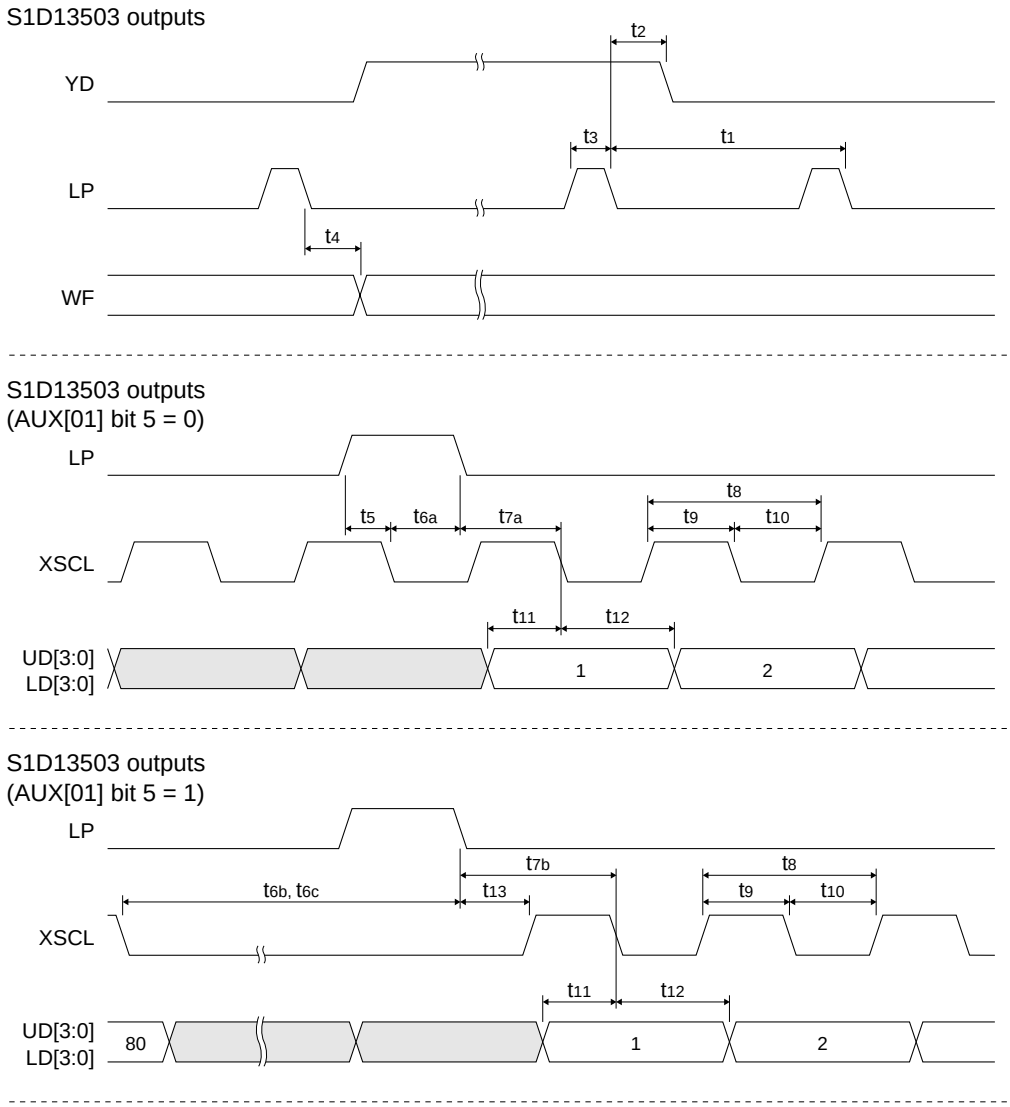



Figure 7-13 LCD Interface Timing - Monochrome Panel

Table 7-12 LCD Interface Timing - 4-Bit Single and 8-Bit Single/Dual Monochrome Panel

| Symbol | Parameter | 4-Bit Single | | 8-Bit Single/Dual | | Units |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| $t_1$ | LP period (single panel mode) | HT + HNDP - 10 | – | HT + HNDP - 10 | – | ns |
| $t_1$ | LP period (dual panel mode) | n/a | – | 2 * (HT + HNDP) - 10 | – | ns |
| $t_2$ | YD hold from LP falling edge (AUX[01] bit 5 = 0) | 8tOSC - 10 | – | 8tOSC - 10 | – | ns |
| $t_2$ | YD hold from LP falling edge (AUX[01] bit 5 = 1) | 13tOSC - 10 | – | 13tOSC - 10 | – | ns |
| $t_3$ | LP pulse width (AUX[01] bit 5 = 0) | 6tOSC - 5 | – | 6tOSC - 5 | – | ns |
| $t_3$ | LP pulse width (AUX[01] bit 5 = 1) | 5tOSC - 5 | – | 5tOSC - 5 | – | ns |
| $t_4$ | WF delay from LP falling edge | 0 | 20 | 0 | 20 | ns |
| $t_5$ | LP setup to XSCL falling edge (AUX[01] bit 5 = 0 and AUX[03] bit 2 = 0) | n/a | – | 2tOSC - 5 | – | ns |
| $t_{6a}$ | LP hold from XSCL falling edge (AUX[01] bit 5 = 0 and AUX[03] bit 2 = 0) | 2tOSC - 5 | – | 4tOSC - 5 | – | ns |
| $t_{6a}$ | LP hold from XSCL falling edge (AUX[01] bit 5 = 0 and AUX[03] bit 2 = 1) | tOSC - 5 | – | 2tOSC - 5 | – | ns |
| $t_{6b}$ | XSCL falling edge to LP falling edge - single panel mode (AUX[01] bit 5 = 1 and AUX[03] bit 2 = 0) | 13tOSC - 5 | – | 15tOSC - 5 | – | ns |
| $t_{6b}$ | XSCL falling edge to LP falling edge - single panel mode (AUX[01] bit 5 = 1 and AUX[03] bit 2 = 1) | 12tOSC - 5 | – | 13tOSC - 5 | – | ns |
| $t_{6c}$ | XSCL falling edge to LP falling edge - dual panel mode (AUX[01] bit 5 = 1 and AUX[03] bit 2 = 0) | n/a | – | 31tOSC - 5 | – | ns |
| $t_{6c}$ | XSCL falling edge to LP falling edge - dual panel mode (AUX[01] bit 5 = 1 and AUX[03] bit 2 = 1) | n/a | – | 29tOSC - 5 | – | ns |
| $t_{7a}$ | LP falling edge to XSCL falling edge (AUX[01] bit 5 = 0 and AUX[03] bit 2 = 0) | 2tOSC - 5 | – | 4tOSC - 5 | – | ns |
| $t_{7a}$ | LP falling edge to XSCL falling edge (AUX[01] bit 5 = 0 and AUX[03] bit 2 = 1) | tOSC - 5 | – | 2tOSC - 5 | – | ns |
| $t_{7b}$ | LP falling edge to XSCL falling edge (AUX[01] bit 5 = 1 and AUX[03] bit 2 = 0) | 7tOSC - 5 | – | 9tOSC - 5 | – | ns |
| $t_{7b}$ | LP falling edge to XSCL falling edge (AUX[01] bit 5 = 1 and AUX[03] bit 2 = 1) | 6tOSC - 5 | – | 7tOSC - 5 | – | ns |
| $t_8$ | XSCL period (AUX[03] bit 2 = 0) | 4tOSC - 5 | – | 8tOSC - 5 | – | ns |
| $t_8$ | XSCL period (AUX[03] bit 2 = 1) | 2tOSC - 5 | – | 4tOSC - 5 | – | ns |
| $t_9$ | XSCL high width (AUX[03] bit 2 = 0) | 2tOSC - 5 | – | 4tOSC - 5 | – | ns |
| $t_9$ | XSCL high width (AUX[03] bit 2 = 1) | tOSC - 5 | – | 2tOSC - 5 | – | ns |
| $t_{10}$ | XSCL low width (AUX[03] bit 2 = 0) | 2tOSC - 10 | – | 4tOSC - 10 | – | ns |
| $t_{10}$ | XSCL low width (AUX[03] bit 2 = 1) | tOSC - 10 | – | 2tOSC - 10 | – | ns |
| $t_{11}$ | UD[3:0], LD[3:0] setup to XSCL falling edge (AUX[03] bit 2 = 0) | 2tOSC - 10** | – | 4tOSC - 10** | – | ns |
| $t_{11}$ | UD[3:0], LD[3:0] setup to XSCL falling edge (AUX[03] bit 2 = 1) | tOSC - 10** | – | 2tOSC - 10** | – | ns |
| $t_{12}$ | UD[3:0], LD[3:0] hold from XSCL falling edge (AUX[03] bit 2 = 0) | 2tOSC - 10 | – | 4tOSC - 10 | – | ns |
| $t_{12}$ | UD[3:0], LD[3:0] hold from XSCL falling edge (AUX[03] bit 2 = 1) | tOSC - 10 | – | 2tOSC - 10 | – | ns |
| $t_{13}$ | LP falling edge to XSCL rising edge (AUX[01] bit 5 = 1) | 5tOSC - 5 | – | 5tOSC - 5 | – | ns |

Where tOSC = 1/fOSC = input (pixel) clock period,
where HT = (number of horizontal panel pixels) * tOSC,
where HNDP = horizontal non-display period in units of tOSC (see Section 9.3 on page 57 for details).
** -10ns for 5V operation, -24ns for 3.0V and 3.3V operation.

## *LCD Interface Timing - 4-Bit Single Color Panel*

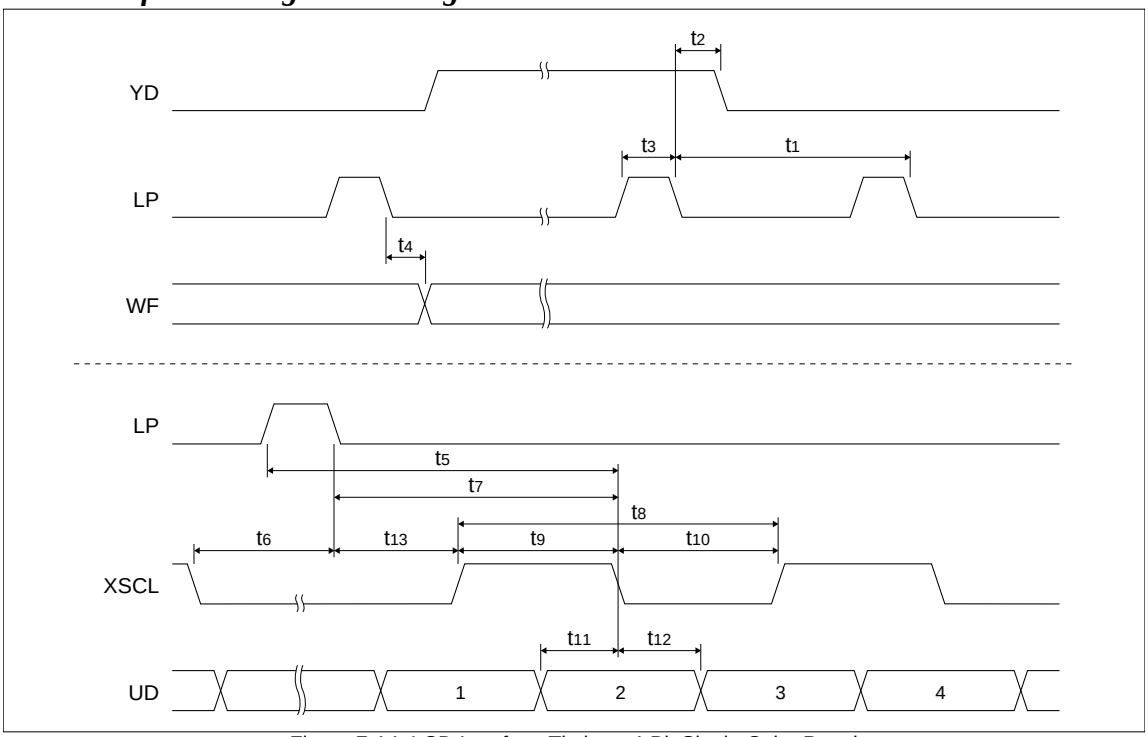



Figure 7-14 LCD Interface Timing - 4-Bit Single Color Panel

Table 7-13 LCD Interface Timing - 4-Bit Single Color Panel

| Symbol | Parameter | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|
| $t_1$ | LP period | HT + HNDP - 10 | – | – | ns |
| $t_2$ | YD hold from LP falling edge | 13$t_{OSC}$ - 10 | – | – | ns |
| $t_3$ | LP pulse width | 5$t_{OSC}$ - 5 | – | – | ns |
| $t_4$ | WF delay from LP falling edge | 0 | – | 20 | ns |
| $t_5$ | LP setup to XSCL falling edge | 19$t_{OSC}$ - 5 | – | – | ns |
| $t_6$ | XSCL falling edge to LP falling edge | 20$t_{OSC}$ - 5 | – | – | ns |
| $t_7$ | LP falling edge to XSCL falling edge | 14$t_{OSC}$ - 5 | – | – | ns |
| $t_8$ | XSCL period | $t_{OSC}$ - 5 | – | – | ns |
| $t_9$ | XSCL high width | 0.5$t_{OSC}$ - 5 | – | – | ns |
| $t_{10}$ | XSCL low width | 0.5$t_{OSC}$ - 5 | – | – | ns |
| $t_{11}$ | UD setup to XSCL falling edge | 0.5$t_{OSC}$ - 10** | – | – | ns |
| $t_{12}$ | UD hold from XSCL falling edge | 0.5$t_{OSC}$ - 10 | – | – | ns |
| $t_{13}$ | LP falling edge to XSCL rising edge | 13.5$t_{OSC}$ - 10 | – | – | ns |

Where $t_{OSC}$ = 1/$f_{OSC}$ = input (pixel) clock period,
where HT = (number of horizontal panel pixels) * $t_{OSC}$,
where HNDP = horizontal non-display period in units of $t_{OSC}$ (see Section 9.3 on page 57 for details).
** 5V operation, for 3.0V and 3.3V operation $t_{11}$ will be 0.5$t_{OSC}$ - 24.

## *LCD Interface Timing - 8-Bit Single Color Panels Format 2 / 8-Bit Dual Color Panels*

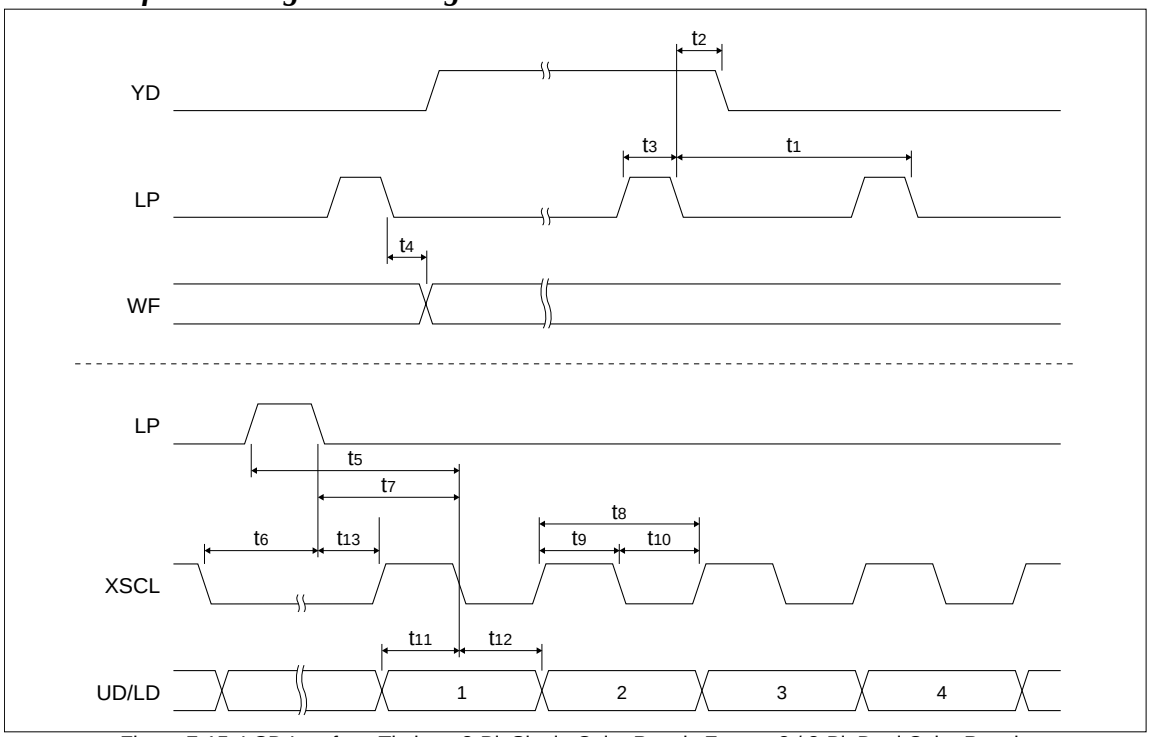



Figure 7-15 LCD Interface Timing - 8-Bit Single Color Panels Format 2 / 8-Bit Dual Color Panels

Table 7-14 LCD Interface Timing - 8-Bit Single Color Panels Format 2 / 8-Bit Dual Color Panels

| Symbol | Parameter | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|
| $t_1$ | LP period (single panel mode) | HT + HNDP - 10 | – | – | ns |
| $t_1$ | LP period (dual panel mode) | 2(HT + HNDP) - 10 | – | – | ns |
| $t_2$ | YD hold from LP falling edge | 13$t_{OSC}$ - 10 | – | – | ns |
| $t_3$ | LP pulse width | 5$t_{OSC}$ - 5 | – | – | ns |
| $t_4$ | WF delay from LP falling edge | 0 | – | 20 | ns |
| $t_5$ | LP setup to XSCL falling edge | 19.5$t_{OSC}$ - 5 | – | – | ns |
| $t_6$ | XSCL falling edge to LP falling edge (single panel mode) | 20$t_{OSC}$ - 5 | – | – | ns |
| $t_6$ | XSCL falling edge to LP falling edge (dual panel mode) | 52$t_{OSC}$ - 5 | – | – | ns |
| $t_7$ | LP falling edge to XSCL falling edge | 14.5$t_{OSC}$ - 5 | – | – | ns |
| $t_8$ | XSCL period | 2.5$t_{OSC}$ - 5 | – | – | ns |
| $t_9$ | XSCL high width | $t_{OSC}$ - 5 | – | – | ns |
| $t_{10}$ | XSCL low width | 1.5$t_{OSC}$ - 5 | – | – | ns |
| $t_{11}$ | UD/LD setup to XSCL falling edge | 1.5$t_{OSC}$ - 10** | – | – | ns |
| $t_{12}$ | UD/LD hold from XSCL falling edge | $t_{OSC}$ - 5 | – | – | ns |
| $t_{13}$ | LP falling edge to XSCL rising edge | 13.5$t_{OSC}$ - 10 | – | – | ns |

Where $t_{OSC} = 1/f_{OSC}$ = input (pixel) clock period,
where HT = (number of horizontal panel pixels) ∗ $t_{OSC}$,
where HNDP = horizontal non-display period in units of $t_{OSC}$ (see Section 9.3 on page 57 for details).
** 5V operation, for 3.0V and 3.3V operation $t_{11}$ will be 1.5$t_{OSC}$ - 24.

## *LCD Interface Timing - 16-Bit Single/Dual Color Panels*

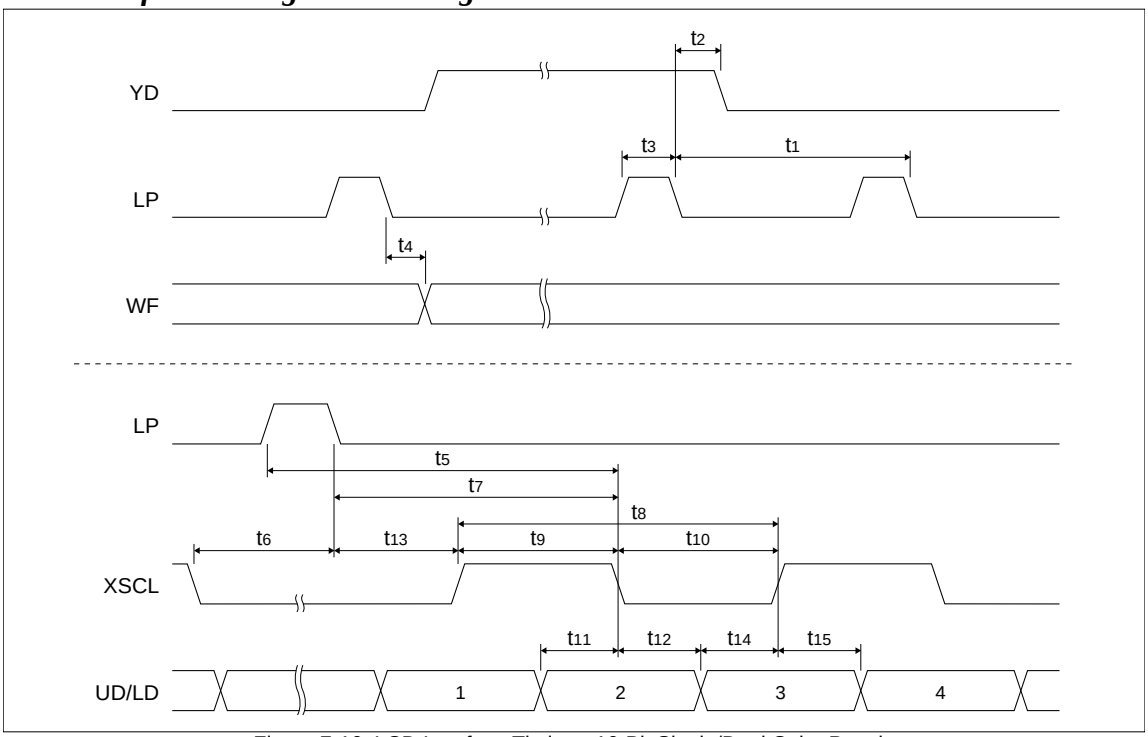



Figure 7-16 LCD Interface Timing - 16-Bit Single/Dual Color Panels

Table 7-15 LCD Interface Timing - 16-Bit Single/Dual Color Panels

| Symbol | Parameter | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|
| $t_1$ | LP period (single panel mode) | HT + HNDP - 10 | – | – | ns |
| $t_1$ | LP period (dual panel mode) | 2(HT + HNDP) - 10 | – | – | ns |
| $t_2$ | YD hold from LP falling edge | 13$t_{OSC}$ - 10 | – | – | ns |
| $t_3$ | LP pulse width | 5$t_{OSC}$ - 5 | – | – | ns |
| $t_4$ | WF delay from LP falling edge | 0 | – | 20 | ns |
| $t_5$ | LP setup to XSCL falling edge | 22$t_{OSC}$ - 5 | – | – | ns |
| $t_6$ | XSCL falling edge to LP falling edge (single panel mode) | 20$t_{OSC}$ - 5 | – | – | ns |
| $t_6$ | XSCL falling edge to LP falling edge (dual panel mode) | 52$t_{OSC}$ - 5 | – | – | ns |
| $t_7$ | LP falling edge to XSCL falling edge | 17$t_{OSC}$ - 5 | – | – | ns |
| $t_8$ | XSCL period | 5$t_{OSC}$ - 5 | – | – | ns |
| $t_9$ | XSCL high width | 2$t_{OSC}$ - 5 | – | – | ns |
| $t_{10}$ | XSCL low width | 3$t_{OSC}$ - 10 | – | – | ns |
| $t_{11}$ | UD/LD setup to XSCL falling edge | 1.5$t_{OSC}$ - 10** | – | – | ns |
| $t_{12}$ | UD/LD hold from XSCL falling edge | $t_{OSC}$ - 5 | – | – | ns |
| $t_{13}$ | LP falling edge to XSCL rising edge | 15$t_{OSC}$ - 10 | – | – | ns |
| $t_{14}$ | UD/LD setup to XSCL rising edge | 1.5$t_{OSC}$ - 10 | – | – | ns |
| $t_{15}$ | UD/LD hold from XSCL rising edge | 0.5$t_{OSC}$ - 5 | – | – | ns |

Where $t_{OSC} = 1/f_{OSC}$ = input (pixel) clock period,
where HT = (number of horizontal panel pixels) $*$ $t_{OSC}$,
where HNDP = horizontal non-display period in units of $t_{OSC}$ (see Section 9.3 on page 57 for details).
** 5V operation, for 3.0V and 3.3V operation $t_{11}$ will be 1.5$t_{OSC}$ - 24.

## *LCD Interface Timing - 8-Bit Single Color Panels Format 1*

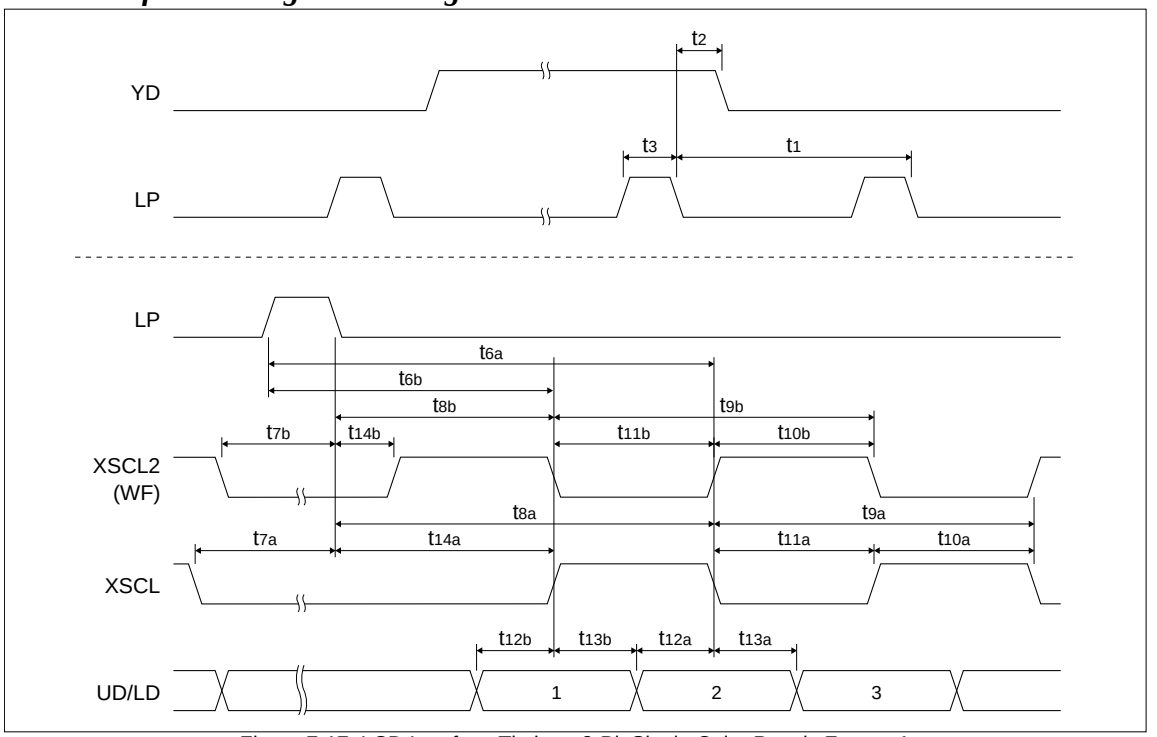



Figure 7-17 LCD Interface Timing - 8-Bit Single Color Panels Format 1

Table 7-16 LCD Interface Timing - 8-Bit Single Color Panels Format 1

| Symbol | Parameter | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|
| $t_1$ | LP period | HT + HNDP - 10 | – | – | ns |
| $t_2$ | YD hold from LP falling edge | 13$t_{OSC}$ - 10 | – | – | ns |
| $t_3$ | LP pulse width | 5$t_{OSC}$ - 5 | – | – | ns |
| $t_{6a}$ | LP setup to XSCL falling edge | 22$t_{OSC}$ - 5 | – | – | ns |
| $t_{6b}$ | LP setup to XSCL2 falling edge | 19.5$t_{OSC}$ - 5 | – | – | ns |
| $t_{7a}$ | XSCL falling edge to LP falling edge | 20$t_{OSC}$ - 5 | – | – | ns |
| $t_{7b}$ | XSCL2 falling edge to LP falling edge | 23.5$t_{OSC}$ - 5 | – | – | ns |
| $t_{8a}$ | LP falling edge to XSCL falling edge | 17$t_{OSC}$ - 5 | – | – | ns |
| $t_{8b}$ | LP falling edge to XSCL2 falling edge | 14.5$t_{OSC}$ - 5 | – | – | ns |
| $t_{9a}$ | XSCL period | 4$t_{OSC}$ - 5 | – | – | ns |
| $t_{9b}$ | XSCL2 period | 4$t_{OSC}$ - 5 | – | – | ns |
| $t_{10a}$ | XSCL high width | $t_{OSC}$ - 5 | – | – | ns |
| $t_{10b}$ | XSCL2 high width | $t_{OSC}$ - 5 | – | – | ns |
| $t_{11a}$ | XSCL low width | 3$t_{OSC}$ - 10 | – | – | ns |
| $t_{11b}$ | XSCL2 low width | 3$t_{OSC}$ - 10 | – | – | ns |
| $t_{12a}$ | UD/LD setup to XSCL falling edge | 1.5$t_{OSC}$ - 10** | – | – | ns |
| $t_{12b}$ | UD/LD setup to XSCL2 falling edge | 1.5$t_{OSC}$ - 10** | – | – | ns |
| $t_{13a}$ | UD/LD hold from XSCL falling edge | $t_{OSC}$ - 5 | – | – | ns |
| $t_{13b}$ | UD/LD hold from XSCL2 falling edge | $t_{OSC}$ - 5 | – | – | ns |
| $t_{14a}$ | LP falling edge to XSCL rising edge | 16$t_{OSC}$ - 10 | – | – | ns |
| $t_{14b}$ | LP falling edge to XSCL2 rising edge | 13.5$t_{OSC}$ - 10 | – | – | ns |

Where $t_{OSC}$ = 1/$f_{OSC}$ = input (pixel) clock period,
where HT = (number of horizontal panel pixels) ∗ $t_{OSC}$,
where HNDP = horizontal non-display period in units of $t_{OSC}$ (see Section 9.3 on page 57 for details).
** 5V operation, for 3.0V and 3.3V operation $t_{12}$ will be 1.5$t_{OSC}$ - 24.

## *LCD Interface Options*

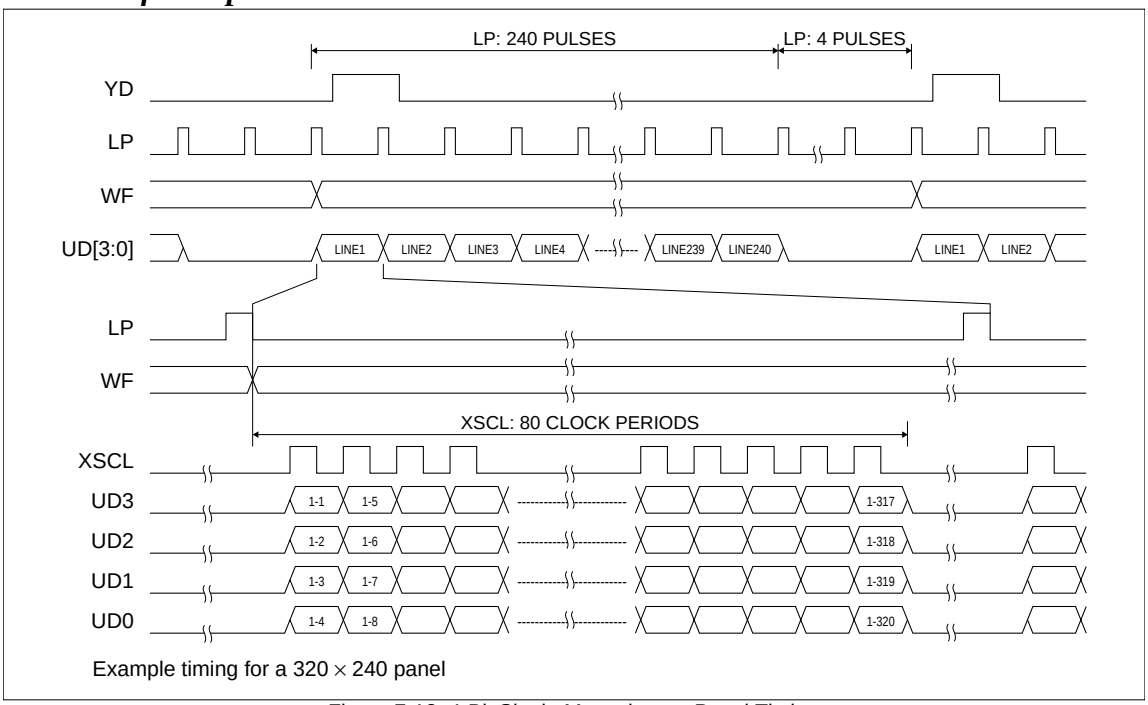



Figure 7-18 4-Bit Single Monochrome Panel Timing

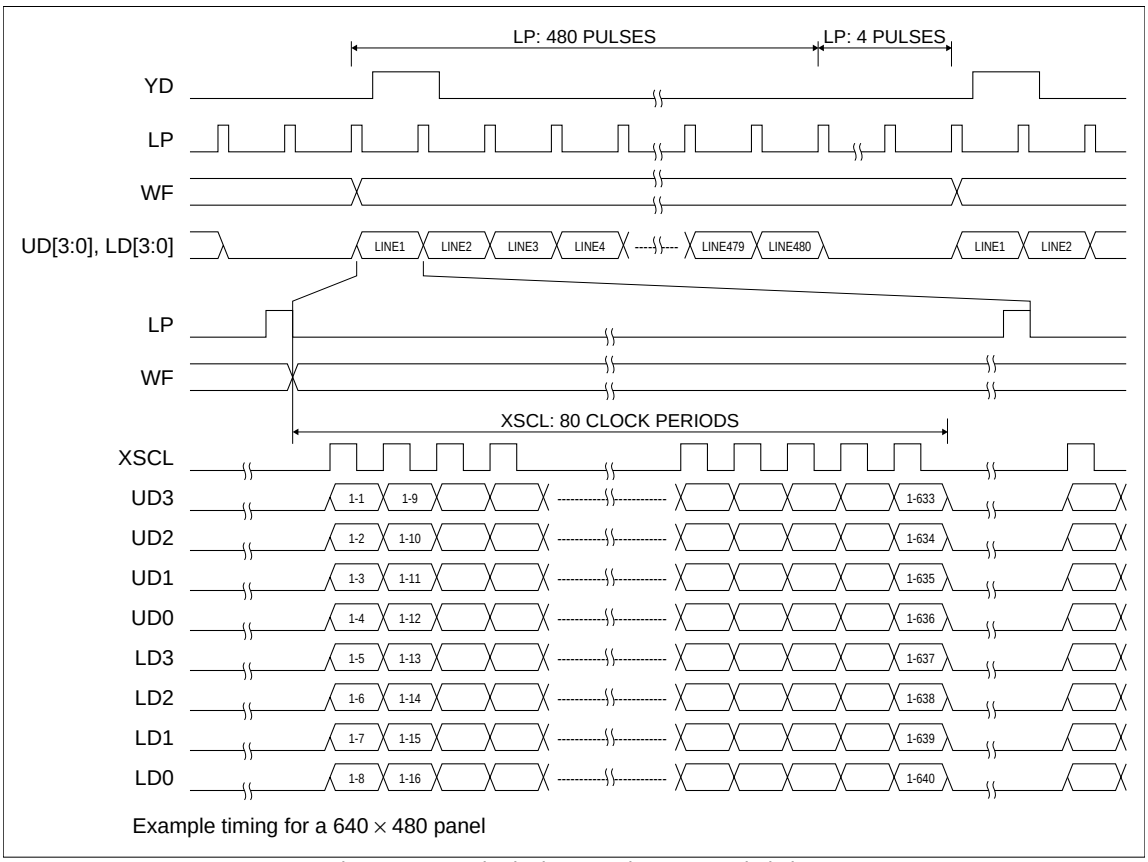



Figure 7-19 8-Bit Single Monochrome Panel Timing

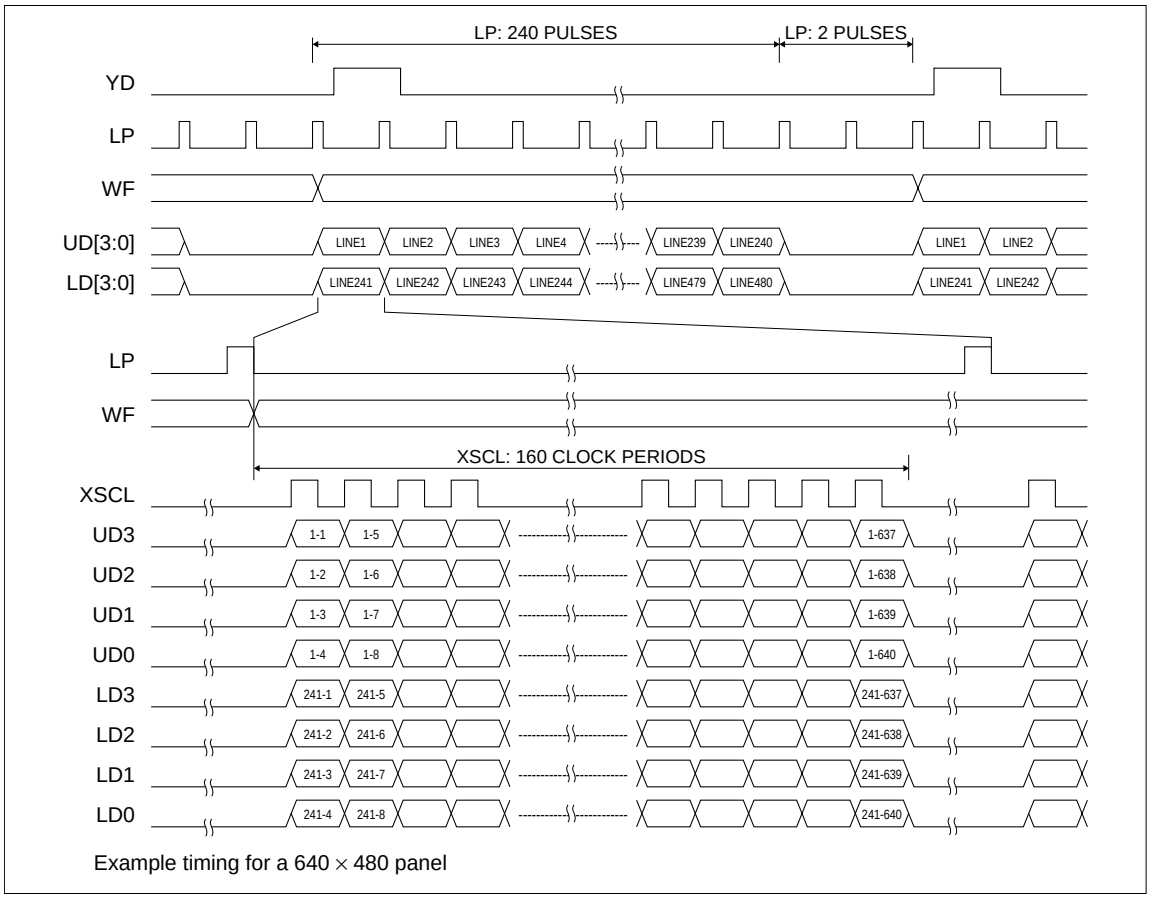



Figure 7-20 8-Bit Dual Monochrome Panel Timing

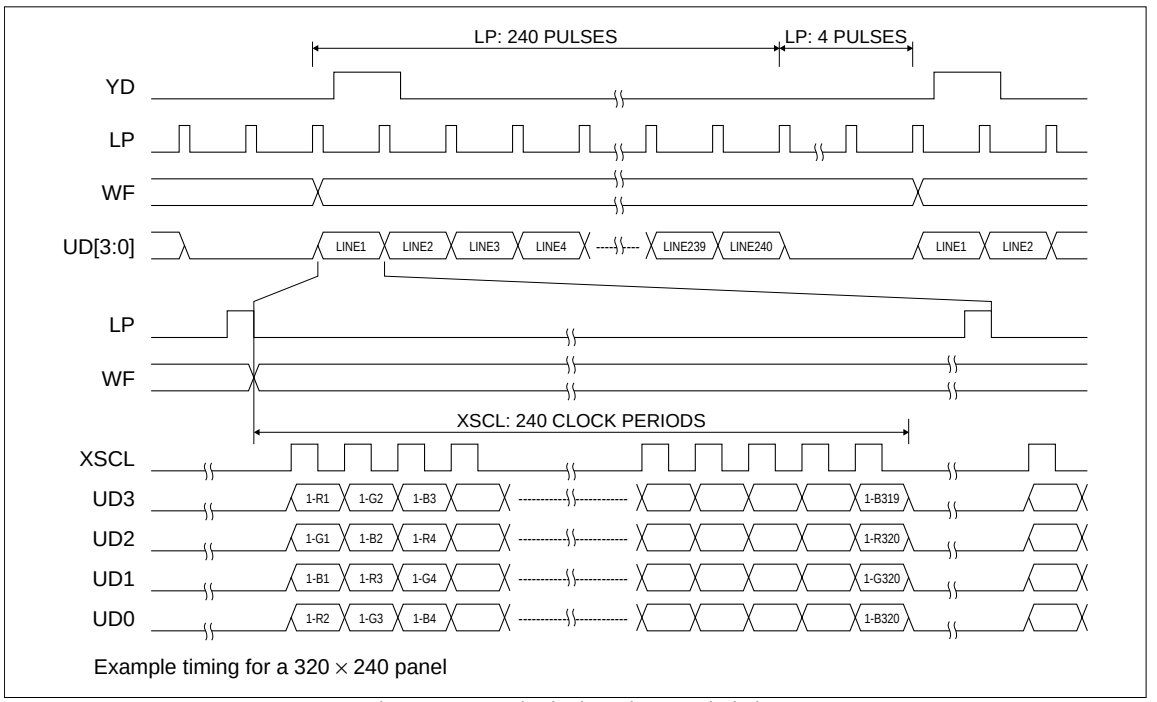



Figure 7-21 4-Bit Single Color Panel Timing

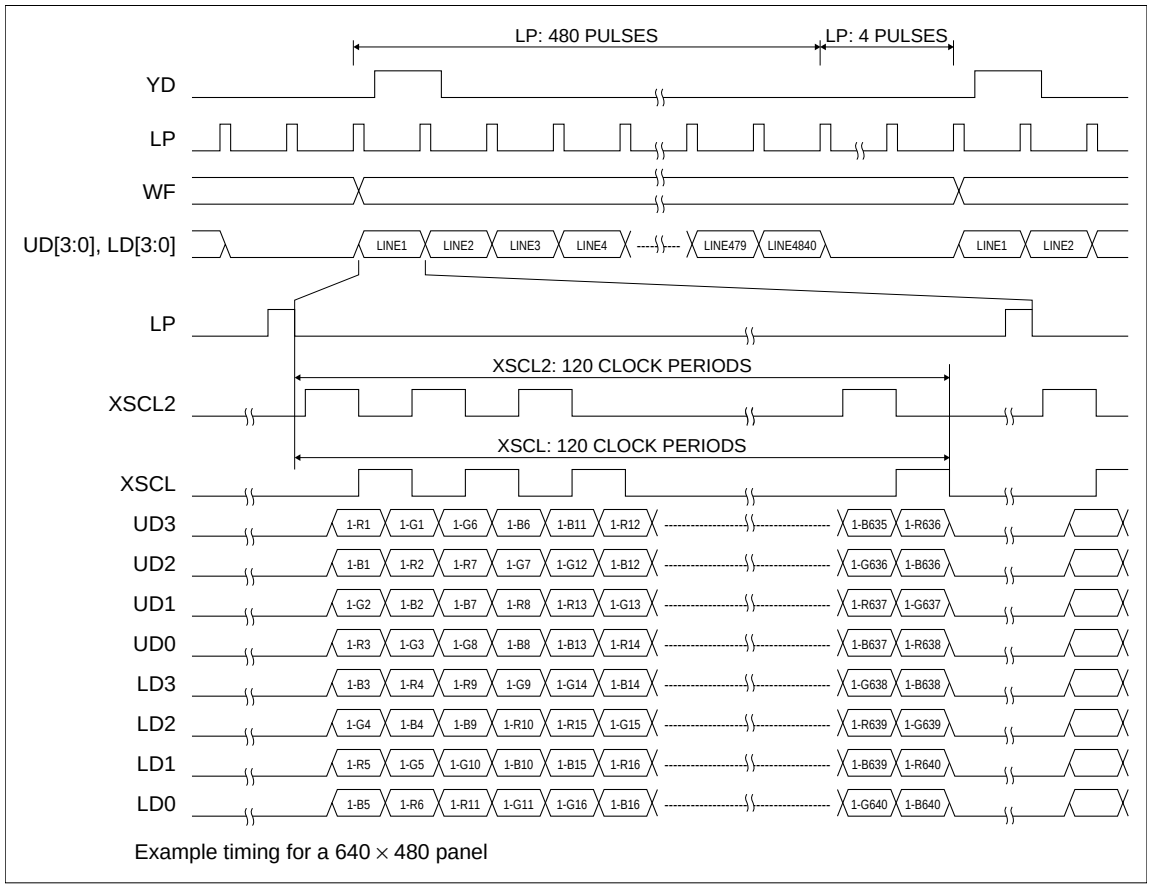



Figure 7-22 8-Bit Single Color Panel Timing - Format 1: AUX[03] Bit 3 = 0 and AUX[01] Bit 2 = 1

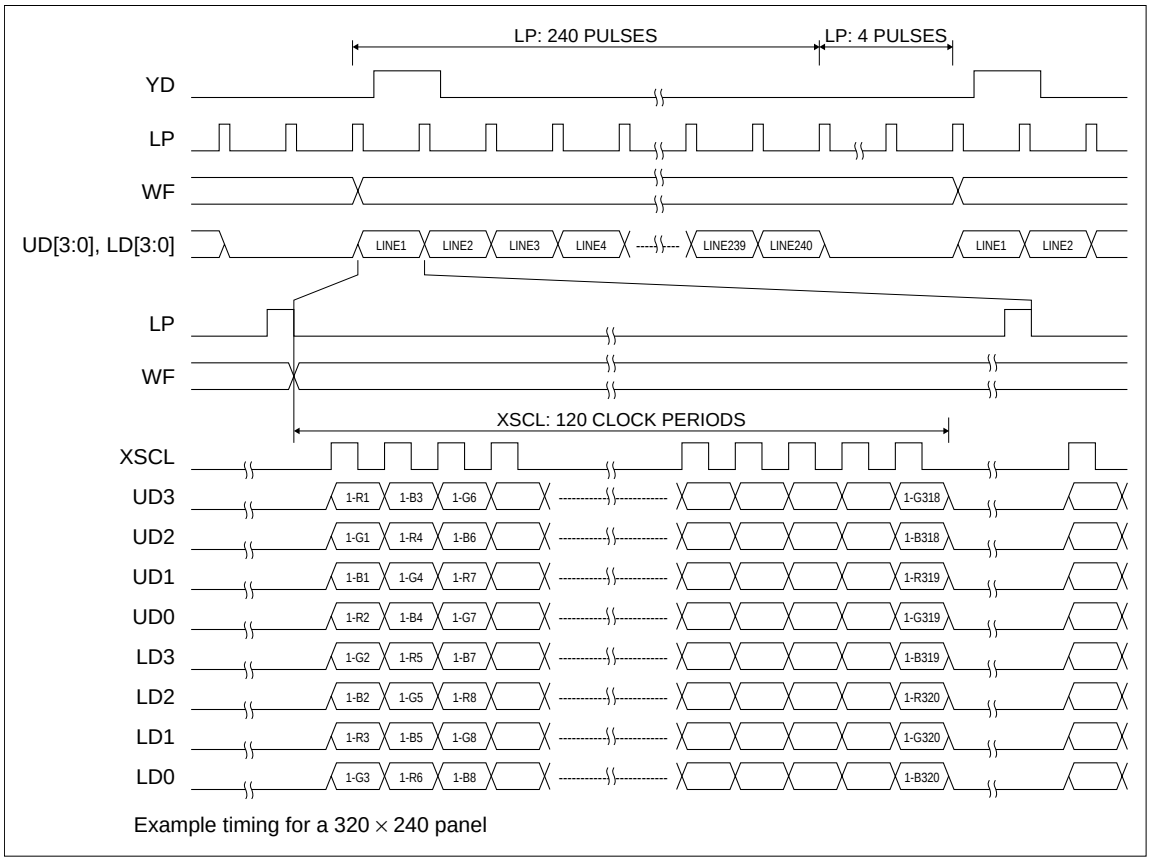



Figure 7-23 8-Bit Single Color Panel Timing - Format 2: AUX[03] Bit 3 = 1 and AUX[01] Bit 2 = 1

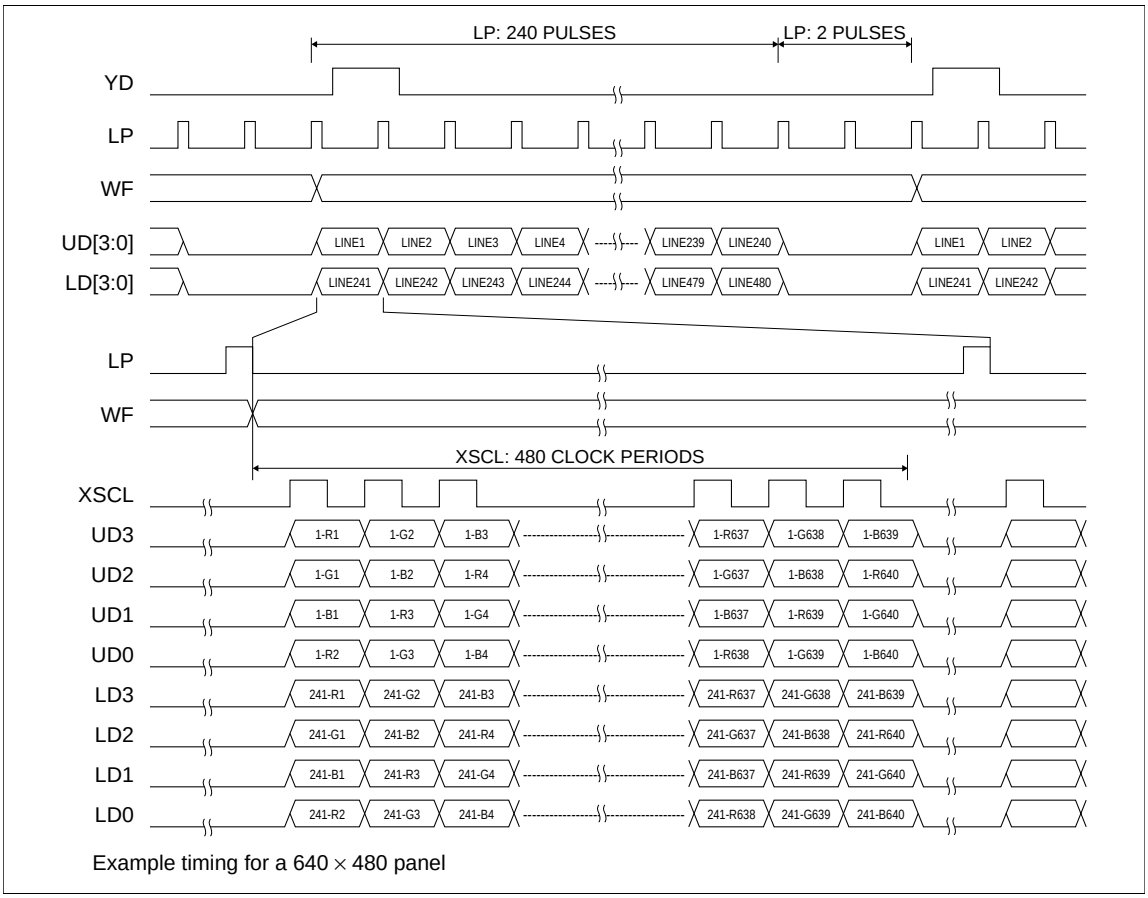



Figure 7-24 8-Bit Dual Color Panel Timing

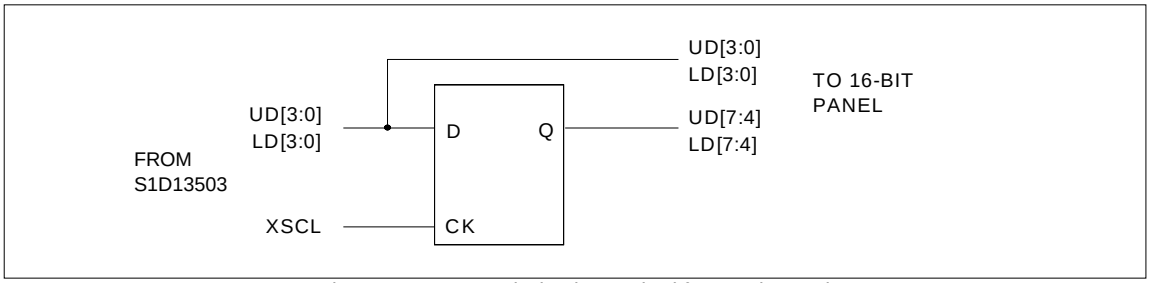
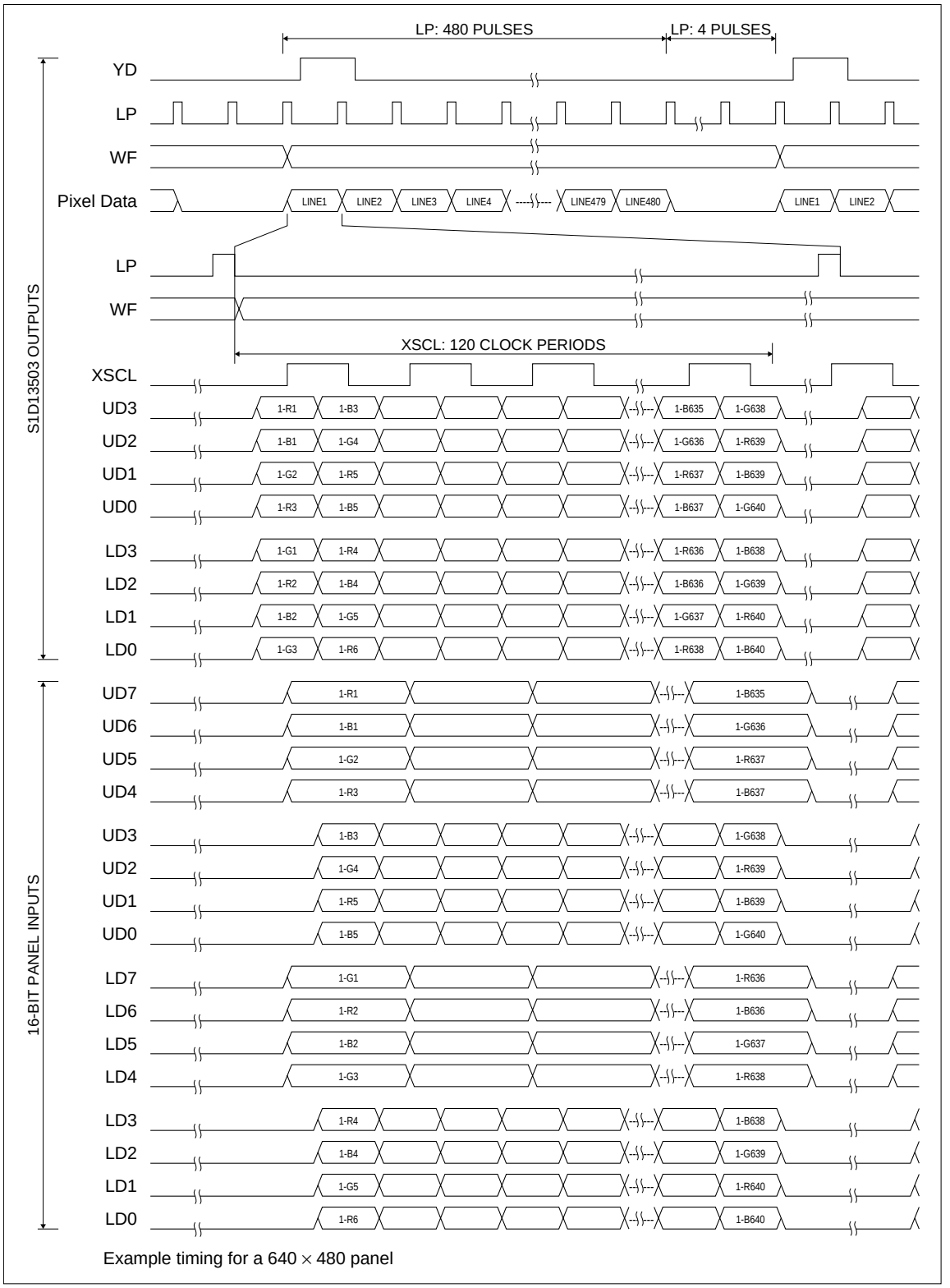



Figure 7-25 External Circuit Required for 16-Bit Panel




Example timing for a 640 × 480 panel

Figure 7-26 16-Bit Single Color Panel Timing with External Circuit

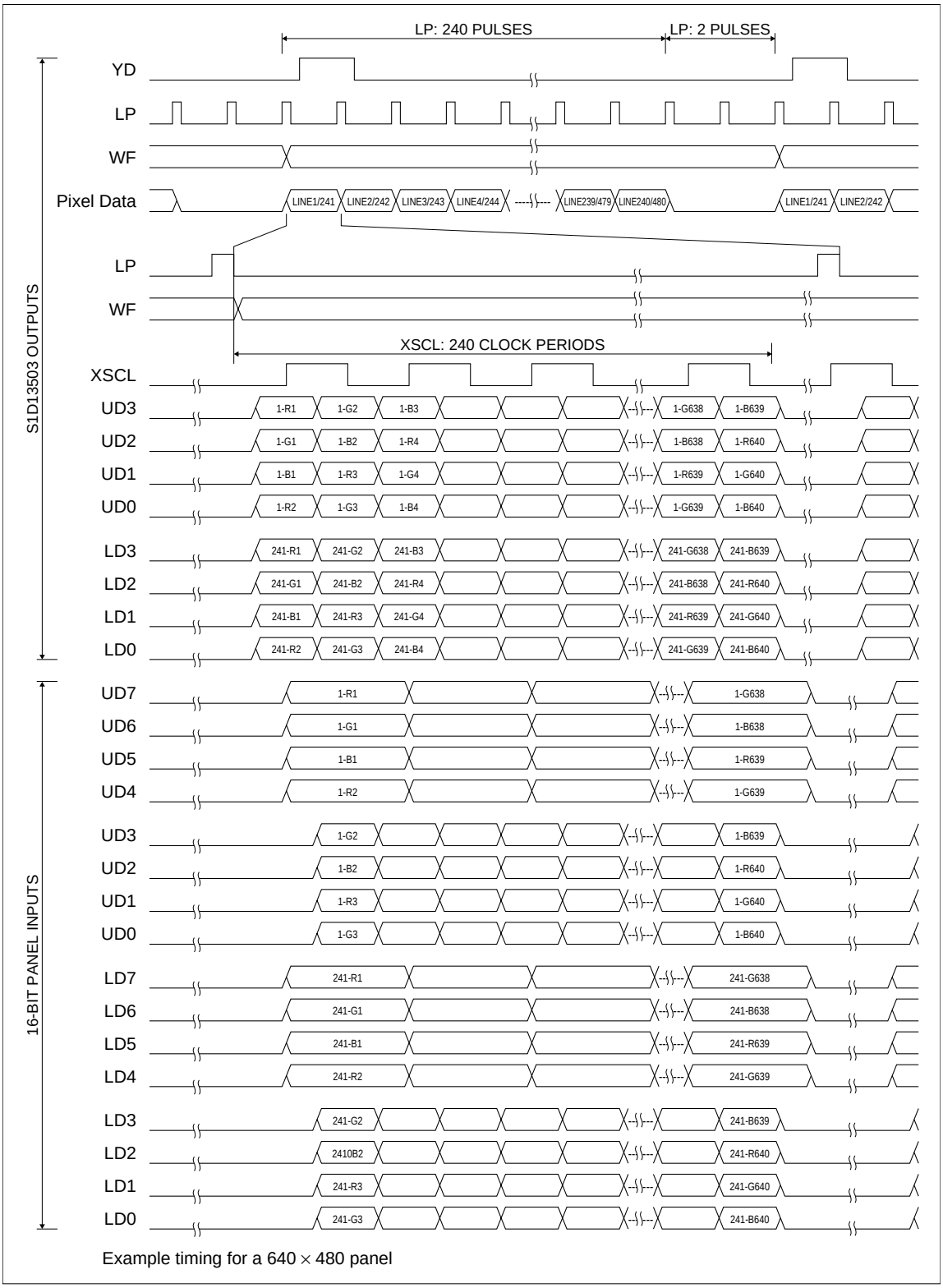



Figure 7-27 16-Bit Dual Color Panel Timing with External Circuit

# 8 HARDWARE REGISTER INTERFACE

The S1D13503 is configured and controlled via 16 internal 8-bit registers. There are two ways to map these registers into the system I/O space.

1. **Direct-mapping**: Absolute I/O address = system address lines AB[3:0] + base I/O mapped address (where base I/O address is selected by VD7–VD12, see Table 5-6.)

   This scheme requires 16 sequential I/O addresses starting from the I/O mapped base address selected by VD7–VD12 (see Table 5-6).

   To perform an I/O access;
   write data IOW {absolute I/O address}, {data}
   read data IOR {absolute I/O address}

2. **Indexing**: I/O address = internal index register bits [3:0]

   This scheme requires 2 sequential I/O addresses starting from the base address selected by VD4–VD12 (see Table 5-6).

   To perform an 8-bit I/O access;
   write index IOW {I/O mapped address}, {index} ; write the index of the register to be accessed
   then
   write data IOW {I/O mapped address +1}, {data} ; write data to the indexed register
   or
   read data IOR {I/O mapped address +1} ; read the indexed register

   To perform a 16-bit I/O access;
   write data IOW {I/O mapped address}, {index,data} ; write the index and data of the register to be accessed
   read data IOW {I/O mapped address}, {index} ; write to the indexed register
   IOR {I/O mapped address +1} ; read the indexed register

## 8.1 Register Descriptions

**AUX[00] Test Register**
I/O address = 0000b, Read/Write

| Test Mode Enable | Reserved | Test Input Select Bit 2 | Test Input Select Bit 1 | Test Input Select Bit 0 | Test Output Select Bit 2 | Test Output Select Bit 1 | Test Output Select Bit 0 |
|---|---|---|---|---|---|---|---|

**bit 7** **Test Mode Enable**
When this bit = 0 normal operation is enabled. When this bit = 1 the chip is placed in a special test mode. The test input bits and test output bits (bits 6:0) are used to select various internal test functions.

**bit 6** **Reserved**
During normal operation this bit must = 0.

**bits 5–0** **Test Mode Input and Output Bits [2:0]**
When bit 7 = 1 these are the Test Input Select Input and Output bits. When bits 6 and 7 = 0 (normal operation) these bits may be used as read/write scratch registers.

| AUX[01] Mode Register 0<br>I/O address = 0001b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| DISP | Panel | Mask XSCL | LCDE | Gray Shade / Color | LCD Data Width Bit 0 | Memory Interface | RAMS |

**bit 7 DISP**

This bit selects display on or off. When this bit = 0, Display OFF is selected (LD0–3 and UD0–3 are forced to 0). When this bit = 1, Display ON is selected. This bit goes low on RESET.

**bit 6 Panel**

This bit selects the LCD panel configuration (single or dual). When this bit = 0, Single LCD panel drive is selected. When this bit = 1 Dual LCD panel drive is selected. This bit goes low on RESET.

**bit 5 Mask XSCL**

XSCL is automatically masked during the horizontal non-display period if any of the following criteria is meet:

- AUX[0C] value is greater than 00h.
- Color panel is selected.
- This bit (AUX[01] bit 5) = 1.

XSCL will not be masked during the horizontal non-display period if color panel is not selected, AUX[0C] = 00h and this bit = 0.

**bit 4 LCDE**

This control bit, and its associated pin LCDENB, are intended for use in systems that implement LCD DC voltage protection. However, LCDE can be used for other purposes if required. When LCDE = 0, LCDENB is forced low. When LCDE = 1, LCDENB is forced high. LCDE goes low on RESET.

**bit 3 Gray Shade/Color**

In gray shade display modes, this bit selects between 16-level or 4-level gray shade display. When this bit = 1, 16 gray shades are displayed (4 bits/pixel). When this bit = 0, 4 gray shades of a possible 16 are displayed (2 bits/pixel).

In color display modes, this bit selects between 16 color or 4 color display. When this bit = 1, 16 colors are displayed out of a possible of 4096 colors (4 bits/pixel). When this bit = 0, 4 colors are displayed out of a possible of 4096 colors (2 bits/pixel).

This bit is ignored when either black-and-white (BW) or 256 color mode is selected (AUX[03] bit 2 = 1).
This bit goes low on RESET.

Table 8-1 Gray Shade/Color Mode Selection

| Display Modes | Gray Shade/ Color AUX[01] bit 3 | BW/ 256 Colors AUX[03] bit 2 | Color Mode AUX[03] bit 1 |
|---|---|---|---|
| 256 Colors | don't care | 1 | 1 |
| 16 Colors | 1 | 0 | 1 |
| 4 Colors | 0 | 0 | 1 |
| 16 Grays | 1 | 0 | 0 |
| 4 Grays | 0 | 0 | 0 |
| BW | don't care | 1 | 0 |

**bit 2** **LCD Data Width Bit 0**

Together with LCD Data Width bit 1 (AUX[03] bit 3) this bit selects different display data formats. The following table shows the function of these two bits:

Table 8-2 LCD Data Width

| Panel | LCD Data Width Bit 1 AUX[03] bit 3 | LCD Data Width Bit 0 AUX[01] bit 2 | Function |
|---|---|---|---|
| Monochrome | don't care | 0 | 4-bit LCD data width |
| Monochrome | don't care | 1 | 8-bit LCD data width |
| Color | 0 | 0 | 4-bit LCD data width |
| Color | 0 | 1 | 8-bit LCD data width - format 1 |
| Color | 1 | 0 | 16-bit LCD data width (with external circuit) |
| Color | 1 | 1 | 8-bit LCD data width - format 2 |

For 8-bit dual panels, the data transfer width is forced to 4 bits per panel. This bit goes low on RESET.

**bit 1** **Memory Interface**

This bit selects between the 8-bit or 16-bit memory interface. When this bit = 0, the 16-bit memory interface is selected. When this bit = 1, the 8-bit memory interface is selected. If 16-bit bus interface (VD0 = 1 on RESET) or 256 color mode (AUX[03] bits 2–1 = 11) is selected, the Memory Interface bit is forced to 0 internally (16-bit). This bit goes low on RESET.

**bit 0** **RAMS**

This bit configures the display memory address lines for an 8-bit memory interface system. When this bit = 0, addressing for 8K × 8 SRAM on an 8-bit display memory data bus interface is selected. When this bit = 1, addressing for 32K × 8 SRAM on an 8-bit display memory data bus interface is selected. This bit goes low on RESET. This bit is ignored for a 16-bit memory interface.

**AUX[02] Line Byte Count Register (LSB)**
I/O address = 0010b, Read/Write

| Line Byte Count Bit 7 | Line Byte Count Bit 6 | Line Byte Count Bit 5 | Line Byte Count Bit 4 | Line Byte Count Bit 3 | Line Byte Count Bit 2 | Line Byte Count Bit 1 | Line Byte Count Bit 0 |
|---|---|---|---|---|---|---|---|

**bits 7–0** **Line Byte Count Bits [7:0]**

Along with Line Byte Count Bit 8 (AUX[03] bit 0), this is the number of bytes to be fetched per display line minus 1. To calculate the Line Byte Count use the following formula:

$$\mathrm{LineByteCount(Decimal)} = \left(\frac{\mathrm{BitsPerPixel}}{\mathrm{MemoryInterfaceWidth}} \times \mathrm{HorizontalResolution}\right) - 1$$

**Example:**

To calculate the Line Byte Count for 640 horizontal pixels with 16 gray shades (4 bits-per-pixel) and 16-bit memory interface:

$$\mathrm{LineByteCount(Decimal)} = \frac{4\mathrm{BitsPerPixel}}{16\mathrm{Bits}} \times 640 - 1 = 159$$

The following two tables summarize the maximum value of the Line Byte Count Register for different display modes and display memory interface.

Table 8-3 Maximum Value of Line Byte Count Register - 8-Bit Display Memory Interface

| Display Modes | Maximum Value of Line Byte Count Register | Corresponding Maximum Number of Pixels in One Display Line |
|---|---|---|
| black-and-white (BW) | 0FFh | 256 × 8 = 2048 |
| 4-level gray shade / 4 colors | 0FFh | 256 × 4 = 1024 |
| 16-level gray shade / 16 colors | 1FFh | 512 × 2 = 1024 |

Table 8-4 Maximum Value of Line Byte Count Register - 16-Bit Display Memory Interface

| Display Modes | Maximum Value of Line Byte Count Register | Corresponding Maximum Number of Pixels in One Display Line |
|---|---|---|
| black-and-white (BW) | 0FFh | 256 × 16 = 4096 |
| 4-level gray shade / 4 colors | 0FFh | 256 × 8 = 2048 |
| 16-level gray shade / 16 colors | 0FFh | 256 × 4 = 1024 |
| 256 colors | 1FFh | 512 × 2 = 1024 |

**AUX[03] Mode Register 1**
I/O address = 0011b, Read/Write

| PS Bit 1 | PS Bit 0 | LCD Signal State | LUT Bypass | LCD Data Width Bit 1 | BW / 256 Colors | Color Mode | Line Byte Count Bit 8 |
|---|---|---|---|---|---|---|---|

**bits 7–6 PS Bits [1:0]**

Selects the Power Save Modes as shown in the following table. The PS bits [1:0] go low on RESET.

Table 8-5 Power Save Mode Selection

| PS1 | PS0 | Mode Activated |
|---|---|---|
| 0 | 0 | Normal Operation |
| 0 | 1 | Power Save Mode 1 |
| 1 | 0 | Power Save Mode 2 |
| 1 | 1 | Reserved |

Refer to *"Power Save Modes"* on page 52 for a complete Power Save Mode description.

**bit 5 LCD Signal State**

When this bit = 0, all LCD interface signals are forced low during Power Save modes. When this bit = 1, all LCD interface signals are forced to a high impedance (Hi-Z) state during Power Save modes. This bit goes low on RESET.

**bit 4 LUT Bypass**

When the LUT Bypass bit = 0, the Look-Up Table is used for display data output in gray shade modes. When this bit = 1, the Look-Up Table is bypassed for display data output in gray shade modes (for power save purposes). There is no effect on changing this bit in BW and color modes. In BW display mode, the Look-Up Table is always bypassed and in color display mode the Look-Up Table cannot be bypassed. The LUT Bypass bit goes low on RESET.

**bit 3 LCD Data Width Bit 1**

Together with LCD Data Width bit 0 (AUX[01] bit 2), this bit selects different display data formats. See Table 8-2, "LCD Data Width," on page 40 for details. This bit goes low on RESET.

**bit 2 BW / 256 Colors**

In BW/gray shade display modes, when this bit = 1, black-and-white (BW) mode is selected. When this bit = 0, either 4-level gray shade mode or 16-level gray shade mode is selected (depending on AUX[01] bit 3).

In color display modes, when this bit = 1, 256 color mode is selected. When this bit = 0, either 4 color mode or 16 color mode is selected. See Table 8-1, "Gray Shade/Color Mode Selection," on page 39 for details. This bit goes low on RESET.

**bit 1** **Color Mode**

When this bit = 1, color display modes are selected. When bit = 0, BW/gray shade display modes are selected. This bit goes low on RESET. See Table 8-1, “Gray Shade/Color Mode Selection,” on page 39 for details.

**bit 0** **Line Byte Count Bit 8**

This is the MSB of the number of bytes to be fetched per display line minus 1 (see AUX[02]). This bit only has effect when in either 16 colors/gray shades with 8-bit memory interface or 256 colors with 16-bit memory interface.

**AUX[04] Total Display Line Count Register (LSB) (Vertical Total)**
I/O address = 0100b, Read/Write

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Total Disp. Line Count Bit 7 | Total Disp. Line Count Bit 6 | Total Disp. Line Count Bit 5 | Total Disp. Line Count Bit 4 | Total Disp. Line Count Bit 3 | Total Disp. Line Count Bit 2 | Total Disp. Line Count Bit 1 | Total Disp. Line Count Bit 0 |

**bits 7–0** **Total Display Line Count Bits [7:0]**

These are the 8 LSB of the 10-bit Total Display Line Count and represent the number of scan lines -1, to a maximum value of 3FFh or 1024 scan lines. In dual panel mode, the actual number of display lines is twice the programmed value, i.e. to a maximum of 2048 scan lines. Note that the value programmed partially determines the frame period, and hence affects display duty cycle. Bits 8 and 9 are located in the following register (AUX[05]).

**AUX[05] Total Display Line Count (MSB) and WF Count Register**
I/O address = 0101b, Read/Write

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| WF Count Bit 5 | WF Count Bit 4 | WF Count Bit 3 | WF Count Bit 2 | WF Count Bit 1 | WF Count Bit 0 | Total Disp. Line Count Bit 9 | Total Disp. Line Count Bit 8 |

**bits 7–2** **WF Count Bits [5:0]**

These bits are used to adjust the WF output signal period. The binary value stored in these bits represents the number of LP pulses -1 between toggles of the WF output. The power up reset value of these bits is 0, which causes the WF output to toggle every frame. When values of 01h to 3Fh are programmed into these bits, the results are WF toggling every 1+n LP pulses, where n is the value programmed. These bits have no effect when 8-bit single color panel format 1 is selected.

**bits 1–0** **Total Display Line Count Bits [9:8]**

These bits are the two MSB of the Total Display Line Count Register (AUX[04]).

| AUX[06] Screen 1 Display Start Address Register (LSB)<br>I/O address = 0110b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| Screen 1 Display Start Addr Bit 7 | Screen 1 Display Start Addr Bit 6 | Screen 1 Display Start Addr Bit 5 | Screen 1 Display Start Addr Bit 4 | Screen 1 Display Start Addr Bit 3 | Screen 1 Display Start Addr Bit 2 | Screen 1 Display Start Addr Bit 1 | Screen 1 Display Start Addr Bit 0 |

| AUX[07] Screen 1 Display Start Address Register (MSB)<br>I/O address = 0111b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| Screen 1 Display Start Addr Bit 15 | Screen 1 Display Start Addr Bit 14 | Screen 1 Display Start Addr Bit 13 | Screen 1 Display Start Addr Bit 12 | Screen 1 Display Start Addr Bit 11 | Screen 1 Display Start Addr Bit 10 | Screen 1 Display Start Addr Bit 9 | Screen 1 Display Start Addr Bit 8 |

**AUX[06] bits 7–0, AUX[07] bits 7–0**

**Screen 1 Display Start Address Bits [15:0]**

These 16 bits determine the Screen 1 Display Start Address. In an 8-bit memory configuration these bits set the 16-bit start address (i.e., byte access). In a 16-bit memory configuration these are the 16 most significant bits of a 17-bit start address (i.e., word access).

**Note:** The absolute address into display memory is determined by the Memory Mapping Address which is set by VD13–VD15 (see Table 5-6, “Summary of Power On / Reset Options,” on page 15).

The Screen 1 Display Start Address is the memory address corresponding to the first displayed pixel (top left corner). In a dual panel configuration, screen 1 refers to the upper half of the display. While in a single panel configuration, screen 1 refers to the first screen of the Split Screen Display feature where two different images (screen 1 and screen 2) can be displayed at the same time on one display.

| AUX[08] Screen 2 Display Start Address Register (LSB)<br>I/O address = 1000b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| Screen 2 Display Start Addr Bit 7 | Screen 2 Display Start Addr Bit 6 | Screen 2 Display Start Addr Bit 5 | Screen 2 Display Start Addr Bit 4 | Screen 2 Display Start Addr Bit 3 | Screen 2 Display Start Addr Bit 2 | Screen 2 Display Start Addr Bit 1 | Screen 2 Display Start Addr Bit 0 |

| AUX[09] Screen 2 Display Start Address Register (MSB)<br>I/O address = 1001b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| Screen 2 Display Start Addr Bit 15 | Screen 2 Display Start Addr Bit 14 | Screen 2 Display Start Addr Bit 13 | Screen 2 Display Start Addr Bit 12 | Screen 2 Display Start Addr Bit 11 | Screen 2 Display Start Addr Bit 10 | Screen 2 Display Start Addr Bit 9 | Screen 2 Display Start Addr Bit 8 |

**AUX[08] bits 7–0, AUX[09] bits 7–0**

**Screen 2 Display Start Address Bits [15:0]**

These 16 bits determine the Screen 2 Display Start Address. In an 8-bit memory configuration these bits set the 16-bit start address (i.e., byte access). In a 16-bit memory configuration these are the 16 most significant bits of a 17-bit start address (i.e., word access).

In a dual panel configuration, screen 2 refers to the lower half of the display. The Screen 2 Display Start Address is the memory address corresponding to first displayed pixel in the first line of the lower half of the display and is calculated with the following formula.

$$\text{Screen2DisplayStartAddress(hex)} = \frac{(\text{ImageHorizontalResolution}) \times (\text{ImageVerticalResolution}) \times (\text{BytesPerPixel})}{2 \times \left(\frac{\text{MemoryInterfaceWidth}}{8}\right)} + \text{Screen1DisplayStartAddress}$$

In a single panel configuration, screen 2 refers to the second screen of the Split Screen Display Feature where two different images (screen 1 and screen 2) can be displayed at the same time on one display. The Screen 2 Display Start Address is the memory address corresponding to the first pixel of the second image stored in display memory. To display screen 2 refer to *“AUX[0A] Screen 1 Display Line Count Register (LSB)”* on page 44 and *“AUX[0C] Horizontal Non-Display Period”* on page 44.

| AUX[0A] Screen 1 Display Line Count Register (LSB)<br>I/O address = 1010b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| Screen 1 Display Line Count Bit 7 | Screen 1 Display Line Count Bit 6 | Screen 1 Display Line Count Bit 5 | Screen 1 Display Line Count Bit 4 | Screen 1 Display Line Count Bit 3 | Screen 1 Display Line Count Bit 2 | Screen 1 Display Line Count Bit 1 | Screen 1 Display Line Count Bit 0 |

| AUX[0B] Screen 1 Display Line Count Register (MSB)<br>I/O address = 1011b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| n/a | n/a | n/a | n/a | n/a | n/a | Screen 1Display Line Count Bit 9 | Screen 1Display Line Count Bit 8 |

**AUX[0A] bits 7–0, AUX[0B] bits 1–0**

**Screen 1 Display Line Count Bits [9:0]**

These bits are used to determine the number of lines displayed for screen 1. The remaining lines will automatically display from the screen 2 display start address. The 10-bit value programmed is the number of display lines -1.

This register is used to enable the split screen display feature (single panel only) where two different images can be displayed at the same time on one display.

For example; AUX[0A] = 20h for a 320 × 240 display system. The display will display 20h + 1 = 33 lines on the upper part of the screen as dictated by the screen 1 display start address registers (AUX[06] and AUX[07]), and 240 - 33 = 207 lines will be displayed on the lower part of the screen as dictated by the screen 2 display start address registers (AUX[08] and AUX[09]).

Two different images can be displayed when using a dual panel configuration by changing the screen 2 display start address. However, by using this method screen 2 is limited to the lower half of the display.

This register is ignored in dual panel mode.

| AUX[0C] Horizontal Non-Display Period<br>I/O address = 1100b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| Horizontal Non-Display Period Bit 7 | Horizontal Non-Display Period Bit 6 | Horizontal Non-Display Period Bit 5 | Horizontal Non-Display Period Bit 4 | Horizontal Non-Display Period Bit 3 | Horizontal Non-Display Period Bit 2 | Horizontal Non-Display Period Bit 1 | Horizontal Non-Display Period Bit 0 |

**bits 7–0 Horizontal Non-Display Period Bits [7:0]**

These bits are used to adjust the horizontal non-display period (see *"Frame Rate Calculation"* on page 57 for details). When these bits = 0, the fixed default non-display period (DHNDP) is used. Otherwise, a non-display period of DHNDP & AUX[0C] +1 is used. The unit of AUX[0C] is the same as the unit of Line Byte Count Register, i.e. number of bytes to be fetched. See description of AUX[02] and Section 9.3 on page 57 for details.

For example, if an additional 32 pixels wide of horizontal non-display period is desired in a 4 grays (2 bits-per-pixel) and 16-bit display memory interface system: AUX[0C] = [32 / (16 / 2)] - 1 = 3.

Note that the value programmed determines the period of one line, and hence affects the frame period.

| AUX[0D] Address Pitch Adjustment Register<br>I/O address = 1101b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| Addr Pitch Adjustment Bit 7 | Addr Pitch Adjustment Bit 6 | Addr Pitch Adjustment Bit 5 | Addr Pitch Adjustment Bit 4 | Addr Pitch Adjustment Bit 3 | Addr Pitch Adjustment Bit 2 | Addr Pitch Adjustment Bit 1 | Addr Pitch Adjustment Bit 0 |

**bits 7–0 Address Pitch Adjustment Bits [7:0]**

This register controls the virtual display by setting the numerical difference between the last address of a display line, and the first address in the following line.

If the Address Pitch Adjustment is not equal to zero, then a virtual screen is formed. The size of the virtual screen is only limited by the available display memory. The actual display output is a window that is part of the whole image stored in the display memory. For example, with 128K of display memory, a 640 × 400 16-gray image can be stored. If the output display size is 320 × 240, then the whole image can be seen by changing display starting addresses through AUX[06] and [07], and AUX[08] and [09]. Note that a virtual screen can be produced on either a single or dual panel.

In 8-bit memory interface, if the Address Pitch Adjustment is not equal to zero, a virtual screen with a line length of (Line Byte Count + AUX[0D]) bytes is created, with the display reflecting the contents of a window (Line Byte Count + 1) bytes wide. The position of the window on the virtual screen is determined by AUX[06] and [07], and AUX[08] and [09].

In 16-bit memory interface, if the Address Pitch Adjustment is not equal to zero, then a virtual screen with a line length of 2 × (Line Byte Count + AUX[0D]) bytes is created, with the display reflecting the contents of a window 2 × (Line Byte Count + 1) bytes wide. The position of the window on the virtual screen is determined by AUX[06] and [07], and AUX[08] and [09].

| AUX[0E] Look-Up Table Address Register<br>I/O address = 1110b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| Green Bank Bit 1 | Green Bank Bit 0 | ID Bit / RGB Index Bit 1 | ID Bit / RGB Index Bit 0 | Palette Address Bit 3 | Palette Address Bit 2 | Palette Address Bit 1 | Palette Address Bit 0 |

The S1D13503 has three internal 16 position, 4-bit wide Look-Up Tables (palettes). The 4-bit value programmed into each table position determines the output gray shade / color weighting of display data. These tables are bypassed in black-and-white (BW) display mode.

These three 16 position Look-Up Tables can be arranged in many different configurations to accommodate all the gray shade / color display modes.

Refer to *"Look-Up Table Architecture"* on page 48 for formats.

**bits 7–6 Green Bank Bits [1:0]**

In 4-level gray / color display modes (2 bits/pixel), the 16 position Green palette is arranged into four, 4 position "banks". These two bits control which bank is currently selected. These bits have no effect in 16-level gray / color display modes (4 bits/pixel).

In 256 color display modes (8 bits/pixel), the 16 position Green palette is arranged into two, 8 position "banks" for the display of "green" colors. Only bit 0 of these two bits controls which bank is currently selected.

**bits 5–4 ID Bits / RGB Index Bits [1:0]**

These bits have dual purpose;

***ID Bits:*** After power on or hardware reset, these bits can be read to identify the current revision of the S1D13503. These same bits are used to identify the pin compatible SED1352 and would only be used in system implementations where common software is being used. As these bits are R/W they must be read before being written in order to be used as ID bits.

Table 8-6 ID Bit Usage

| | Chip | AUX[0E] | |
|---|---|---|---|
| | | Bit 5 | Bit 4 |
| Power On or RESET | S1D13503 | 0 | 0 |
| | reserved | 0 | 1 |
| | SED1352F0B/F1B/D0B | 1 | 0 |
| | SED1352F0A/F1A/D0A | 1 | 1 |

***RGB Index Bits [1:0]:*** These bits are also used to provide access to the three internal Look-Up Tables (RGB).

Table 8-7 Look-Up Table Access

| AUX[0E] | | Look-Up Table Access |
|---|---|---|
| Bit 5 | Bit 4 | |
| 0 | 0 | Auto-increment (see Note) |
| 0 | 1 | Red palette R/W access |
| 1 | 0 | Green palette R/W access |
| 1 | 1 | Blue palette R/W access |

**Note:** When auto-increment is selected, an internal pointer will default to the Red palette on power on reset. Each read/write access to AUX[0F] will increment the counter to point to the next palette in order (RGB). Whenever the Look-Up Table Address register AUX[0E] is written, the RGB Index will reset the pointer to the Red palette. This provides a efficient method for sequential writing of RGB data.

**bits 3–0 Palette Address Bits [3:0]**

These 4 bits provide a pointer into the 16 position Look-Up Table currently selected for CPU R/W access.

**Note:** The Look-Up Table configuration (e.g. 1/2/4 banks) does not affect the R/W access from the CPU as all 16 positions can be accessed sequentially.

| **AUX[0F] Look-Up Table Data Register**<br>I/O address = 1111b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| Red Bank Bit 1 | Red Bank Bit 0 | Blue Bank Bit 1 | Blue Bank Bit 0 | Palette Data Bit 3 | Palette Data Bit 2 | Palette Data Bit 1 | Palette Data Bit 0 |

**bits 7–6 Red Bank Bits [1:0]**

In 4-level color display modes, the 16 position Red palette is arranged into four, 4 position “banks”. These two bits control which bank is currently selected.
In 256 color display modes, the 16 position, Red palette is arranged into two, 8 position “banks” for the display of “red” colors. Only bit 0 of these two bits controls which bank is currently selected.
These bits have no effect in all gray shade or 16-color display modes.

**bits 5–4 Blue Bank Bits [1:0]**

In both the 4 and 256 color display modes, the 16 position Blue palette is arranged into four 4 position “banks” for the display of “blue” colors. These two bits control which bank is currently selected.
These bits have no effect in all gray shade display modes or 16 color display modes.

**bits 3–0 Palette Data Bits [3:0]**

These 4 bits are the gray shade / color values used for display data output. They are programmed into the 4-bit Look-Up Table (palettes) positions pointed to by Palette Address bits [3:0] and RGB Index bits [1:0] (if in color display modes).

For example; in a 16-level gray shade display mode, a data value of 0001b (4 bits/pixel) will point to Look-Up Table position one and display the 4-bit gray shade corresponding to the value programmed into that location.

# 8.2 Look-Up Table Architecture

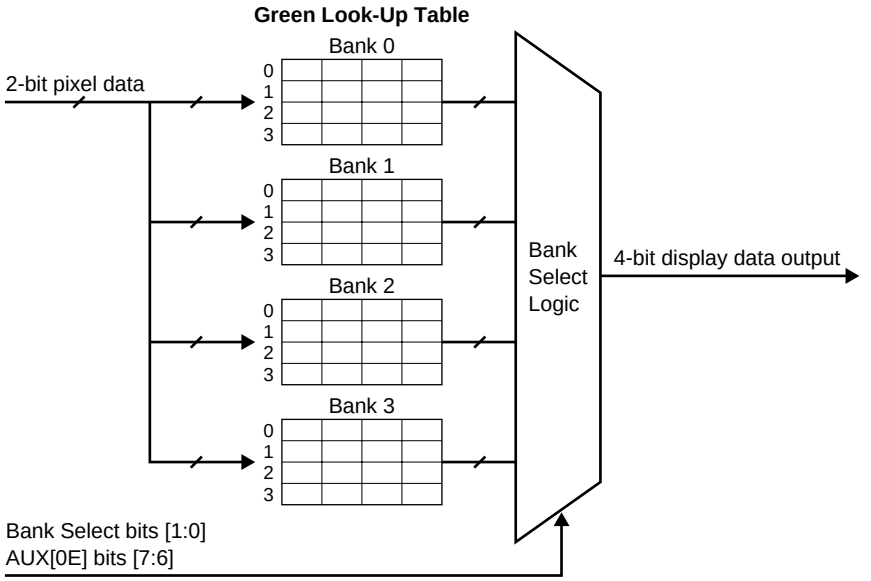
Table 8-8 Look-Up Table Configurations

| Display Mode | 4-Bit Wide Palette | | |
|---|---|---|---|
| | RED | GREEN | BLUE |
| Black & White | | | |
| 4-level gray | | 4 banks of 4 | |
| 16-level gray | | 1 bank of 16 | |
| 4 color | 4 banks of 4 | 4 banks of 4 | 4 banks of 4 |
| 16 color | 1 bank of 16 | 1 bank of 16 | 1 bank of 16 |
| 256 color | 2 banks of 8 | 2 banks of 8 | 4 banks of 4 |

Indicates the palette is not used for that display mode.

## *Gray Shade Display Modes*

### 4-Level Gray Shade Mode




**Note:** The above depiction is intended to show the display data output path only. The CPU R/W access to the individual Look-Up Tables is not affected by the various 'banking' configurations.

Figure 8-1 4-Level Gray-Shade Mode Look-Up Table Architecture

### 16-Level Gray Shade Mode

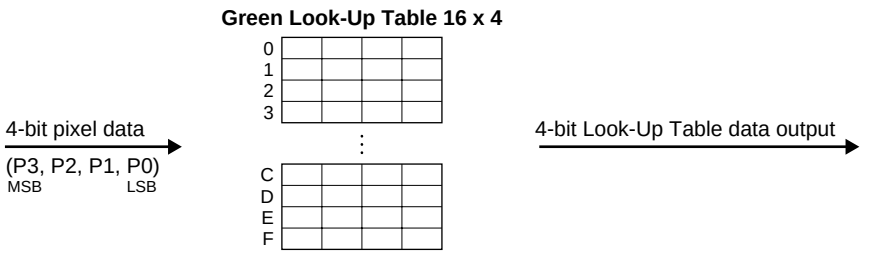



Figure 8-2 16-Level Gray-Shade Mode Look-Up Table Architecture

**Note:** The Look-Up Table is bypassed in black-and-white display mode.

# *Color Display Modes*

## 4-Level Color Mode

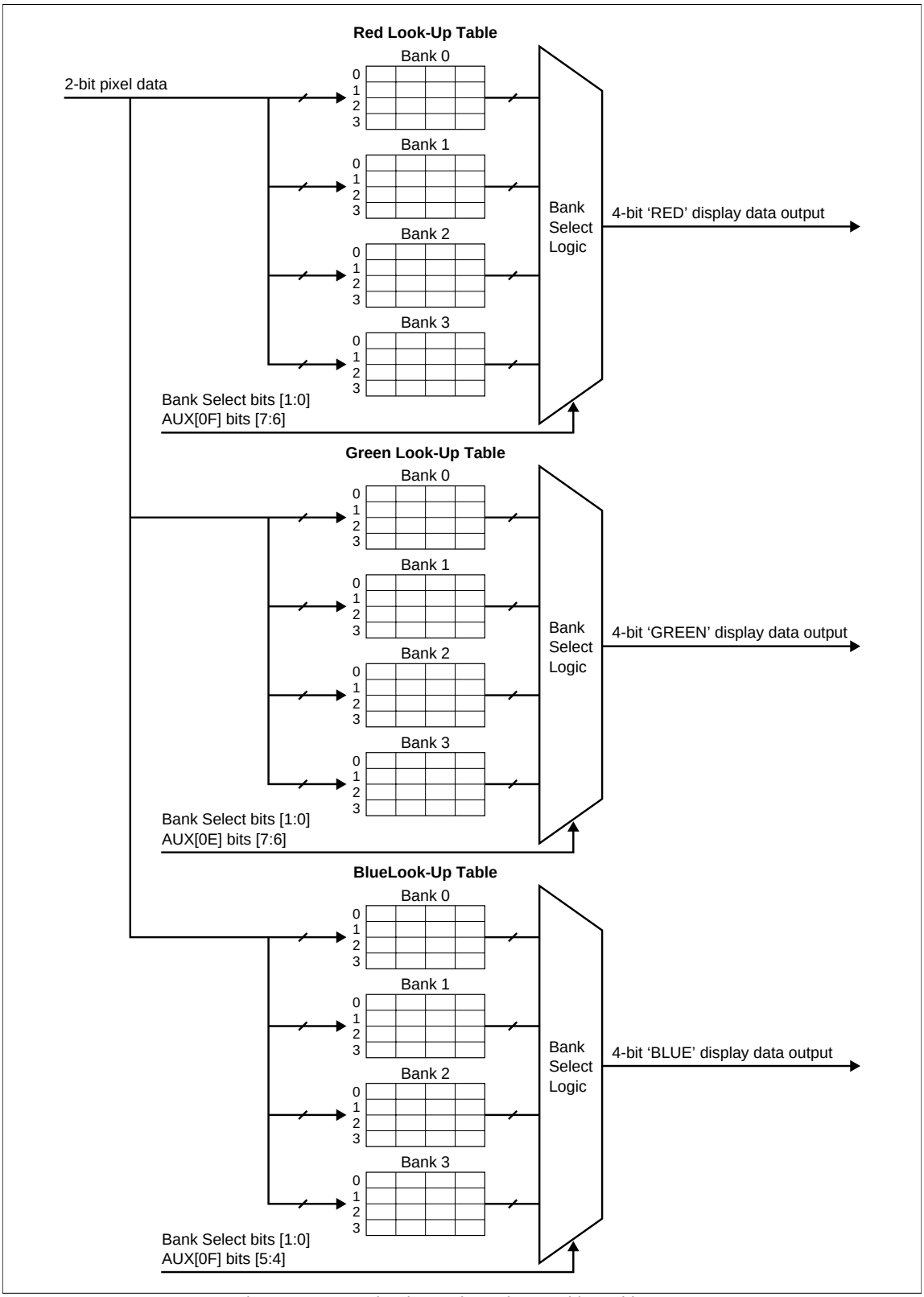



Figure 8-3 4-Level Color Mode Look-Up Table Architecture

## 16-Level Color Mode

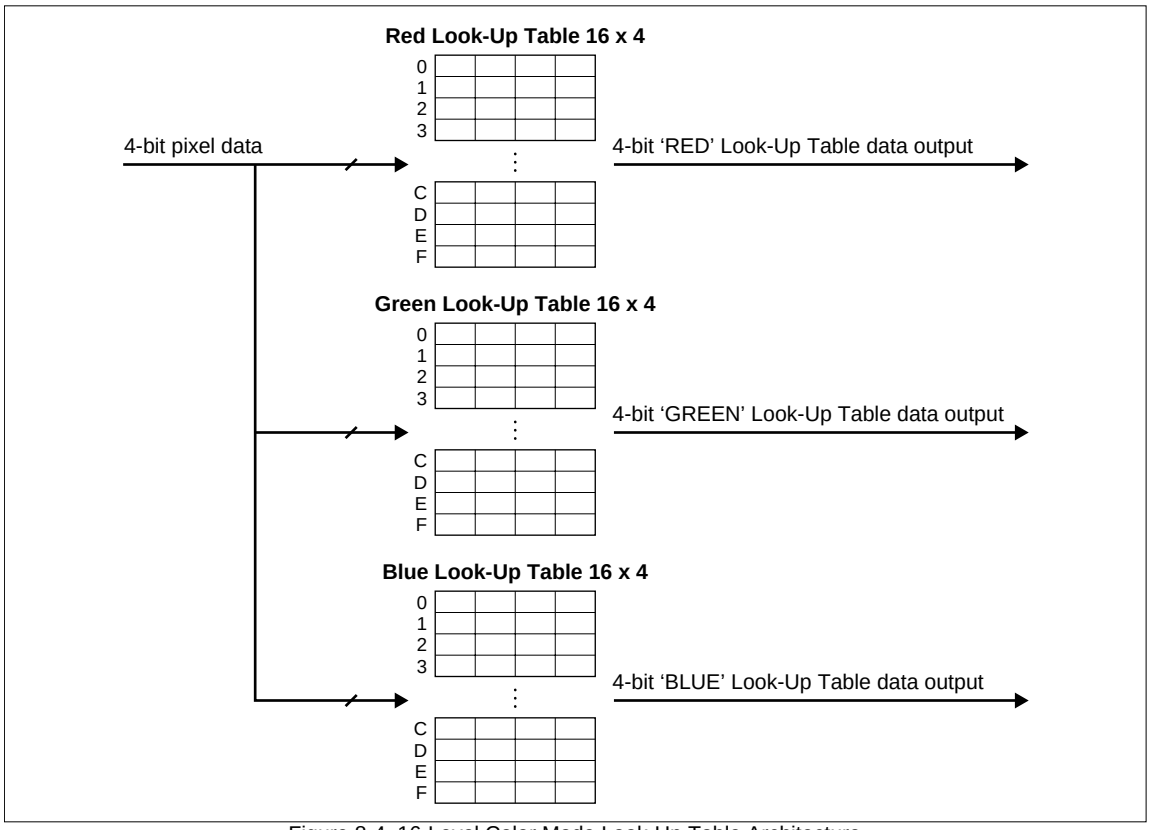



Figure 8-4 16-Level Color Mode Look-Up Table Architecture

## 256-Level Color Mode

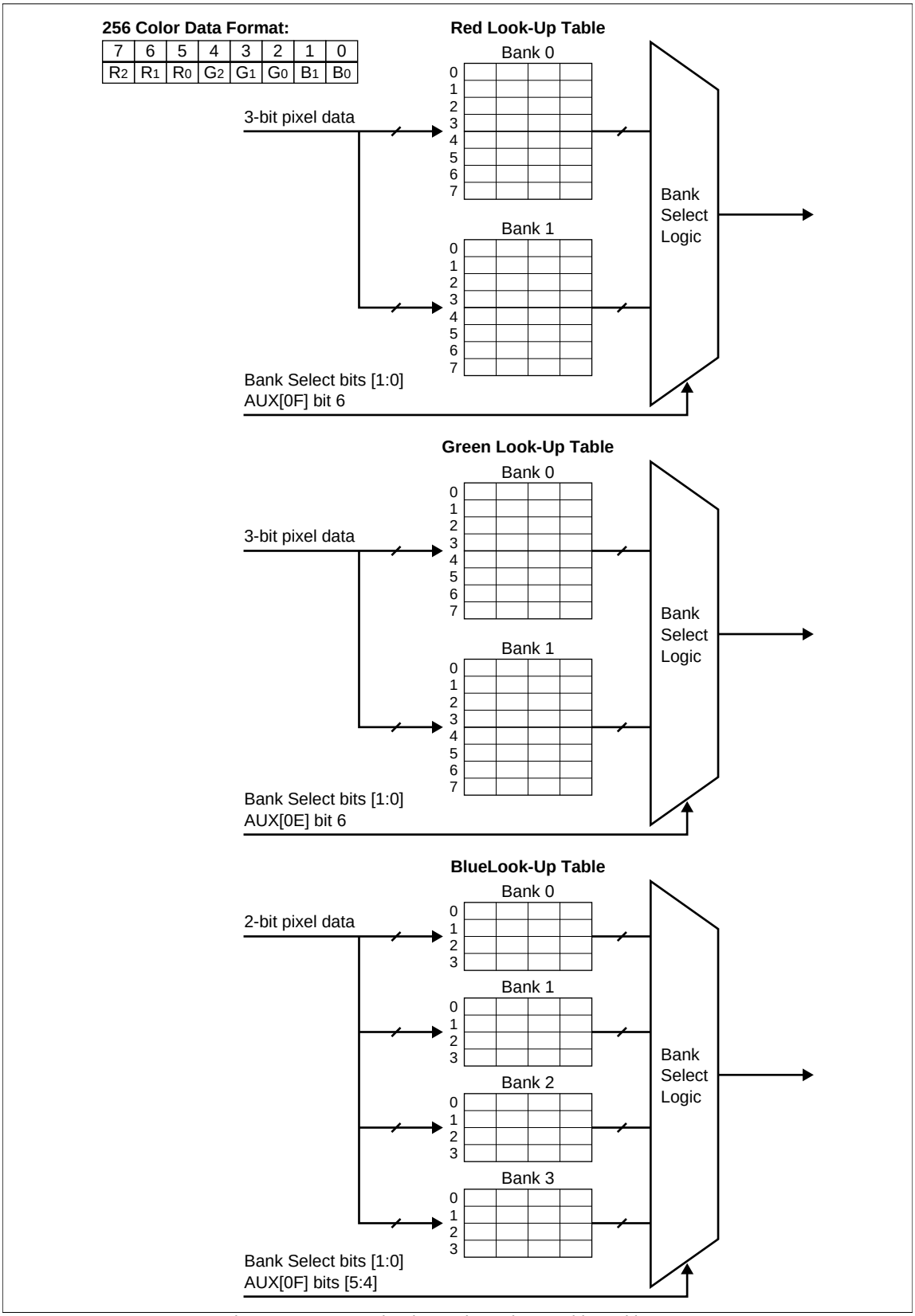



Figure 8-5 256-Level Color Mode Look-Up Table Architecture

## 8.3 Power Save Modes

Two software-controlled Power Save Modes have been incorporated into the S1D13503 to accommodate the important need for power reduction in the hand-held devices market. These modes can be enabled by setting the two Power Save bits (AUX[03] bits 7:6).

The various settings are:

Table 8-9 Power Save Mode Selection

| Bit 7 | Bit 6 | Mode Activated |
|---|---|---|
| 0 | 0 | Normal Operation |
| 0 | 1 | Power Save Mode 1 |
| 1 | 0 | Power Save Mode 2 |
| 1 | 1 | Reserved |

### Power Save Mode 1

Power Save Mode 1 has two states. Initially when set, the S1D13503 enters State 1. If no valid memory cycle is detected within 1, 2, or 4 clocks (input clock frequency dependent), the chip will enter State 2. The number of clocks of inactivity before entering State 2 is dependent on the display memory interface and the number of Gray shades.

**State 1**

- I/O read/write of all registers allowed
- Memory read/write allowed
- LCD outputs are either forced low (AUX[03] bit 5 = 0), or high impedance (AUX[03] bit 5 = 1)

**State 2**

The same as State 1 as well as:

- Master clock for MPU access is disabled

Once a valid memory read/write cycle is detected, the S1D13503 returns to State 1 where the MPU access is serviced. The transition from going from State 2 to State 1 requires 1, 2, or 4 clocks (as described above).

### Power Save Mode 2

- I/O read/write of all registers allowed
- Memory read/write is disabled
- Master clock for MPU access is disabled
- LCD outputs are either forced low (AUX[03] bit 5 = 0), or high impedance (AUX[03] bit 5 = 1)
- Internal oscillator is disabled.

### Power Save Mode Function Summary

Table 8-10 Power Save Mode Function Summary

| Function | Power Save Mode (PSM) | | | |
|---|---|---|---|---|
| | Normal (Active) | PSM1 | | PSM2 |
| | | State 1 | State 2 | |
| Display Active? | Yes | No | No | No |
| I/O Access Possible? | Yes | Yes | Yes | Yes |
| Memory Access Possible? | Yes | Yes | No | No |
| Sequence Controller Running? | Yes | No | No | No |
| Internal Oscillator Disabled? | No | No | No | Yes |

## *Pin States in Power Save Modes*

Table 8-11 Pin States in Power Save Modes

| Pin | Pin State | | | |
|---|---|---|---|---|
| | Normal (Active) | PSM1 | | PSM2 |
| | | State 1 | State 2 | |
| UD[3:0], LD[3:0], LP, XSCL, YD, WF/XSCL2 *(Note 1)* | Active | High Impedance | High Impedance | High Impedance |
| UD[3:0], LD[3:0], LP, XSCL, YD, WF/XSCL2 *(Note 2)* | Active | Forced Low | Forced Low | Forced Low |
| AB[19:0], DB[15:0] | Active | Active | Active | Active |
| IOR#, IOW# | Active | Active | Active | Active |
| MEMR#, MEMW# | Active | Active | Active | Active |
| RESET | Active | Active | Active | Active |

**Notes:** 1. Internal Register AUX[03], bit 5 = 1

2. Internal Register AUX[03], bit 5 = 0

# 9 DISPLAY MEMORY INTERFACE

## 9.1 SRAM Configurations Supported

### 8-Bit Mode

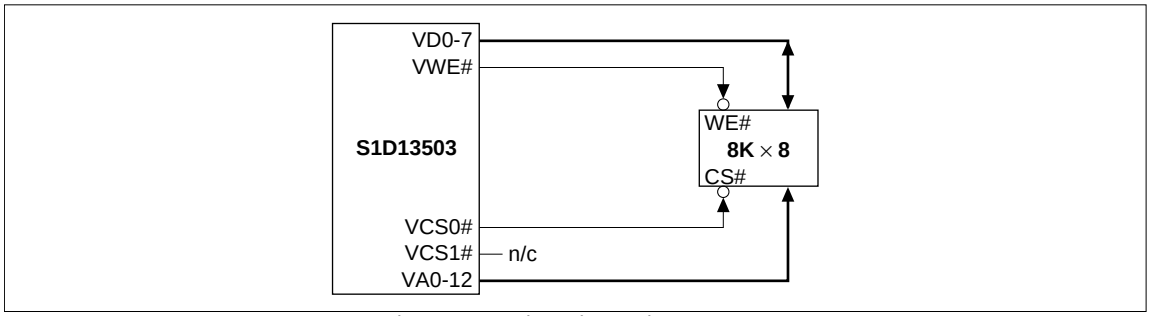



Figure 9-1 8-Bit Mode - 8K bytes SRAM
(Requires AUX[01] bit 0 = 0)

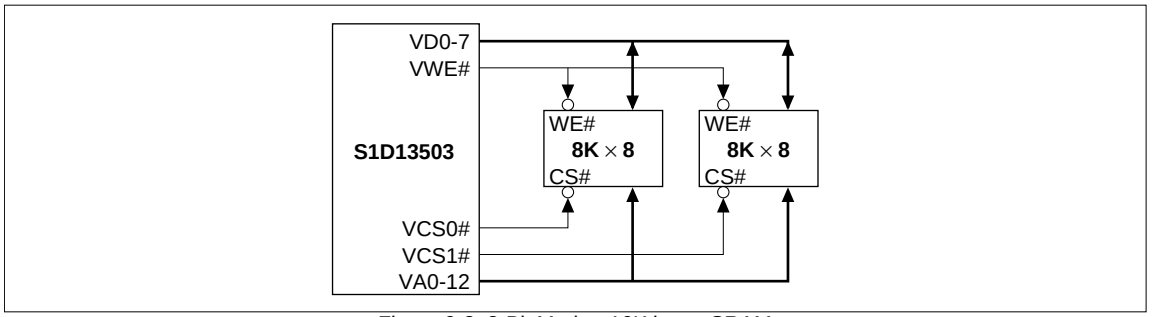



Figure 9-2 8-Bit Mode - 16K bytes SRAM
(Requires AUX[01] bit 0 = 0)

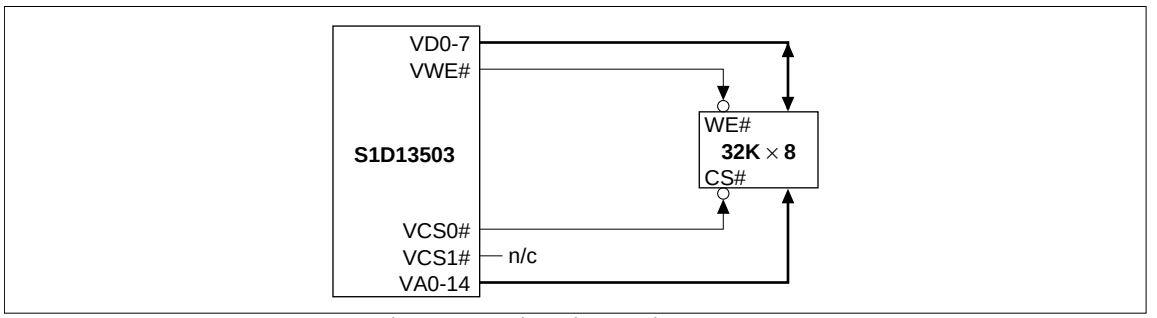



Figure 9-3 8-Bit Mode - 32K bytes SRAM
(Requires AUX[01] bit 0 = 1)

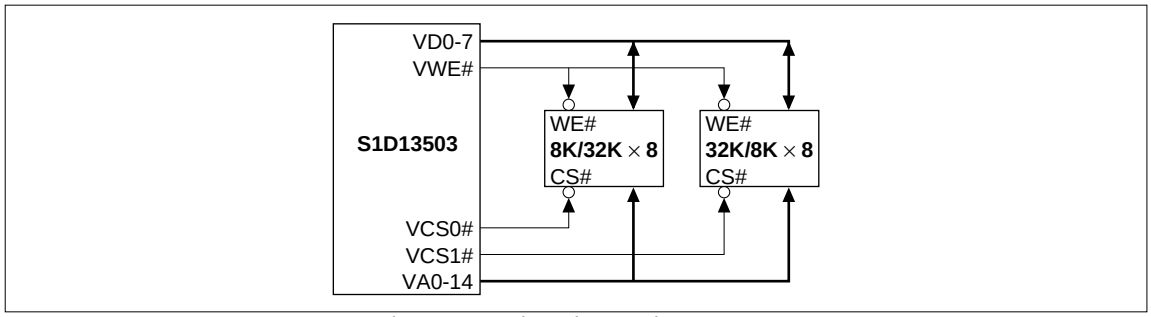



Figure 9-4 8-Bit Mode - 40K bytes SRAM
[Either (8K × 8 + 32K × 8) requiring AUX[01] bit 0 = 0 or (32K × 8 + 8K × 8) requiring AUX[01] bit 0 = 1]

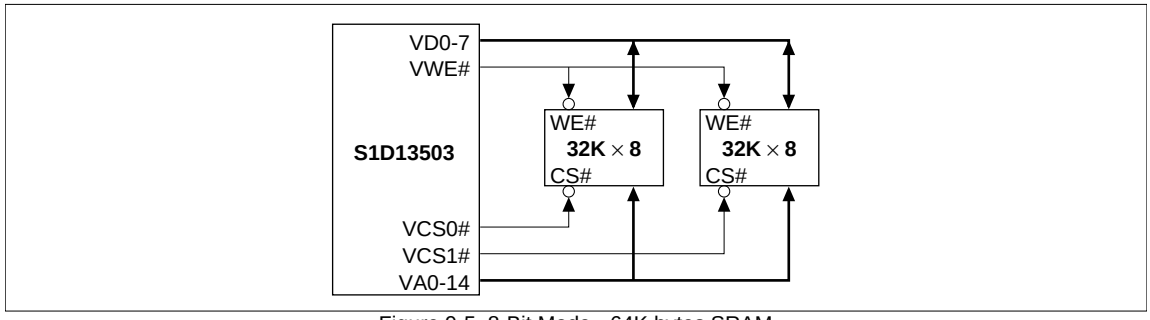



Figure 9-5 8-Bit Mode - 64K bytes SRAM
(Requires AUX[01] bit 0 = 1)

## *16-Bit Mode*

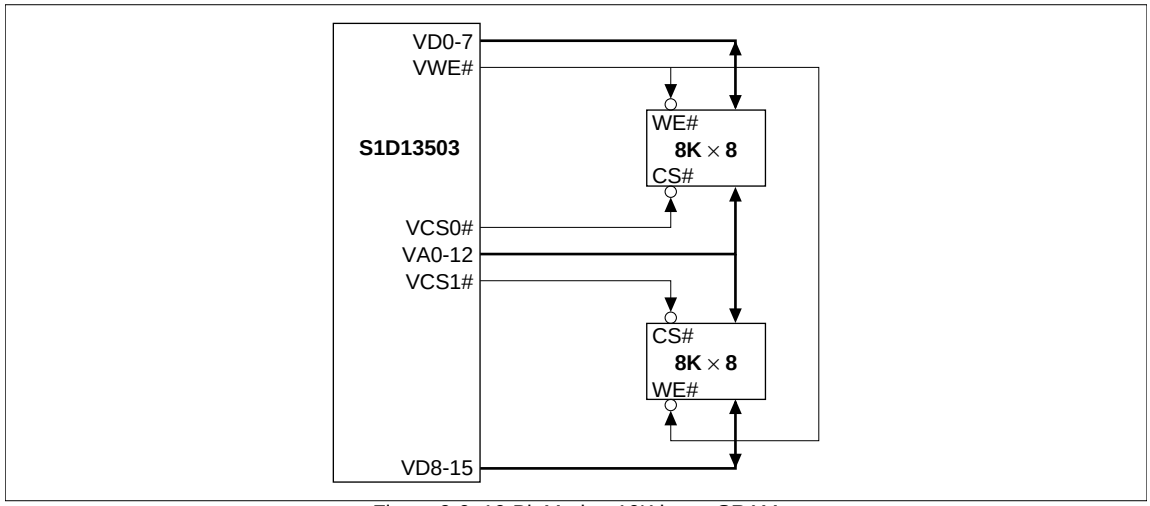



Figure 9-6 16-Bit Mode - 16K bytes SRAM

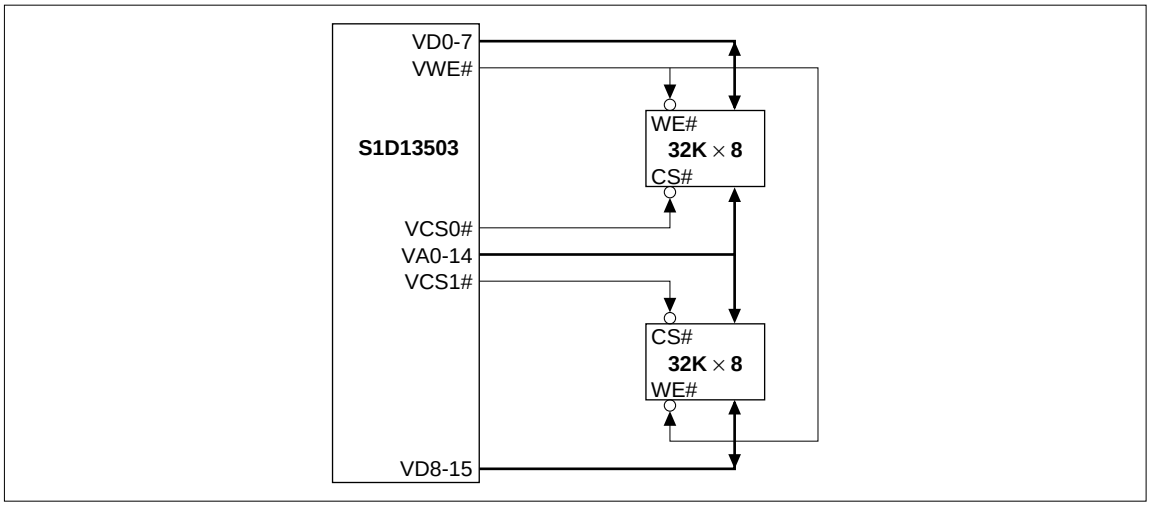



Figure 9-7 16-Bit Mode - 64K bytes SRAM

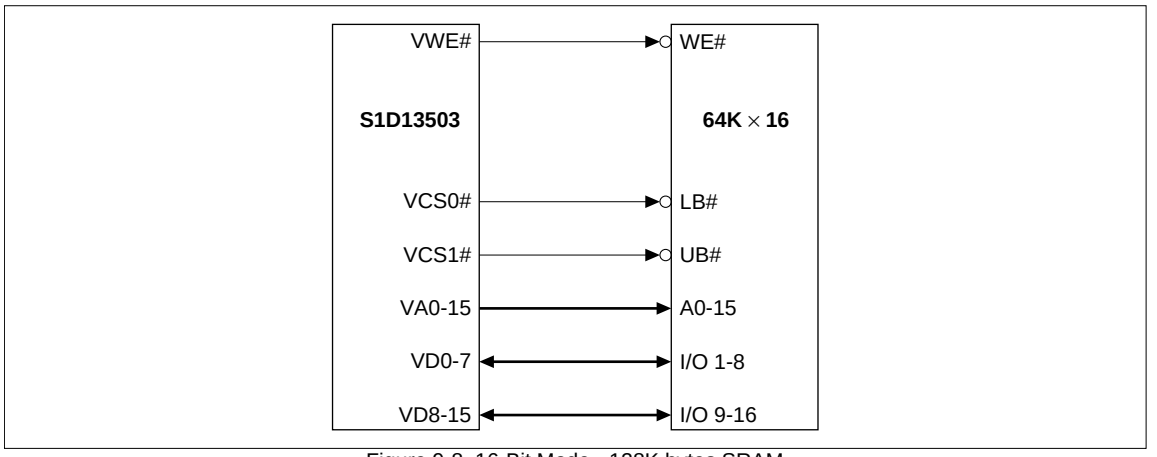



Figure 9-8 16-Bit Mode - 128K bytes SRAM

## 9.2 SRAM Access Time

### 8-Bit Display Memory Interface

Table 9-1 8-Bit Display Memory Interface SRAM Access Time

| Display Mode | 3V/3.3V | 5V |
|---|---|---|
| 16-level gray shades / 16-level colors | Access time ≤ 1 / fosc - 40ns | Access time ≤ 1 / fosc - 25ns |
| 4-level gray shades / 4-level colors | Access time ≤ 2 / fosc - 40ns | Access time ≤ 2 / fosc - 25ns |
| Black-and-White (BW) | Access time ≤ 2 / fosc - 40ns | Access time ≤ 2 / fosc - 25ns |

### 16-Bit Display Memory Interface

Table 9-2 16-Bit Display Memory Interface SRAM Access Time

| Display Mode | 3V/3.3V | 5V |
|---|---|---|
| 256-level colors | Access time ≤ 1 / fosc - 40ns | Access time ≤ 1 / fosc - 25ns |
| 16-level gray shades / 16-level colors | Access time ≤ 2 / fosc - 40ns | Access time ≤ 2 / fosc - 25ns |
| 4-level gray shades / 4-level colors | Access time ≤ 4 / fosc - 40ns | Access time ≤ 4 / fosc - 25ns |
| Black-and-White (BW) | Access time ≤ 4 / fosc - 40ns | Access time ≤ 4 / fosc - 25ns |

## 9.3 Frame Rate Calculation

### For Single Panel

Black-and-White (BW) Display Mode:

$$\text{FrameRate} = \frac{2 \times \text{fosc}}{(\text{HorizontalPixels} + \text{PHNDP} + \text{DHNDP}) \times (\text{VerticalLines} + 4)}$$

All Other Display Modes:

$$\text{FrameRate} = \frac{\text{fosc}}{(\text{HorizontalPixels} + \text{PHNDP} + \text{DHNDP}) \times (\text{VerticalLines} + 4)}$$

### For Dual Panel

Black-and-White (BW) Display Mode:

$$\text{FrameRate} = \frac{2 \times \text{fosc}}{(\text{HorizontalPixels} + \text{PHNDP} + \text{DHNDP}) \times 2 \times \left(\frac{\text{VerticalLines}}{2} + 2\right)}$$

All Other Display Modes:

$$\text{FrameRate} = \frac{\text{fosc}}{(\text{HorizontalPixels} + \text{PHNDP} + \text{DHNDP}) \times 2 \times \left(\frac{\text{VerticalLines}}{2} + 2\right)}$$

Where DHNDP is Default Horizontal Non-Display Period in term of pixels:
DHNDP = 16 pixels in gray shade display modes, and
DHNDP = 32 pixels in BW display mode and in color display modes.
Where PHNDP is Programmable Horizontal Non-Display Period in term of pixels:
PHNDP = 0 pixel when AUX[0C] = 0, and

$$\text{PHNDP} = \frac{(\text{AUX[0C]} + 1) \times (\text{MemoryInterfaceWidth})}{(\text{BitsPerPixel})}$$ pixels when AUX[0C] not equal to zero.

## 9.4 Memory Size Calculation

$$\text{Memory Size (bytes)} = \frac{(\text{HorizontalPixels}) \times (\text{VerticalLines}) \times (\text{BitsPerPixel})}{8}$$

Example: For a 640 × 480, 4 gray shades (2 bits-per-pixel) system:

$$\text{Memory Size (bytes)} = \frac{(640) \times (480) \times (2)}{8} = 76800\text{bytes} = 75\text{Kbytes}$$

## 9.5 Memory Size Requirement

The following tables summarize the preceding information (formulae).

Input clock (fOSC) is limited by SRAM access time depending on the display mode and display memory interface that is being used. As a result, different resolutions will have different input clock and memory requirements for a particular frame rate. Tables 9-3 through 9-5 summarize the minimum memory size and access time requirements for various resolutions at a particular input clock along with the corresponding frame rates.

Table 9-3 Memory Size Requirement: Number of Horizontal Pixels = 640

| | | Number of Horizontal Pixels = 640 | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Display Mode | Black-and-White (BW) (1 bit-per-pixel) | | | 4 Grays / 4 Colors (2 bits-per-pixel) | | | 16 Grays / 16 Colors (4 bits-per-pixel) | | | 256 Colors (8 bits-per-pixel) | | | | |
| | | Condition | AUX[0C] = AUX[02] | | | AUX[0C] = 0 | | | AUX[0C] = 0 | | | AUX[0C] = 0 | | | Example | | |
| | | Display Memory Interface | Size (KB) | Access Time | | Size (KB) | Access Time | | Size (KB) | Access Time | | Size (KB) | Access Time | | Input Clock (fosc) | Frame Rate | |
| | | | | 3V/ 3.3V | 5V | | 3V/ 3.3V | 5V | | 3V/ 3.3V | 5V | | 3V/ 3.3V | 5V | | BW / Gray | Color |
| Number of Vertical Lines | 480 | 8-bit 16-bit | 37.5 | 40 ns 125 ns | 55 ns 140 ns | 75 | (2) 125 ns | (2) 140 ns | 150 | (1) | (1) | 300 | (1) | (1) | 24 MHz | 76 Hz | 74 Hz |
| | 400 | 8-bit 16-bit | 32 | 60 ns 160 ns | 75 ns 175 ns | 62.5 | 60 ns 160 ns | 75 ns 175 ns | 125 | (2) 60 ns | (2) 75 ns | 250 | (1) | (1) | 20 MHz | 75 Hz | 74 Hz |
| | 320 | 8-bit 16-bit | 25 | 85 ns 210 ns | 100 ns 225 ns | 50 | 85 ns 210 ns | 100 ns 225 ns | 100 | (2) 85 ns | (2) 100 ns | 200 | (1) | (1) | 16 MHz | 75 Hz | 73 Hz |
| | 256 | 8-bit 16-bit | 20 | 125 ns 290 ns | 140 ns 305 ns | 40 | 125 ns 290 ns | 140 ns 305 ns | 80 | (2) 125 ns | (2) 140 ns | 160 | (1) | (1) | 12 MHz | 70 Hz | 69 Hz |
| | 240 | 8-bit 16-bit | 19 | 125 ns 290 ns | 140 ns 305 ns | 37.5 | 125 ns 290 ns | 140 ns 305 ns | 75 | (2) 125 ns | (2) 140 ns | 150 | (1) | (1) | 12 MHz | 75 Hz | 73 Hz |
| | 200 | 8-bit 16-bit | 16 | 160 ns 360 ns | 175 ns 375 ns | 32 | 160 ns 360 ns | 175 ns 375 ns | 62.5 | 60 ns 160 ns | 75 ns 175 ns | 125 | (2)(3) 60 ns | (2)(3) 75 ns | 10 MHz | 75 Hz | 73 Hz |

(1) Memory more than 128KB cannot be supported by S1D13503.
(2) Memory more than 64KB can only be supported through 16-bit display memory interface.
(3) 256 color mode must use 16-bit display memory interface.
∗ KB = K byte = 1024 bytes

Table 9-4 Memory Size Requirement: Number of Horizontal Pixels = 480

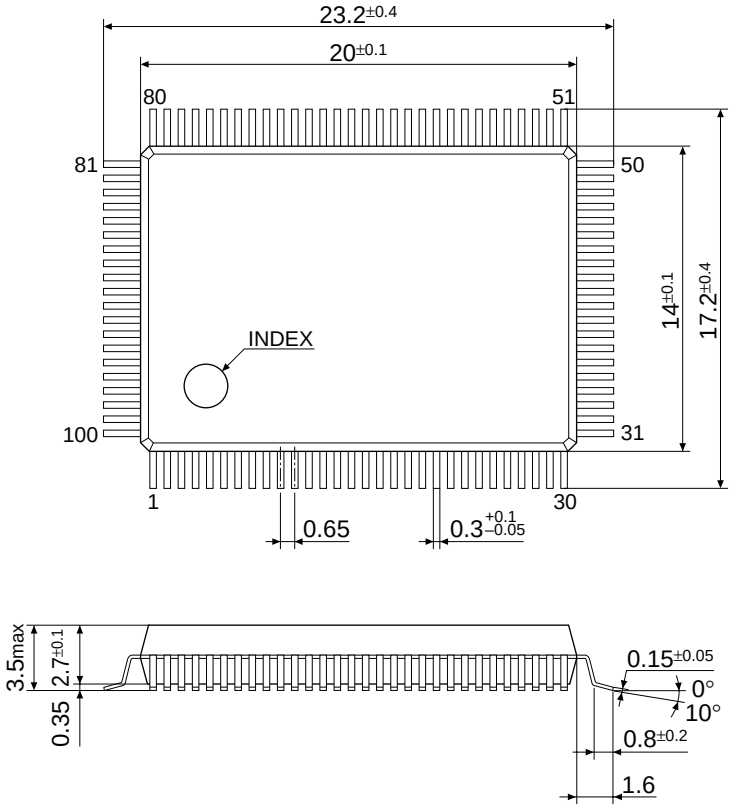
| | | Number of Horizontal Pixels = 480 | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Display Mode | Black-and-White (BW) (1 bit-per-pixel) | | | 4 Grays / 4 Colors (2 bits-per-pixel) | | | 16 Grays / 16 Colors (4 bits-per-pixel) | | | 256 Colors (8 bits-per-pixel) | | | Example | | |
| | | Condition | AUX[0C] = AUX[02] | | | AUX[0C] = 0 | | | AUX[0C] = 0 | | | AUX[0C] = 0 | | | | | |
| | | Display Memory Interface | Size (KB) | Access Time | | Size (KB) | Access Time | | Size (KB) | Access Time | | Size (KB) | Access Time | | Input Clock (fosc) | Frame Rate | |
| | | | | 3V/3.3V | 5V | | 3V/3.3V | 5V | | 3V/3.3V | 5V | | 3V/3.3V | 5V | | BW / Gray | Color |
| Number of Vertical Lines | 480 | 8-bit 16-bit | 29 | 70 ns 180 ns | 85 ns 195 ns | 57 | 70 ns 180 ns | 85 ns 195 ns | 113 | (2) 70 ns | (2) 85 ns | 225 | (1) | (1) | 18 MHz | 75 Hz | 73 Hz |
| | 400 | 8-bit 16-bit | 23.5 | 100 ns 240 ns | 115 ns 255 ns | 47 | 100 ns 240 ns | 115 ns 255 ns | 94 | (2) 100 ns | (2) 115 ns | 188 | (1) | (1) | 14 MHz | 70 Hz | 68 Hz |
| | 320 | 8-bit 16-bit | 19 | 125 ns 290 ns | 140 ns 305 ns | 37.5 | 125 ns 290 ns | 140 ns 305 ns | 75 | (2) 125 ns | (2) 140 ns | 150 | (1) | (1) | 12 MHz | 75 Hz | 72 Hz |
| | 256 | 8-bit 16-bit | 15 | 160 ns 360 ns | 175 ns 375 ns | 30 | 160 ns 360 ns | 175 ns 375 ns | 60 | 60 ns 160 ns | 75 ns 175 ns | 120 | (2)(3) 60 ns | (2)(3) 75 ns | 10 MHz | 77 Hz | 75 Hz |
| | 240 | 8-bit 16-bit | 14.5 | 210 ns 460 ns | 225 ns 475 ns | 29 | 210 ns 460 ns | 225 ns 475 ns | 57 | 85 ns 210 ns | 100 ns 225 ns | 113 | (2)(3) 85 ns | (2)(3) 100 ns | 8 MHz | 66 Hz | 64 Hz |
| | 200 | 8-bit 16-bit | 12 | 210 ns 460 ns | 225 ns 475 ns | 23.5 | 210 ns 460 ns | 225 ns 475 ns | 47 | 85 ns 210 ns | 100 ns 225 ns | 94 | (2)(3) 85 ns | (2)(3) 100 ns | 8 MHz | 79 Hz | 77 Hz |

Table 9-5 Memory Size Requirement: Number of Horizontal Pixels = 320

| | | Number of Horizontal Pixels = 320 | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Display Mode | Black-and-White (BW) (1 bit-per-pixel) | | | 4 Grays / 4 Colors (2 bits-per-pixel) | | | 16 Grays / 16 Colors (4 bits-per-pixel) | | | 256 Colors (8 bits-per-pixel) | | | Example | | |
| | | Condition | AUX[0C] = AUX[02] | | | AUX[0C] = 0 | | | AUX[0C] = 0 | | | AUX[0C] = 0 | | | | | |
| | | Display Memory Interface | Size (KB) | Access Time | | Size (KB) | Access Time | | Size (KB) | Access Time | | Size (KB) | Access Time | | Input Clock (fosc) | Frame Rate | |
| | | | | 3V/3.3V | 5V | | 3V/3.3V | 5V | | 3V/3.3V | 5V | | 3V/3.3V | 5V | | BW / Gray | Color |
| Number of Vertical Lines | 480 | 8-bit 16-bit | 19 | 125 ns 290 ns | 140 ns 305 ns | 37.5 | 125 ns 290 ns | 140 ns 305 ns | 75 | (2) 125 ns | (2) 140 ns | 150 | (1) | (1) | 12 MHz | 74 Hz | 70 Hz |
| | 400 | 8-bit 16-bit | 16 | 160 ns 360 ns | 175 ns 375 ns | 32 | 160 ns 360 ns | 175 ns 375 ns | 62.5 | 60 ns 160 ns | 75 ns 175 ns | 125 | (2)(3) 60 ns | (2)(3) 75 ns | 10 MHz | 74 Hz | 70 Hz |
| | 320 | 8-bit 16-bit | 12.5 | 210 ns 460 ns | 225 ns 475 ns | 25 | 210 ns 460 ns | 225 ns 475 ns | 50 | 85 ns 210 ns | 100 ns 225 ns | 100 | (2)(3) 85 ns | (2)(3) 100 ns | 8 MHz | 73 Hz | 70 Hz |
| | 256 | 8-bit 16-bit | 10 | 290 ns 625 ns | 305 ns 640 ns | 20 | 290 ns 625 ns | 305 ns 640 ns | 40 | 125 ns 290 ns | 140 ns 305 ns | 80 | (2)(3) 125 ns | (2)(3) 140 ns | 6 MHz | 69 Hz | 66 Hz |
| | 240 | 8-bit 16-bit | 9.5 | 290 ns 625 ns | 305 ns 640 ns | 19 | 290 ns 625 ns | 305 ns 640 ns | 37.5 | 125 ns 290 ns | 140 ns 305 ns | 75 | (2)(3) 125 ns | (2)(3) 140 ns | 6 MHz | 73 Hz | 70 Hz |
| | 200 | 8-bit 16-bit | 8 | 360 ns 760 ns | 375 ns 775 ns | 16 | 360 ns 760 ns | 375 ns 775 ns | 32 | 160 ns 360 ns | 175 ns 375 ns | 62.5 | (2)(3) 160 ns | (2)(3) 175 ns | 5 MHz | 73 Hz | 70 Hz |

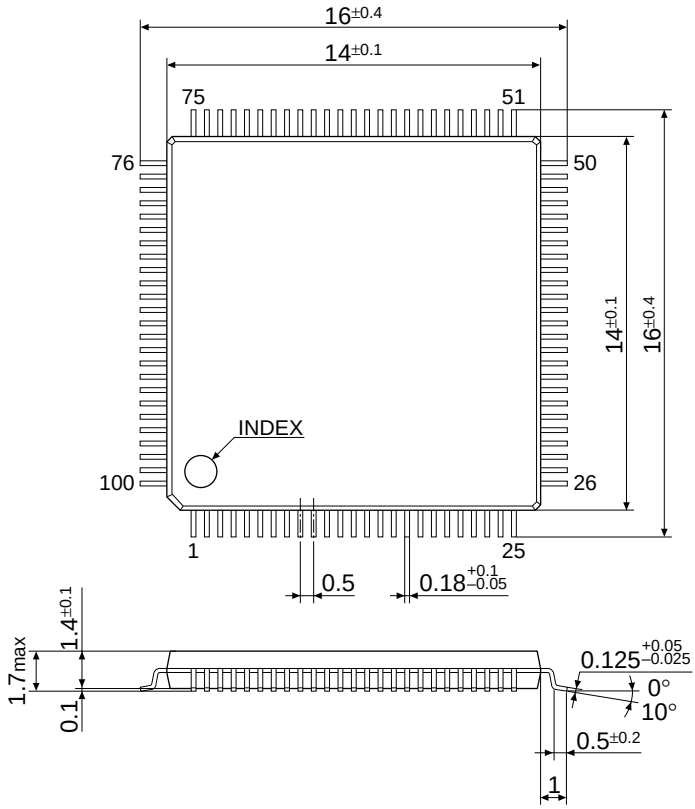
(1) Memory more than 128KB cannot be supported by S1D13503.
(2) Memory more than 64KB can only be supported through 16-bit display memory interface.
(3) 256 color mode must use 16-bit display memory interface.
∗ KB = K byte = 1024 bytes

# 10 MECHANICAL DATA




All dimensions in mm

Figure 10-1 Mechanical Drawing QFP5-100-S2




All dimensions in mm

Figure 10-2 Mechanical Drawing QFP15-100-STD

**THIS PAGE IS BLANK.**

# S1D13503 Series
## Dot Matrix Graphic LCD Controller
## Programmer's Reference

# PROGRAMMER'S REFERENCE

## Table of Contents

## List of Figures



## List of Tables

# 1 INTRODUCTION

The purpose of this guide is to demonstrate how to program the S1D13503 LCD controller, with reference made to the S5U13503P00C evaluation board. The first half of this guide presents the basic concepts of LCD controllers.

The second half of this guide presents programming examples which are combined in a simple menu-driven program. Most of the program is written in the ‘C’ programming language, with some parts written in 8086 assembly.

# 2 INITIALIZING THE S1D13503

This section presents two examples to show how to initialize the S1D13503 registers and write a pixel to the display. Code to initialize the S1D13503 is provided in Section 7.2, *"Initialization Code"* on page 46.

The following examples describe values written to registers.

- A "panel specific" value is one required for the given type of panel. Such a value must never change after initialization of all registers.
- An "implementation specific" value is one required for the hardware implementation of the S1D13503. Such a value must never change after initialization of all registers. Refer to the *Hardware Functional Specification* and *S5U13503P00C Evaluation Board User's Manual* for more information on hardware implementation issues.
- An "application specific" value is one that can be changed by the program after initialization of all registers.

### Example 1

**Initialize the registers for a 256 color 320 x 240 single panel LCD with 128K of display memory. Afterwards write one pixel to the top left corner of the display.**

1. Program S1D13503 Registers in the following order with the data supplied.

| AUX Register | Data (in Binary) | Notes | See Also |
|---|---|---|---|
| AUX[00h] | 0000 0000 | • bits 7 and 6 must be zero | |
| AUX[01h] | 0010 1001 | • bit 7 = display off (application specific; the recommended procedure is to turn this bit off during register initialization and afterwards turn this bit on)<br>• bit 6 = single panel (panel specific)<br>• bit 5 = XSCL is masked (panel specific)<br>• bit 4 = LCDE = LCDENB pin = set to disable specific power supply design (for S5U13503P00C, set bit to 0 to disable power supply) (application specific; the recommended procedure is to disable the power supply during register initialization and afterwards enable the power supply)<br>• bit 3 = N/A for 256 colors (application specific)<br>• bit 2 = 4 bit LCD data width when combined with AUX[03] bit 3 (panel specific)<br>• bit 1 = 16 bit Memory Interface (implementation specific)<br>• bit 0 = RAMS ignored (implementation specific) | |
| AUX[02h] | 1001 1111 | • bits 7–0 = bits 7–0 of Line Byte Count<br>bit 8 of Line Byte Count is bit 0 of AUX[03h] | see Note A at end of Table for calculation |
| AUX[03h] | 0000 0110 | • bits 7–6 = Power Save Mode 0 (application specific - for normal operation set to 00b)<br>• bit 5 = LCD interface signals forced low during Power Save (implementation and panel specific)<br>• bit 4 = no LUT bypass (application specific)<br>• bit 3 = 4 bit LCD data width when combined with AUX[01] bit 2 (panel specific)<br>• bit 2 = 256 color mode (application specific)<br>• bit 1 = color panel attached (panel specific)<br>• bit 0 = bit 8 of Line Byte Count (panel specific, see AUX[02h]) | see Section 5.6, *"Power Saving"* on page 40 |
| AUX[04h] | 1110 1111 | • bits 7–0 = bits 7–0 of Total Display Line Count<br>bits 9–8 of Total Display Line Count in bits 1–0 of AUX[05h] | see Note B and C at end of Table for calculation |
| AUX[05h] | 0000 0000 | • bits 7–2: 0 = WF output toggles every frame (panel specific)<br>• bits 1–0 = bits 9–8 of Total Display Line Count (panel specific, see AUX[04h]) | |

| AUX Register | Data (in Binary) | Notes | See Also |
|---|---|---|---|
| AUX[06h]<br>AUX[07h] | 0000 0000<br>0000 0000 | • bits 15–0 of Screen 1 Display Start Address - normally Screen 1 Start Address = 0000h (application and panel specific)<br>bits 7–0 are in AUX[06h] and bits 15–8 are in AUX[07h]<br>when 0000h, Screen 1 Display Start Address is located at D000:0000h, bank 0, on the S5U13503P00C | see *"S5U13503P00C Evaluation Board Display Memory"* on page 24 and Section 4.1, *"Registers"* on page 22 |
| AUX[08h]<br>AUX[09h] | 0000 0000<br>0000 0000 | • bits 15–0 of Screen 2 Display Start Address - normally Screen 2 Start Address = 0000h (application and panel specific)<br>bits 7–0 are in AUX[08h] and bits 15–8 are in AUX[09h]<br>when 0000h, Screen 2 Display Start Address is located at D000:0000h, bank 0, on the S5U13503P00C | see *"S5U13503P00C Evaluation Board Display Memory"* on page 24 and Section 4.1, *"Registers"* on page 22 |
| AUX[0Ah] | 1110 1111 | • bits 7–0 = bits 7–0 of Screen 1 Display Line Count<br>bits 9–8 of Screen 1 Display Line Count in bits 1–0 of AUX[0Bh]<br>Screen 1 Display Line Count is typically the same as Total Display Line Count (AUX[0Ah] = AUX[04h], bits 1–0 of AUX[0Bh] = bits 1–0 of AUX[05h]) | see Section 5.4, *"Split Screen"* on page 31 |
| AUX[0Bh] | 0000 0000 | • bits 7–2 = don't care; recommend clearing bits<br>• bits 1–0 = bits 9–8 of Screen 1 Display Line Count (application specific, see AUX[0Ah]) | |
| AUX[0Ch] | 0000 0000 | normally programmed to 00h (panel specific)<br>• bits 7–0 = use fixed default non-display period | |
| AUX[0Dh] | 0000 0000 | normally programmed to 00h (normal)<br>• bits 7–0 = no address pitch adjustment when 0 | see Section 5.1, *"Virtual Displays"* on page 27 |
| AUX[0Eh] | 0000 0000 | select palette address<br>• bits 7–6 = green bank 0 (application specific)<br>• bits 5–4 = auto increment palette R/W access (application specific)<br>• bits 3–0 = palette address (application specific) | |
| AUX[0Fh] | 0000 0000 | write Red data<br>• bits 7–6 = red bank 0 (application specific)<br>• bits 5–4 = blue bank 0 (application specific)<br>• bits 3–0 = palette data (application specific) | |
| AUX[0Fh] | 0000 0000 | write Green data | |
| AUX[0Fh] | 0000 0000 | write Blue data | |
| AUX[0Eh] | 0000 0001 | increment palette address | |
| AUX[0Fh] | 0000 0010 | write Red data | |
| AUX[0Fh] | 0000 0010 | write Green data | |
| AUX[0Fh] | 0000 0101 | write Blue data | |
| AUX[0Eh] | 0000 0010 | increment palette address | |
| AUX[0Fh] | 0000 0100 | write Red data | |
| AUX[0Fh] | 0000 0100 | write Green data | |
| AUX[0Fh] | 0000 1010 | write Blue data | |
| AUX[0Eh] | 0000 0011 | increment palette address | |
| AUX[0Fh] | 0000 0110 | write Red data | |
| AUX[0Fh] | 0000 0110 | write Green data | |
| AUX[0Fh] | 0000 1111 | write Blue data | |
| AUX[0Eh] | 0000 0100 | increment palette address | |
| AUX[0Fh] | 0000 1001 | write Red data | |
| AUX[0Fh] | 0000 1001 | write Green data | |
| AUX[0Fh] | 0000 1111 | write Blue data | |
| AUX[0Eh] | 0000 0101 | increment palette address | |
| AUX[0Fh] | 0000 1011 | write Red data | |
| AUX[0Fh] | 0000 1011 | write Green data | |
| AUX[0Fh] | 0000 1010 | write Blue data | |
| AUX[0Eh] | 0000 0110 | increment palette address | |
| AUX[0Fh] | 0000 1101 | write Red data | |
| AUX[0Fh] | 0000 1101 | write Green data | |
| AUX[0Fh] | 0000 0101 | write Blue data | |
| AUX[0Eh] | 0000 0111 | increment palette address | |
| AUX[0Fh] | 0000 1111 | write Red data | |

| AUX Register | Data (in Binary) | Notes | See Also |
|---|---|---|---|
| AUX[0Fh] | 0000 1111 | write Green data | |
| AUX[0Fh] | 0000 0000 | write Blue data | |
| AUX[0Eh] | 0000 1000 | increment palette address | |
| AUX[0Fh] | 0000 1111 | write Red data | |
| AUX[0Fh] | 0000 1111 | write Green data | |
| AUX[0Fh] | 0000 0001 | write Blue data | |
| AUX[0Eh] | 0000 1001 | increment palette address | |
| AUX[0Fh] | 0000 1101 | write Red data | |
| AUX[0Fh] | 0000 1101 | write Green data | |
| AUX[0Fh] | 0000 0110 | write Blue data | |
| AUX[0Eh] | 0000 1010 | increment palette address | |
| AUX[0Fh] | 0000 1011 | write Red data | |
| AUX[0Fh] | 0000 1011 | write Green data | |
| AUX[0Fh] | 0000 1001 | write Blue data | |
| AUX[0Eh] | 0000 1100 | increment palette address | |
| AUX[0Fh] | 0000 0110 | write Red data | |
| AUX[0Fh] | 0000 0110 | write Green data | |
| AUX[0Fh] | 0000 1101 | write Blue data | |
| AUX[0Eh] | 0000 1101 | increment palette address | |
| AUX[0Fh] | 0000 0100 | write Red data | |
| AUX[0Fh] | 0000 0100 | write Green data | |
| AUX[0Fh] | 0000 1001 | write Blue data | |
| AUX[0Eh] | 0000 1110 | increment palette address | |
| AUX[0Fh] | 0000 0010 | write Red data | |
| AUX[0Fh] | 0000 0010 | write Green data | |
| AUX[0Fh] | 0000 0100 | write Blue data | |
| AUX[0Eh] | 0000 1111 | select palette address | |
| AUX[0Fh] | 0000 0000 | write Red data | |
| AUX[0Fh] | 0000 0000 | write Green data | |
| AUX[0Fh] | 0000 0010 | write Blue data | |
| AUX[01h] | 1011 1001 | program Mode Register bit DISP to 1, and set LCDE to enable power supply<br>1001 0000b 'OR' {original value for AUX[01h]}<br>• b7 = display on (application specific)<br>• b4 = LCDE = LCDENB pin = set to enable specific power supply design (for S5U13503P00C, set bit to 1 to enable power supply) (application specific) | |

**Note:** A

$$\text{Line Byte Count} = \left(\frac{\text{Bits Per Pixel}}{\text{Memory Interface Width}} \times \text{Horizontal Resolution}\right) - 1$$

$$= \left(\frac{8}{16} \times 320\right) - 1 = 159 = 9\text{Fh}$$

B Single Panel

$$\text{Total Display Line Count} = \text{Number of Display Lines} - 1 = 240 - 1 = 239 = 0\text{EFh}$$

C Dual Panel

$$\text{Total Display Line Count} = \frac{\text{Number of Display Lines}}{2} - 1$$

2. Write one pixel to the top left corner of display memory.
   If the S5U13503P00C evaluation board is used in indexed I/O mode, there are two video memory banks which begin at D000:0000 (2 banks × 64K per bank; see the following note). If the base port address is 310h, then read from port address 312h. Next, write 0FFh to location D000:0000h; this will be seen as a white pixel at the top left corner of the display.

**Note:** The S5U13503P00C evaluation board maps the 128K of display memory into two banks of 64K, starting at D000:0000. This permits a VGA card to work along with the S5U13503P00C card. Bank 0 represents the first 64K of display memory, and is selected by reading from the base port address + 2. Bank 1 represents the second 64K of display memory, and is selected by writing to the base port address + 2. The values read from or written to the base port address + 2 are not important; only the action of reading or writing is significant. This method of memory banking will only work if the S5U13503P00C is set for indexed port I/O and is specific to this board.

### Example 2

**Initialize the registers for a 4 gray shade 640 x 480 dual panel LCD with 128K of display memory. Afterwards write one pixel to the top left corner of the display's second panel.**

1. Program S1D13503 Registers in the following order with the data supplied.

| AUX Register | Data (in Binary) | Notes | See Also |
|---|---|---|---|
| AUX[00h] | 0000 0000 | • bits 7 and 6 must be zero | |
| AUX[01h] | 0100 0101 | • bit 7 = display off (application specific; the recommended procedure is to turn this bit off during register initialization and afterwards turn this bit on)<br>• bit 6 = dual panel (panel specific)<br>• bit 5 = XSCL not masked (panel specific)<br>• bit 4 = LCDE = LCDENB pin = set to disable specific power supply design (for S5U13503P00C, set bit to 0 to disable power supply) (application specific; the recommended procedure is to disable the power supply during register initialization and afterwards enable the power supply)<br>• bit 3 = 4 grays when combined with AUX[03] bits 1 and 2 (application specific)<br>• bit 2 = 8 bit LCD data width (panel specific)<br>• bit 1 = 16 bit Memory Interface (implementation specific)<br>• bit 0 = RAMS ignored (implementation specific) | |
| AUX[02h] | 0100 1111 | • bits 7–0 = bits 7–0 of Line Byte Count<br>bit 8 of Line Byte Count is bit 0 of AUX[03h] | see Note A at end of Table for calculation |
| AUX[03h] | 0000 0000 | • bits 7–6 = Power Save Mode 0 (application specific - for normal operation set to 00b)<br>• bit 5 = LCD interface signals forced low during Power Save (implementation and panel specific)<br>• bit 4 = no LUT bypass (application specific)<br>• bit 3 = 4 bit LCD data width when combined with AUX[01] bit 2 (panel specific)<br>• bit 2 = 4/16 gray shade mode (application specific)<br>• bit 1 = monochrome panel attached (panel specific)<br>• bit 0 = bit 8 of Line Byte Count (panel specific, see AUX[02h]) | see Section 5.6, *"Power Saving"* on page 40 |
| AUX[04h] | 1110 1111 | • bits 7–0 = bits 7–0 of Total Display Line Count<br>bits 9–8 of Total Display Line Count in bits 1-0 of AUX[05h] | see Note B and C at end of Table for calculation |
| AUX[05h] | 0000 0000 | • bits 7–2: 0 = WF output toggles every frame (panel specific)<br>• bits 1–0 = bits 9–8 of Total Display Line Count (panel specific, see AUX[04h]) | |
| AUX[06h]<br>AUX[07h] | 0000 0000<br>0000 0000 | • bits 15–0 of Screen 1 Display Start Address - normally Screen 1 Start Address = 0000h (application and panel specific)<br>bits 7–0 are in AUX[06h] and bits 15–8 are in AUX[07h]<br>when 0000h, Screen 1 Display Start Address is located at D000:0000h, bank 0, on the S5U13503P00C | see *"S5U13503P00C Evaluation Board Display Memory"* on page 24 and Section 4.1, *"Registers"* on page 22 |
| AUX[08h]<br>AUX[09h] | 0000 0000<br>0100 1011 | • bits 15–0 of Screen 2 Display Start Address - normally Screen 2 Start Address = 4B00h (application and panel specific)<br>bits 7–0 are in AUX[08h] and bits 15–8 are in AUX[09h]<br>when 4B00h, Screen 2 Display Start Address is located at D000:9600h, bank 0, on the S5U13503P00C | see *"S5U13503P00C Evaluation Board Display Memory"* on page 24 and Section 4.1, *"Registers"* on page 22 |

| AUX Register | Data (in Binary) | Notes | See Also |
|---|---|---|---|
| AUX[0Ah] | 1110 1111 | • bits 7–0 = bits 7–0 of Screen 1 Display Line Count<br>bits 9–8 of Screen 1 Display Line Count in bits 1–0 of AUX[0Bh]<br>Screen 1 Display Line Count is typically the same as Total Display Line Count (AUX[0Ah] = AUX[04h], bits 1-0 of AUX[0Bh] = bits 1-0 of AUX[05h]) | see Section 5.4, *"Split Screen"* on page 31 |
| AUX[0Bh] | 0000 0000 | • bits 7–2 = don't care; recommend clearing bits<br>• bits 1–0 = bits 9–8 of Screen 1 Display Line Count (application specific, see AUX[0Ah]) | |
| AUX[0Ch] | 0000 0000 | normally programmed to 00h (panel specific)<br>• bits 7–0 = use fixed default non-display period | |
| AUX[0Dh] | 0000 0000 | normally programmed to 00h (normal)<br>• bits 7–0 = no address pitch adjustment when 0 | see Section 5.1, *"Virtual Displays"* on page 27 |
| AUX[0Eh] | 0000 0000 | select palette address<br>• bits 7–6 = green bank 0 (application specific)<br>• bits 5–4 = auto increment palette R/W access (application specific)<br>• bits 3–0 = palette address (application specific) | |
| AUX[0Fh] | 0000 0000 | write Red data<br>• bits 7–6 = red bank 0 (application specific)<br>• bits 5–4 = blue bank 0 (application specific)<br>• bits 3–0 = palette data (application specific) | |
| AUX[0Fh] | 0000 0000 | write Green data | |
| AUX[0Fh] | 0000 0000 | write Blue data | |
| AUX[0Eh] | 0000 0001 | increment palette address | |
| AUX[0Fh] | 0000 0010 | write Red data | |
| AUX[0Fh] | 0000 0010 | write Green data | |
| AUX[0Fh] | 0000 0101 | write Blue data | |
| AUX[0Eh] | 0000 0010 | increment palette address | |
| AUX[0Fh] | 0000 0100 | write Red data | |
| AUX[0Fh] | 0000 0100 | write Green data | |
| AUX[0Fh] | 0000 1010 | write Blue data | |
| AUX[0Eh] | 0000 0011 | increment palette address | |
| AUX[0Fh] | 0000 0110 | write Red data | |
| AUX[0Fh] | 0000 0110 | write Green data | |
| AUX[0Fh] | 0000 1111 | write Blue data | |
| AUX[0Eh] | 0000 0100 | increment palette address | |
| AUX[0Fh] | 0000 1001 | write Red data | |
| AUX[0Fh] | 0000 1001 | write Green data | |
| AUX[0Fh] | 0000 1111 | write Blue data | |
| AUX[0Eh] | 0000 0101 | increment palette address | |
| AUX[0Fh] | 0000 1011 | write Red data | |
| AUX[0Fh] | 0000 1011 | write Green data | |
| AUX[0Fh] | 0000 1010 | write Blue data | |
| AUX[0Eh] | 0000 0110 | increment palette address | |
| AUX[0Fh] | 0000 1101 | write Red data | |
| AUX[0Fh] | 0000 1101 | write Green data | |
| AUX[0Fh] | 0000 0101 | write Blue data | |
| AUX[0Eh] | 0000 0111 | increment palette address | |
| AUX[0Fh] | 0000 1111 | write Red data | |
| AUX[0Fh] | 0000 1111 | write Green data | |
| AUX[0Fh] | 0000 0000 | write Blue data | |
| AUX[0Eh] | 0000 1000 | increment palette address | |
| AUX[0Fh] | 0000 1111 | write Red data | |
| AUX[0Fh] | 0000 1111 | write Green data | |
| AUX[0Fh] | 0000 0001 | write Blue data | |
| AUX[0Eh] | 0000 1001 | increment palette address | |
| AUX[0Fh] | 0000 1101 | write Red data | |
| AUX[0Fh] | 0000 1101 | write Green data | |
| AUX[0Fh] | 0000 0110 | write Blue data | |
| AUX[0Eh] | 0000 1010 | increment palette address | |
| AUX[0Fh] | 0000 1011 | write Red data | |

| AUX Register | Data (in Binary) | Notes | See Also |
|---|---|---|---|
| AUX[0Fh] | 0000 1011 | write Green data | |
| AUX[0Fh] | 0000 1001 | write Blue data | |
| AUX[0Eh] | 0000 1100 | increment palette address | |
| AUX[0Fh] | 0000 0110 | write Red data | |
| AUX[0Fh] | 0000 0110 | write Green data | |
| AUX[0Fh] | 0000 1101 | write Blue data | |
| AUX[0Eh] | 0000 1101 | increment palette address | |
| AUX[0Fh] | 0000 0100 | write Red data | |
| AUX[0Fh] | 0000 0100 | write Green data | |
| AUX[0Fh] | 0000 1001 | write Blue data | |
| AUX[0Eh] | 0000 1110 | increment palette address | |
| AUX[0Fh] | 0000 0010 | write Red data | |
| AUX[0Fh] | 0000 0010 | write Green data | |
| AUX[0Fh] | 0000 0100 | write Blue data | |
| AUX[0Eh] | 0000 1111 | select palette address | |
| AUX[0Fh] | 0000 0000 | write Red data | |
| AUX[0Fh] | 0000 0000 | write Green data | |
| AUX[0Fh] | 0000 0010 | write Blue data | |
| AUX[01h] | 1101 0101 | program Mode Register bit DISP to 1, and set LCDE to enable power supply<br>1001 0000b 'OR' {original value for AUX[01h]}<br>• b7 = display on (application specific)<br>• b4 = LCDE = LCDENB pin = set to enable specific power supply design (for S5U13503P00C, set bit to 1 to enable power supply) (application specific) | |

**Note:** A

$$\text{Line Byte Count} = \left(\frac{\text{Bits Per Pixel}}{\text{Memory Interface Width}} \times \text{Horizontal Resolution}\right) - 1$$

$$= \left(\frac{2}{16} \times 640\right) - 1 = 79 = 4\text{Fh}$$

B Single Panel

$$\text{Total Display Line Count} = \text{Number of Display Lines} - 1$$

C Dual Panel

$$\text{Total Display Line Count} = \frac{\text{Number of Display Lines}}{2} - 1 = \frac{480}{2} - 1 = 239 = 0\text{EFh}$$

2. Write one pixel to the top left corner of the display's second panel.
   If the S5U13503P00C evaluation board is used in indexed mode, there are two video memory banks which begin at D000:0000 (2 banks × 64K per bank; see the note on page 5). If the base port address is 310h, then read from port address 312h. Next, write 0C0h to location D000:9600h; this will be seen as a white pixel at the top left corner of the display's second panel.

# 3 GRAY SHADES / COLORS AND LOOK-UP TABLES

This section discusses how the S1D13503 shows color and monochrome images on LCD panels.

## 3.1 Pixels

A pixel is physically stored in display memory as a series of bits. The more bits, the more colors the pixel can show.

Table 3-1 Number of Bits as Related to Colors

| Bits per Pixel | Levels of | |
|---|---|---|
| | Gray Shades | Colors |
| 1 | 2 | n/a |
| 2 | 4 | 4 |
| 4 | 16 | 16 |
| 8 | n/a | 256 |

The following sections show how these pixels are stored in display memory.

### *Memory Organization for One Bit Pixel (Black-and-White)*

To store one bit pixels, eight pixels are grouped into one byte of display memory as shown below:

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Pixel 0 Bit 0 | Pixel 1 Bit 0 | Pixel 2 Bit 0 | Pixel 3 Bit 0 | Pixel 4 Bit 0 | Pixel 5 Bit 0 | Pixel 6 Bit 0 | Pixel 7 Bit 0 |

Figure 3-1 Pixel Storage for 1 Bit (Black-and-White) in one byte of Display Memory

When these pixels are shown, Pixel 0 is seen to be left of Pixel 1, Pixel 1 is seen to be left of Pixel 2, and so on. One bit pixels are only available on monochrome panels, and can only be displayed in black-and-white (no Look-Up Table is used).

### *Memory Organization for Two Bit Pixels (4 Colors/Gray Shades)*

To store two bit pixels, four pixels are grouped into one byte of display memory as shown below:

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Pixel 0 Bit 1 | Pixel 0 Bit 0 | Pixel 1 Bit 1 | Pixel 1 Bit 0 | Pixel 2 Bit 1 | Pixel 2 Bit 0 | Pixel 3 Bit 1 | Pixel 3 Bit 0 |

Figure 3-2 Pixel Storage for 2 Bits (4 colors/gray shades) in one byte of Display Memory

When these pixels are shown, Pixel 0 is seen to be left of Pixel 1, Pixel 1 is seen to be left of Pixel 2, and so on. Two bit pixels are available in both monochrome and color panels.

### *Memory Organization for Four Bit Pixels (16 Colors/Gray Shades)*

To store four bit pixels, two pixels are grouped into one byte of display memory as shown below:

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Pixel 0 Bit 3 | Pixel 0 Bit 2 | Pixel 0 Bit 1 | Pixel 0 Bit 0 | Pixel 1 Bit 3 | Pixel 1 Bit 2 | Pixel 1 Bit 1 | Pixel 1 Bit 0 |

Figure 3-3 Pixel Storage for 4 Bits (16 colors/gray shades) in one byte of Display Memory

When these pixels are shown, Pixel 0 is seen to be left of Pixel 1. For color panels, each four bit pixel represents an index into the red, green, and blue LUTs. For monochrome panels, each four bit pixel represents an index into the green LUT.

### *Memory Organization for Eight Bit Pixels (256 Colors)*

To store eight bit pixels, one pixel is stored in one byte of display memory as shown below:

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Red Bit 2 | Red Bit 1 | Red Bit 0 | Green Bit 2 | Green Bit 1 | Green Bit 0 | Blue Bit 1 | Blue Bit 0 |

Figure 3-4 Pixel Storage for 8 Bits (256 colors) in one byte of Display Memory

As shown above, the 256 color pixel is divided into three parts: three bits for red, three bits for green, and two bits for blue. The red bits represent an index into the red LUT, the green bits represent an index into the green LUT, and the blue bits represent an index into the blue LUT. Eight bit pixels are only available in color panels.

## *3.2 Look-Up Table (LUT)*

This section provides a concise description of the LUT registers, followed by a description of the color and monochrome LUTs. Next is a series of examples which show how to initialize the LUTs, create an inverted LUT, and how to select one of four banks in both the 4 gray shade and color modes.

### *LUT Registers*

| **AUX[0E] Look-Up Table Address Register**<br>I/O address = 1110b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| Green Bank Bit 1 | Green Bank Bit 0 | ID Bit / RGB Index Bit 1 | ID Bit / RGB Index Bit 0 | Palette Address Bit 3 | Palette Address Bit 2 | Palette Address Bit 1 | Palette Address Bit 0 |

The S1D13503 has three internal 16 position, 4-bit wide Look-Up Tables (also referred to as palettes). The 4-bit value programmed into each table position determines the output gray shade / color weighting of display data. These tables are bypassed in black-and-white (BW) display mode.

These three 16 position Look-Up Tables can be arranged in many different configurations to accommodate all the gray shade / color display modes.

Refer to *"Look-Up Table Description"* on page 11 for formats.

**bits 7–6 Green Bank Bits [1:0]**

In 4-level gray / color display modes (2 bits/pixel), the 16 position Green palette is arranged into four, 4 position "banks". These two bits control which bank is currently selected. These bits have no effect in 16-level gray / color display modes (4 bits/pixel).

In 256 color display modes (8-bit/pixel), the 16 position Green palette is arranged into two, 8 position "banks" for the display of "green" colors. Only bit 0 of these two bits controls which bank is currently selected.

**bits 5–4 ID Bits / RGB Index Bits [1:0]**

These bits have dual purpose;

***ID Bits:*** After "power on" or hardware reset, these bits can be read to identify the current revision of the S1D13503. These same bits are used to identify the pin compatible S1D13502F0x and would only be used in system implementations where common software is being used. As these bits are R/W they must be read before being written in order to be used as ID bits.

Table 3-2 ID-Bit Usage

| | Chip | AUX[0E] | |
|---|---|---|---|
| | | Bit 5 | Bit 4 |
| Power On or RESET | S1D13503 | 0 | 0 |
| | reserved | 0 | 1 |
| | SED1352F0B /F1B/D0B | 1 | 0 |
| | SED1352F0A /F1A/D0A | 1 | 1 |

***RGB Index Bits [1:0]:*** These bits are also used to provide access to the three internal Look-Up Tables (RGB).

Table 3-3 Look-Up Table Access

| AUX[0E] | | Look-Up Table Access |
|---|---|---|
| Bit 5 | Bit 4 | |
| 0 | 0 | Auto-increment (see Note) |
| 0 | 1 | Red palette R/W access |
| 1 | 0 | Green palette R/W access |
| 1 | 1 | Blue palette R/W access |

**Note:** When auto-increment is selected, an internal pointer will default to the Red palette on power on reset. Each read/write access to AUX[0F] will increment the counter to point to the next palette in order (RGB). Whenever the Look-Up Table Address register AUX[0E] is written, the RGB Index will reset the pointer to the Red palette. This provides a efficient method for sequential writing of RGB data.

**bits 3–0 Palette Address Bits [3:0]**

These 4 bits provide a pointer into the 16 position Look-Up Table currently selected for CPU R/W access.

**Note:** The Look-Up Table configuration (e.g. 1/2/4 banks) does not affect the R/W access from the CPU. All 16 positions can be accessed sequentially.

**AUX[0F] Look-Up Table Data Register**
I/O address = 1111b, Read/Write

| Red Bank Bit 1 | Red Bank Bit 0 | Blue Bank Bit 1 | Blue Bank Bit 0 | Palette Data Bit 3 | Palette Data Bit 2 | Palette Data Bit 1 | Palette Data Bit 0 |
|---|---|---|---|---|---|---|---|

**bits 7–6 Red Bank Bits [1:0]**

In 4-level color display modes, the 16 position Red palette is arranged into four, 4 position “banks”. These two bits control which bank is currently selected.
In 256 color display modes, the 16 position, Red palette is arranged into two, 8 position “banks” for the display of “red” colors. Only bit 0 of these two bits controls which bank is currently selected.
These bits have no effect in all gray shade or 16-color display modes.

**bits 5–4 Blue Bank Bits [1:0]**

In both the 4 and 256 color display modes, the 16 position Blue palette is arranged into four 4 position “banks” for the display of “blue” colors. These two bits control which bank is currently selected.
These bits have no effect in all gray shade display modes or 16 color display modes.

**bits 3–0 Palette Data Bits [3:0]**

These 4 bits are the gray shade / color values used for display data output. They are programmed into the 4-bit Look-Up Table (palettes) positions pointed to by Palette Address bits [3:0] and RGB Index bits [1:0] (if in color display modes).

For example; in a 16-level gray shade display mode, a data value of 0001b (4 bits/pixel) will point to Look-Up Table position one and display the 4-bit gray shade corresponding to the value programmed into that location.

## *Look-Up Table Description*

- The Look-Up Table (LUT, or palette) treats the value of a pixel as an index into an array of colors or gray shades. For example, a pixel value of zero would point to the first LUT entry; a pixel value of 7 would point to the eighth LUT entry.
- The value inside each LUT entry represents the intensity of the given color or gray shade. This value ranges between 0 and 0Fh.
- The S1D13503 Look-Up table is linear; increasing the LUT entry number results in a lighter color or gray shade. For example, a LUT entry of 0Fh into the red Look-Up entry will always result in a bright red output. An entry of 00h into a Look-Up entry will always result in the removal of this color (black if monochrome).
- Because LUT entries represent the actual colors shown on the LCD panel, pixel values indirectly select which color or gray shade to display.
- When the number of bits in a pixel is less than 4, there are several different LUT configurations based on whether the display is monochrome or color, and the number of gray shades or colors.

Table 3-4 Look-Up Table Configuration

| Display Mode | 4-Bit Wide Palette | | |
|---|---|---|---|
| | RED | GREEN | BLUE |
| Black & White | | | |
| 4-level gray | | 4 banks of 4 | |
| 16-level gray | | 1 bank of 16 | |
| 4 color | 4 banks of 4 | 4 banks of 4 | 4 banks of 4 |
| 16 color | 1 bank of 16 | 1 bank of 16 | 1 bank of 16 |
| 256 color | 2 banks of 8 | 2 banks of 8 | 4 banks of 4 |

Indicates the palette is not used for that display mode.

### Color Mode

In color mode, the S1D13503 supports three 16 position, 4 bit wide color LUTs (red, green, and blue). Depending on the selected pixel size, these LUTs will provide from 1 to 4 banks.

- **2 bits-per-pixel (4 colors)**
  In this format the pixel is an index into the red, green, and blue LUTs. Each color LUT supports 4 banks (see *"Four Colors (Two Bits/Pixel in Color Mode)"* on page 16).
- **4 bits-per-pixel (16 colors)**
  In this format the pixel is an index into the red, green, and blue LUTs. Each color LUT supports only one bank (see *"Sixteen Colors (Four Bits/Pixel in Color Mode)"* on page 19).
- **8 bits-per-pixel (256 colors)**
  In this format the pixel is divided into three parts: 3 bits for red, 3 bits for green, and 2 bits for blue. If the red, green, and blue LUTs were programmed to show a linear increase in intensity of the given color, the 8 bit pixel describes the intensity of the given set of colors. For example, a pixel value of 00h would be black, E0h would be bright red, 1Ch would be bright green, and 03h would be bright blue. Because there are 16 entries for each color LUT, the S1D13503 provides two red banks, two green banks, and four blue banks in 256 color mode (see *"256 Colors (Eight Bits/Pixel in Color Mode)"* on page 20).

### Monochrome Mode

In monochrome mode, the S1D13503 treats the green LUT as a 16 position, 4-bit wide monochrome LUT. Depending on the selected pixel size, this LUT will provide from 1 to 4 banks.

- **1 bit-per-pixel (black-and-white)**
  In this format no LUT is used. A pixel value of 0 is black, and a pixel value of 1 is white.

- **2 bits-per-pixel (4 gray shades)**
  In this format the pixel is an index into the monochrome LUT. The monochrome LUT supports 4 banks (see *"Four Gray Shades (Two Bits/Pixel in Monochrome Mode)"* on page 14).
- **4 bits-per-pixel (16 gray shades)**
  In this format the pixel is an index into the monochrome LUT. The monochrome LUT supports only one bank (see *"Sixteen Gray Shades (Four Bits/Pixel in Monochrome Mode)"* on page 18).

### Example 3

### Initialize the Look-Up Table for 256 colors (Bank 0 Only)

Table 3-5 shows the color LUTs with intensities starting from black (index 0) and finishing in maximum color intensity (at the largest index available for the color in bank 0). For example, the red LUT would have a maximum intensity at index 07h, the green LUT would have a maximum intensity at index 07h, and the blue LUT would have a maximum intensity at index 03h. A normal display would use bank 0 for the red, green, and blue LUTs.

1. Write LUT index to Look-Up Table Address Register AUX[0Eh], set to automatic increment mode.
2. Write red LUT entry value to Look-Up Table Data Register AUX[0Fh].
3. Write green LUT entry value to Look-Up Table Data Register AUX[0Fh].
4. Write blue LUT entry value to Look-Up Table Data Register AUX[0Fh].
5. Repeat steps 1-4 until all 16 LUT entries have been written.

Table 3-5 S1D13503 Color Look-Up Table for 256 Color Mode

| Index (hex) | Red LUT (hex) | Green LUT (hex) | Blue LUT (hex) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 2 | 2 | 5 |
| 2 | 4 | 4 | A |
| 3 | 6 | 6 | F |
| 4 | 9 | 9 | X |
| 5 | B | B | X |
| 6 | D | D | X |
| 7 | F | F | X |

Where X is Don't Care

## Example 4

### Initialize the Look-Up Table for 16 gray shades

The following describes how to initialize the Look-Up table for 16 gray shades. Table 3-6 shows a LUT with gray shades starting from black (index 0) and finishing in white (index 15, or 0Fh).

1. Write LUT index to Look-Up Table Address Register AUX[0Eh].
2. Write LUT entry value to Look-Up Table Data Register AUX[0Fh].
3. Repeat steps 1 and 2 until all 16 LUT entries have been written.

Table 3-6 S1D13503 Black-To-White Look-Up Table

| Index (hex) | Look-Up Table (hex) | Index (hex) | Look-Up Table (hex) |
|---|---|---|---|
| 0 | 0 | 8 | 8 |
| 1 | 1 | 9 | 9 |
| 2 | 2 | A | A |
| 3 | 3 | B | B |
| 4 | 4 | C | C |
| 5 | 5 | D | D |
| 6 | 6 | E | E |
| 7 | 7 | F | F |

## Example 5

### Initialize an inverted Look-Up Table

This example shows how to invert an image by changing only the LUT. Inverting means that pixels formally shown as light gray shades are now shown as dark gray shades, and vise versa. It does not matter whether the S1D13503 is in 4 gray shade or 16 gray shade mode.

1. Read LUT entry.
   Write LUT index to Look-Up Table Address Register AUX[0Eh]
   Read “Old LUT Entry” from Look-Up Table Data Register AUX[0Fh]
2. Calculate “New LUT Entry” according to the following formula:

$$\text{New LUT Entry} = 15 - \text{Old LUT Entry}$$

3. Write LUT entry back.
   Write LUT index to Look-Up Table Address Register AUX[0Eh]
   Write “New LUT Entry” to Look-Up Table Data Register AUX[0Fh]
4. Repeat steps 1 to 3 until all 16 LUT entries have been changed.

If Table 3-6 was previously programmed into the S1D13503, the new inverted LUT would be the following:

Table 3-7 S1D13503 Inverted Look-Up Table (White-To-Black)

| Index (hex) | Look-Up Table (hex) | Index (hex) | Look-Up Table (hex) |
|---|---|---|---|
| 0 | F | 8 | 7 |
| 1 | E | 9 | 6 |
| 2 | D | A | 5 |
| 3 | C | B | 4 |
| 4 | B | C | 3 |
| 5 | A | D | 2 |
| 6 | 9 | E | 1 |
| 7 | 8 | F | 0 |

## *Black-and-White (One Bit/Pixel)*

When the S1D13503 is configured for one bit pixels, the monochrome (green) LUT is not used. Instead, a pixel value of 0 represents black and a pixel value of 1 represents white.

**Note:** One bit/pixel is only available in monochrome mode.

## *Four Gray Shades (Two Bits/Pixel in Monochrome Mode)*

When the S1D13503 is configured for two bit pixels in monochrome mode, each pixel can index one of four monochrome LUT entries. Note that in monochrome mode, the S1D13503 uses the green LUT as the monochrome LUT. The 16 LUT entries are divided into four separate Look-Up tables or *banks*, each having four entries (see Figure 3-5). The following examples show how to program and select these banks.

**Example 6**

**In 4 gray shade mode, program bank 2 LUT entries and select for use.**

1. Determine location of bank 2 in LUT.
   The first four entries in the 16 entry LUT represent the first bank (bank 0). The following four entries in the LUT represent the second bank (bank 1), etc. Consequently bank 2 starts at LUT index 8 as shown below:

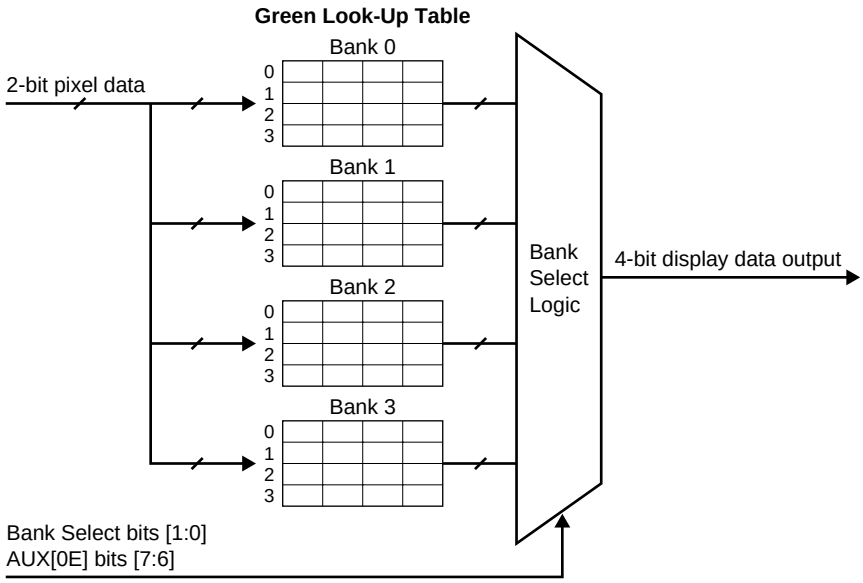
   $$\text{Start of bank index} = \text{bank number} \times 4$$
   $$\text{Start of bank 2} = 2 \times 4 = 8$$

   Monochrome (green) Bank 2 is shown in Figure 3-5.
2. Write LUT index to Look-Up Table Address Register AUX[0Eh].
   For bank 2, the index will one of the following values: 08h, 09h, 0Ah, or 0Bh
3. Write LUT entry value to Look-Up Table Data Register AUX[0Fh].
   For a linear LUT, use the Look-Up table entries in Table 3-8.
4. Repeat steps 2 and 3 until all 4 LUT entries have been written.
5. To display data using Bank 2, write 10b to AUX[0E] bits 7, 6.

Table 3-8 S1D13503 Black-To-White Look-Up Table for 4 Gray Shades

| Index (hex) | Look-Up Table (hex) |
|---|---|
| 8 | 0 |
| 9 | 5 |
| A | A |
| B | F |

## 4-Level Gray Shade Mode




**Note:** The above depiction is intended to show the display data output path only. The CPU R/W access to the individual Look-Up Tables is not affected by the various 'banking' configurations.

Figure 3-5 4-Level Gray-Shade Mode Look-Up Table Architecture

## *Four Colors (Two Bits/Pixel in Color Mode)*

When the S1D13503 is configured for two bit pixels in color mode, each pixel can index one of four color LUT entries. The 16 LUT entries are divided into four separate Look-Up tables or *banks*, each having four entries (see Figure 3-6). The following examples show how to program and select these banks.

### Example 7

**In 4 color mode, program red bank 3 LUT entries and select for use.**

1. Determine location of bank 3 in the red LUT.
   The first four entries in the 16 entry LUT represent the first bank (bank 0). The following four entries in the LUT represent the second bank (bank 1), etc. Consequently bank 3 starts at LUT index 0Ch as shown below:

   $$\text{Start of bank index} = \text{Bank number} \times 4$$
   $$\text{Start of bank 3} = 3 \times 4 = 12 = 0\text{Ch}$$

   Red Bank 3 is shown in Figure 3-6.
2. Write LUT index and Red LUT selection to Look-Up Table Address Register AUX[0Eh].
   AUX[0Eh] = LUT index ‘OR’ 0001 0000b
   For bank 3, the index will one of the following values: 0Ch, 0Dh, 0Eh, or 0Fh, so the value written to AUX[0Eh] will be one of the following: 1Ch, 1Dh, 1Eh, or 1Fh. This selects the Red LUT only, indexes C, D, E and F.
3. Write LUT entry value to Look-Up Table Data Register AUX[0Fh].
   For a linear LUT, use the Look-Up table entries in Table 3-9.
4. Repeat steps 2 and 3 until all 4 LUT entries have been written.
5. To display data using Red Bank 3 write 11b to AUX[0Fh] bits 7,6:
   AUX[0Fh] = original AUX[0Fh] ‘OR’ 1100 0000b

Table 3-9 S1D13503 Low to High Intensity Color Look-Up Table for 4 Colors

| Index (hex) | Look-Up Table (hex) |
|---|---|
| C | 0 |
| D | 5 |
| E | A |
| F | F |

## 4-Level Color Mode

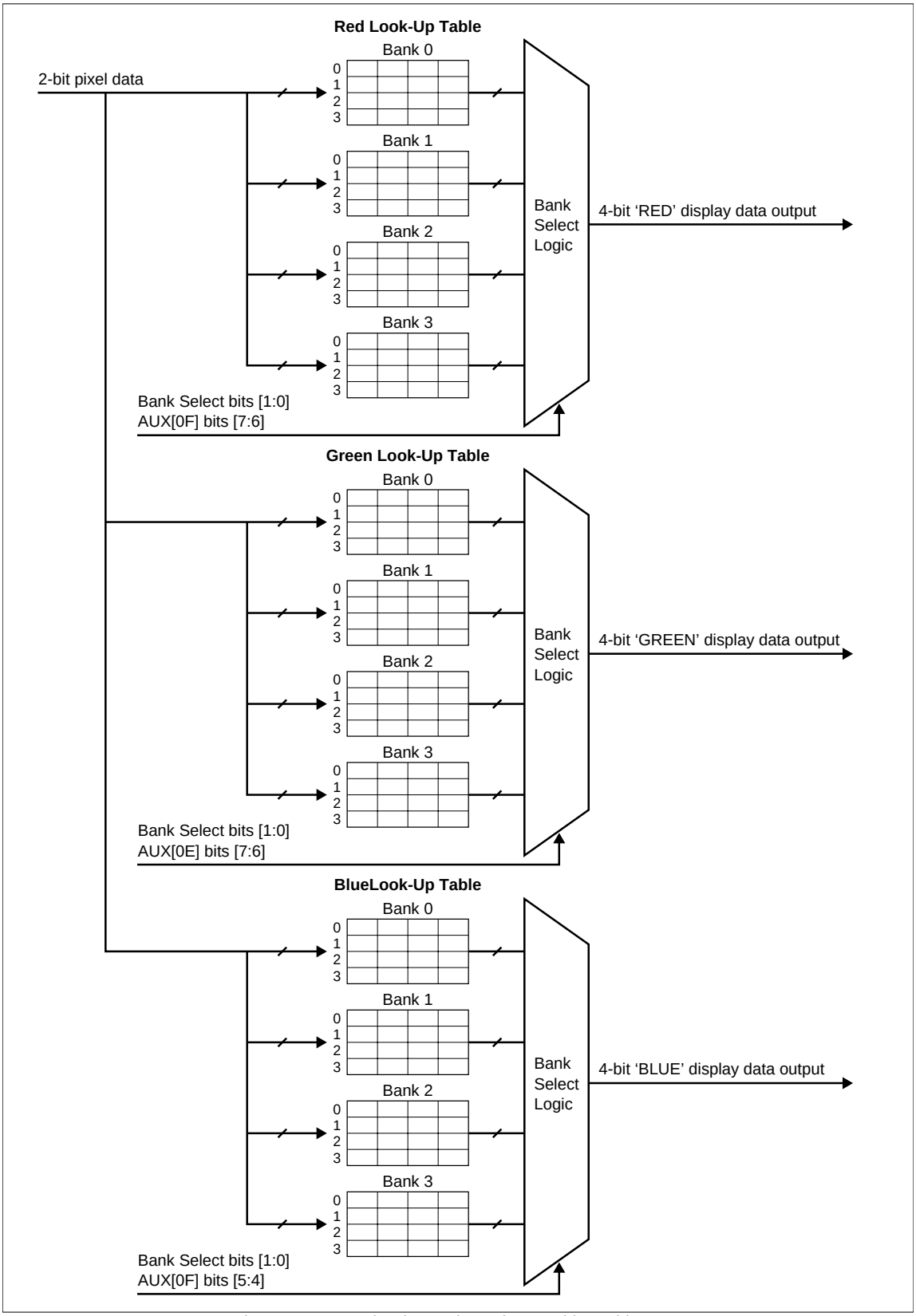



Figure 3-6 4-Level Color Mode Look-Up Table Architecture

## *Sixteen Gray Shades (Four Bits/Pixel in Monochrome Mode)*

When the S1D13503 has 4-bit monochrome pixels, each pixel can index into one of 16 LUT entries. The LUT bank bits are ignored in this mode.

### 16-Level Gray Shade Mode

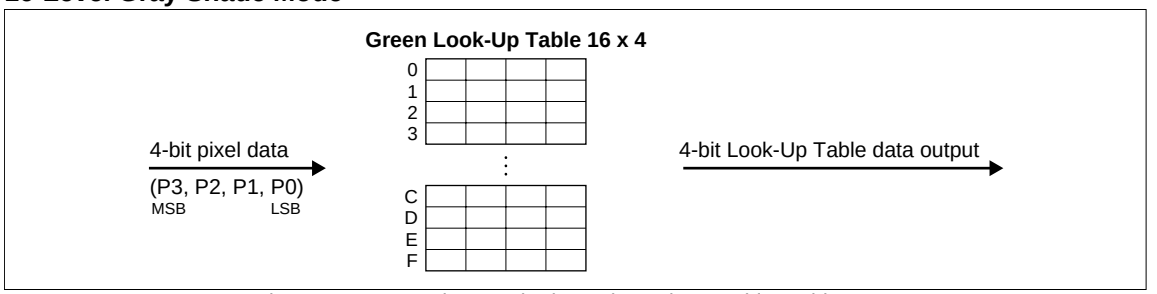



Figure 3-7 16-Level Gray-Shade Mode Look-Up Table Architecture

## *Sixteen Colors (Four Bits/Pixel in Color Mode)*

When the S1D13503 has 4-bit color pixels, each pixel can index into each of the three color LUTs. The LUT bank bits are ignored in this mode.

### 16-Level Color Mode

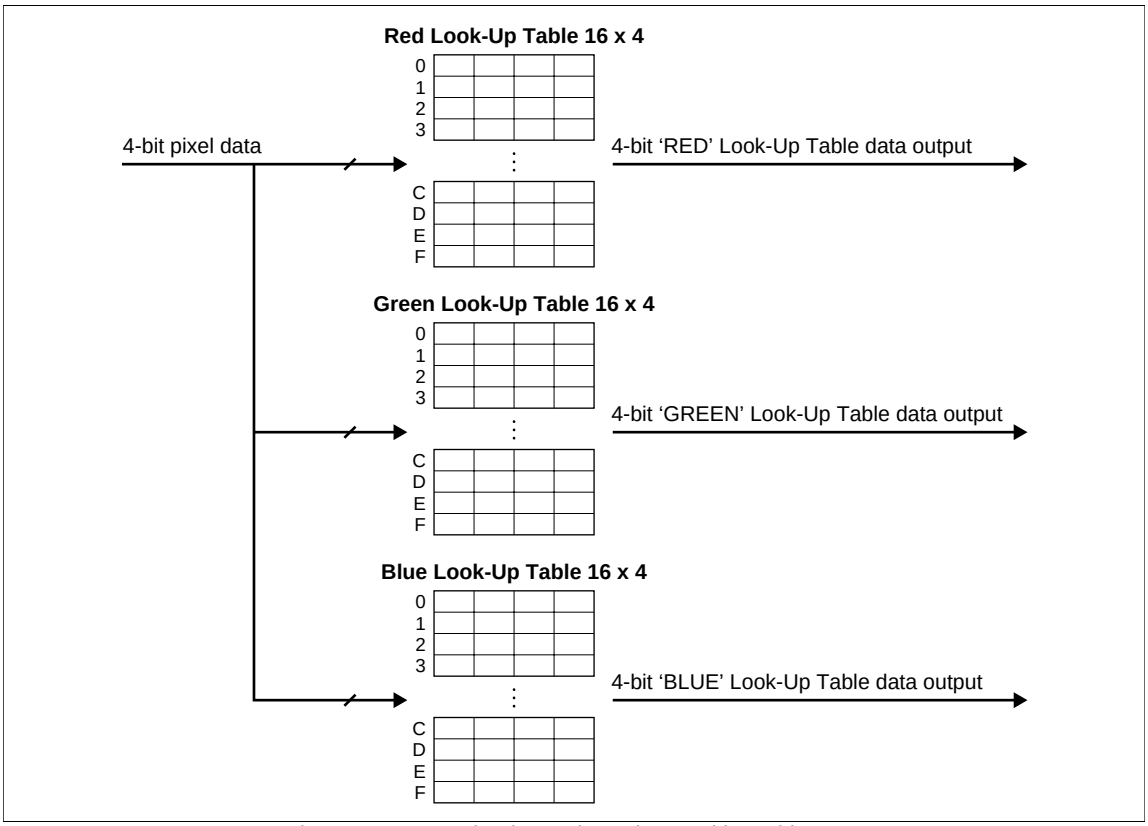



Figure 3-8 16-Level Color Mode Look-Up Table Architecture

Table 3-10 Simulation of First 16 Entries of Standard VGA Palette

| Address | Red | Green | Blue |
|---|---|---|---|
| 00 | 00 | 00 | 00 |
| 01 | 00 | 00 | 0A |
| 02 | 00 | 0A | 00 |
| 03 | 00 | 0A | 0A |
| 04 | 0A | 00 | 00 |
| 05 | 0A | 00 | 0A |
| 06 | 0A | 0A | 00 |
| 07 | 0A | 0A | 0A |

| Address | Red | Green | Blue |
|---|---|---|---|
| 08 | 00 | 00 | 00 |
| 09 | 00 | 00 | 0F |
| 0A | 00 | 0F | 00 |
| 0B | 00 | 0F | 0F |
| 0C | 0F | 00 | 00 |
| 0D | 0F | 00 | 0F |
| 0E | 0F | 0F | 00 |
| 0F | 0F | 0F | 0F |

## *256 Colors (Eight Bits/Pixel in Color Mode)*

When the S1D13503 has 8-bit color pixels, bits 7–5 represent the red LUT index, bits 4–2 represent the green LUT index, and bits 1-0 represent the blue LUT index (see Figure 3-9, “256-Level Color Mode Look-Up Table Architecture,” on page 21). It is recommended that the three LUTs are programmed according to Table 3-5, “S1D13503 Color Look-Up Table for 256 Color Mode,” on page 12, and only bank 0 were used for each of the three colors. This method results in each color index inside the pixel to represent its respective color intensity (see Table 3-11 below).

Table 3-11 Examples of 256 Pixel Colors Using Linear LUT

| Pixel Value (binary) | Color |
|---|---|
| 000 000 00 | black |
| 000 000 10 | dark blue |
| 000 100 00 | dark green |
| 000 100 10 | dark cyan |
| 100 000 00 | dark red |
| 100 000 10 | dark magenta |
| 100 100 00 | dark yellow |
| 100 100 10 | gray |

| Pixel Value (binary) | Color |
|---|---|
| 000 000 00 | black |
| 000 000 11 | bright blue |
| 000 111 00 | bright green |
| 000 111 11 | bright cyan |
| 111 000 00 | bright red |
| 111 000 11 | bright magenta |
| 111 111 00 | bright yellow |
| 111 111 11 | white |

## 256-Level Color Mode

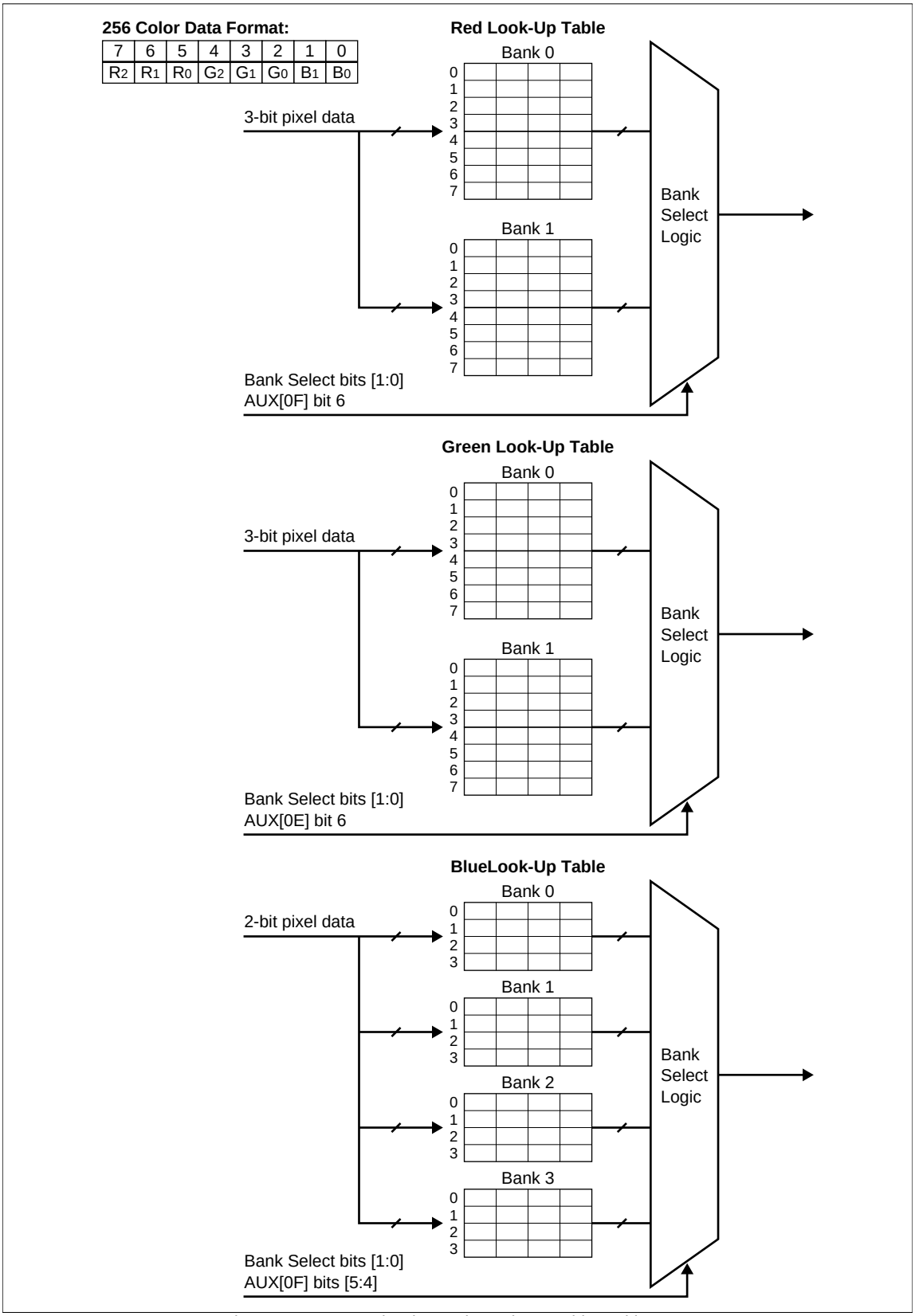



Figure 3-9 256-Level Color Mode Look-Up Table Architecture

# 4 DISPLAY MEMORY MODELS

This section includes a concise description of the Display Start Address Registers, followed by a description of display memory. Afterwards examples are provided, illustrating how to calculate the display memory model for a given display resolution and color/gray level mode. Once this model is calculated, examples on programming the Display Start Address Registers are provided.

## 4.1 Registers

Register bits discussed in this section are highlighted.

**AUX[01] Mode Register 0**
I/O address = 0001b, Read/Write

| DISP | Panel | Mask XSCL | LCDE | Gray Shade / Color | LCD Data Width Bit 0 | Memory Interface | RAMS |
|---|---|---|---|---|---|---|---|

**bit 1 Memory Interface**

This bit selects between the 8-bit or 16-bit memory interface. When this bit = 0, the 16-bit memory interface is selected. When this bit = 1, the 8-bit memory interface is selected. If 16-bit bus interface (VD0 = 1 on RESET) or 256 color mode (AUX[03] bits 2–1 = 11) is selected, the Memory Interface bit is forced to 0 internally (16-bit). This bit goes low on RESET.

**AUX[06] Screen 1 Display Start Address Register (LSB)**
I/O address = 0110b, Read/Write

| Screen 1 Display Start Addr Bit 7 | Screen 1 Display Start Addr Bit 6 | Screen 1 Display Start Addr Bit 5 | Screen 1 Display Start Addr Bit 4 | Screen 1 Display Start Addr Bit 3 | Screen 1 Display Start Addr Bit 2 | Screen 1 Display Start Addr Bit 1 | Screen 1 Display Start Addr Bit 0 |
|---|---|---|---|---|---|---|---|

**AUX[07] Screen 1 Display Start Address Register (MSB)**
I/O address = 0111b, Read/Write

| Screen 1 Display Start Addr Bit 15 | Screen 1 Display Start Addr Bit 14 | Screen 1 Display Start Addr Bit 13 | Screen 1 Display Start Addr Bit 12 | Screen 1 Display Start Addr Bit 11 | Screen 1 Display Start Addr Bit 10 | Screen 1 Display Start Addr Bit 9 | Screen 1 Display Start Addr Bit 8 |
|---|---|---|---|---|---|---|---|

**AUX[06] bits 7–0, AUX[07] bits 7–0**
**Screen 1 Display Start Address Bits [15:0]**

These 16 bits determine the Screen 1 Display Start Address. In an 8-bit memory configuration these bits set the 16-bit start address (i.e., byte access). In a 16-bit memory configuration these are the 16 most significant bits of a 17-bit start address (i.e., word access).

**Note:** The absolute address into display memory is determined by the Memory Mapping Address which is set by the reset state of VD13–VD15.

The Screen 1 Display Start Address is the memory address corresponding to the first displayed pixel (top left corner). In a dual panel configuration, screen 1 refers to the upper half of the display. While in a single panel configuration, screen 1 refers to the first screen of the Split Screen Display feature where two different images (screen 1 and screen 2) can be displayed at the same time on one display.

**AUX[08] Screen 2 Display Start Address Register (LSB)**
I/O address = 1000b, Read/Write

| Screen 2 Display Start Addr Bit 7 | Screen 2 Display Start Addr Bit 6 | Screen 2 Display Start Addr Bit 5 | Screen 2 Display Start Addr Bit 4 | Screen 2 Display Start Addr Bit 3 | Screen 2 Display Start Addr Bit 2 | Screen 2 Display Start Addr Bit 1 | Screen 2 Display Start Addr Bit 0 |
|---|---|---|---|---|---|---|---|

**AUX[09] Screen 2 Display Start Address Register (MSB)**
I/O address = 1001b, Read/Write

| Screen 2 Display Start Addr Bit 15 | Screen 2 Display Start Addr Bit 14 | Screen 2 Display Start Addr Bit 13 | Screen 2 Display Start Addr Bit 12 | Screen 2 Display Start Addr Bit 11 | Screen 2 Display Start Addr Bit 10 | Screen 2 Display Start Addr Bit 9 | Screen 2 Display Start Addr Bit 8 |
|---|---|---|---|---|---|---|---|

**AUX[08] bits 7–0, AUX[09] bits 7–0**

**Screen 2 Display Start Address Bits [15:0]**

These 16 bits determine the Screen 2 Display Start Address. In an 8-bit memory configuration these bits set the 16-bit start address (i.e., byte access). In a 16-bit memory configuration these are the 16 most significant bits of a 17-bit start address (i.e., word access).

In a dual panel configuration, screen 2 refers to the lower half of the display. The Screen 2 Display Start Address is the memory address corresponding to first displayed pixel in the first line of the lower half of the display and is calculated with the following formula.

$$\text{Screen 2 display start address} = \frac{(\text{Image horizontal resolution}) \times (\text{Image vertical resolution}) \times (\text{Bytes per pixel})}{2 \times \left(\frac{\text{Memory interface width}}{8}\right)} + \text{Screen 1 display start address}$$

In a single panel configuration, screen 2 refers to the second screen of the Split Screen Display Feature where two different images (screen 1 and screen 2) can be displayed at the same time on one display. The Screen 2 Display Start Address is the memory address corresponding to the first pixel of the second image stored in display memory. To display screen 2 refer to “AUX[0A] Screen 1 Display Line Count Register (LSB)” on page 31 and “AUX[0B] Screen 1 Display Line Count Register (MSB)” on page 31.

## 4.2 Description

When displaying an image, the S1D13503 must read pixel data from display memory. This memory is organized to match the display resolution of the given LCD panel. To organize display memory, the following registers must be programmed:

1. Screen 1 Display Start Address Registers
2. Screen 2 Display Start Address Registers
3. Address Pitch Adjustment Register

For the first example, the Address Pitch Adjustment Register is programmed to zero. This means that no virtual display is available; for information on virtual displays see Section 5.1, *"Virtual Displays"* on page 27.

### *S5U13503P00C Evaluation Board Display Memory*

There are several issues to consider when programming the Screen Display Start Address Registers for the S5U13503P00C evaluation board:

- The S5U13503P00C is always set for 128K of display memory. This memory exists as two 64K banks at addresses D000:0000h to D000:FFFFh. To access bank 0, read from the base port address + 2. To access bank 1, write to base port address + 2. The values read from or written to base port address + 2 are not important. The start of bank 0 represents the top left corner of display memory.
- For the S5U13503P00C, the Screen Display Start Address Registers are always in reference to the display memory address D000:0000h, bank 0. Writing 0 to a Display Start Address Register will always refer to D000:0000h, bank 0.
- Although the S1D13503 can set the Memory Interface to 8 or 16 bits, the S5U13503P00C evaluation board must be set for 16 bits in order to access 128K of display memory. As a result, the Display Start Address Registers are word pointers, not byte pointers. To illustrate how to use a word pointer, refer to Example 8. In general, any system which uses more than 64K of display memory must ***always*** have the Memory Interface set to 16 bits.

**Example 8**

**For the S5U13503P00C, calculate the required start address register value which refers to location D000:0000h, bank 1.**

Location D000:0000h bank 1 refers to the start of the second 64K bank of display memory. Consequently the start address is 10000h bytes (64K), or 8000h words.
START ADDRESS[LSB] = 00h
START ADDRESS[MSB] = 80h

## *Display Start Address Registers*

This section illustrates how to properly calculate the values for the Screen Start Address Registers for a given LCD panel resolution. However, this section is limited to single panel displays; refer to *"Dual Panel LCD"* on page 33 to program the Screen Start Address Registers for a dual panel display.

In the following example, the Display Start Address Registers are programmed for a 16 color 320 × 240 single panel LCD display. The technique shown, however, can also be used to calculate the memory map of other resolutions. In addition, reference is made to the S5U13503P00C evaluation board; other hardware implementations of the S1D13503 may assign different display and port addresses from those of the S5U13503P00C. Refer to the *S5U13503P00C Evaluation Board User's Manual* for more information on these hardware issues.

**Example 9**

**Program the Display Start Address Registers for a single LCD panel; the display is attached to the S5U13503P00C evaluation board.**

Normally images are loaded at the start of display memory (D000:0000h, bank 0), so the display start address registers must be set to 0000h words.

AUX[06h] = 00h

AUX[07h] = 00h

**Example 10**

**Program the Display Start Address Registers for a dual panel LCD.**

Refer to *"Displaying a Single Image on a Dual Panel"* on page 36.

**Example 11**

**Determine if the S1D13503 implementation can support a 640 x 480 LCD with 4 colors.**

1. Calculate the number of bytes per scan line:

$$\frac{\text{Pixels per scan line}}{\text{Pixels per byte}} = \frac{640}{4} = 160 \text{ bytes per scan line}$$

2. Calculate the total number of bytes required for display memory:

$$(160 \text{ bytes per scan line})(480 \text{ scan lines}) = 76800 \text{ bytes}$$

3. Compare the required number of bytes with the amount of memory available to the S1D13503.

- The S1D13503 has 128K available, so there is 131,072 bytes available. Since this number is greater than the 76,800 bytes required for 640 × 480 with 4 colors, the S1D13503 implementation can support a 640 × 480 LCD with 4 colors.

**Note:** The memory required for 4 colors at 640 × 480 is the same as the memory required for 4 gray shades at 640 × 480. Consequently the S1D13503 implementation can also support a 640 × 480 LCD with 4 gray shades.

## 4.3 Common Display Memory Requirements for LCD Panel Sizes

The following is a list of memory requirements and memory maps for common LCD resolutions. Note that the memory required for 640 × 480 with 4 or 8 bits/pixel exceeds 128K and is therefore not supported on the S1D13503.

Table 4-1 Memory Size Requirements

| Display Resolution | Pixel Storage | | Memory Requirements | |
|---|---|---|---|---|
| | Bits/Pixel | Colors/ Gray Shades | Bytes | Hex |
| 320 × 240 | 1 | 2 | 9,600 | 0000 2580 |
| | 2 | 4 | 19,200 | 0000 4B00 |
| | 4 | 16 | 38,400 | 0000 9600 |
| | 8 | 256 | 76,800 | 0001 2C00 |
| 480 × 240 | 1 | 2 | 14,400 | 0000 3840 |
| | 2 | 4 | 28,800 | 0000 7080 |
| | 4 | 16 | 57,600 | 0000 E100 |
| | 8 | 256 | 115,200 | 0001 C200 |
| 640 × 200 | 1 | 2 | 16,000 | 0000 3E80 |
| | 2 | 4 | 32,000 | 0000 7D00 |
| | 4 | 16 | 64,000 | 0000 FA00 |
| | 8 | 256 | 128,000 | 0001 F400 |
| 640 × 480 | 1 | 2 | 38,400 | 0000 9600 |
| | 2 | 4 | 76,800 | 0001 2C00 |
| | 4 | 16 | N/A | N/A |
| | 8 | 256 | N/A | N/A |

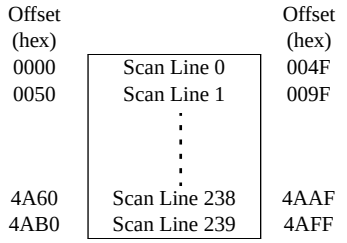



Figure 4-1 Memory Map Example for 320 × 240 LCD Panel with 4 Colors/Gray Shades

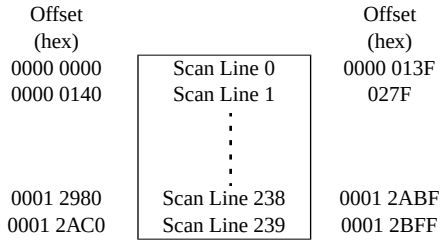



Figure 4-2 Memory Map Example for 320 × 240 LCD Panel with 256 Colors

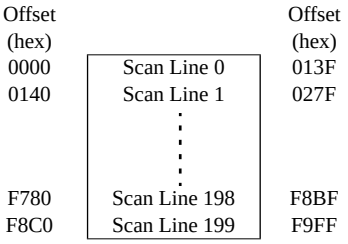



Figure 4-3 Memory Map Example for 640 × 200 LCD Panel with 16 Colors/Gray Shades

# 5 ADVANCED TECHNIQUES

This section presents information on the following:
- virtual displays
- bitmaps and text displays
- reading and writing to the S1D13503 registers
- split screen displays
- panning and scrolling
- power saving

## 5.1 Virtual Displays

This section presents a detailed description of the Address Pitch Adjustment Register, followed by a description of a virtual display. Afterwards an example is given, showing how to create a virtual display.

### Registers

Register bits discussed in this section are highlighted.

**AUX[0D] Address Pitch Adjustment Register**
I/O address = 1101b, Read/Write

| Addr Pitch Adjustment Bit 7 | Addr Pitch Adjustment Bit 6 | Addr Pitch Adjustment Bit 5 | Addr Pitch Adjustment Bit 4 | Addr Pitch Adjustment Bit 3 | Addr Pitch Adjustment Bit 2 | Addr Pitch Adjustment Bit 1 | Addr Pitch Adjustment Bit 0 |
|---|---|---|---|---|---|---|---|

**bits 7–0 Address Pitch Adjustment Bits [7:0]**

This register controls the virtual display by setting the numerical difference between the last address of a display line, and the first address in the following line.

If the Address Pitch Adjustment is not equal to zero, then a virtual screen is formed. The size of the virtual screen is only limited by the available display memory. The actual display output is a window that is part of the whole image stored in the display memory. For example, with 128K of display memory, a 640 × 400 16-gray image can be stored. If the output display size is 320 × 240, then the whole image can be seen by changing display starting addresses through AUX[06] and [07], and AUX[08] and [09]. Note that a virtual screen can be produced on either a single or dual panel.

In 8-bit memory interface, if the Address Pitch Adjustment is not equal to zero, a virtual screen with a line length of (Line Byte Count + AUX[0D]) bytes is created, with the display reflecting the contents of a window (Line Byte Count + 1) bytes wide. The position of the window on the virtual screen is determined by AUX[06] and [07], and AUX[08] and [09].

In 16-bit memory interface, if the Address Pitch Adjustment is not equal to zero, then a virtual screen with a line length of 2 × (Line Byte Count + AUX[0D]) bytes is created, with the display reflecting the contents of a window 2 × (Line Byte Count + 1) bytes wide. The position of the window on the virtual screen is determined by AUX[06] and [07], and AUX[08] and [09].

## *Description*

The S1D13503 can be programmed to wrap memory offsets in such a way that the physical display behaves as a viewport into a much larger "virtual" memory space. This viewport can be panned and/or scrolled to display this larger memory space.

Referring to the figure below, a virtual image of 640 × 480 can be viewed by navigating the 320 × 240 viewport around the image by panning and scrolling.

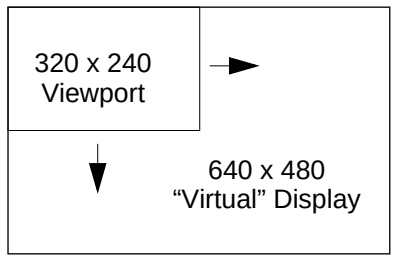



Figure 5-1 Moving a Viewport Inside a Virtual Display

To create a virtual display, the Address Pitch Adjustment Register must be programmed to indicate the horizontal size of the larger, "virtual" image stored in display memory. The Address Pitch Adjustment Register tells the S1D13503 how many bytes or words of display memory are part of the nonvisible region of display memory (see Example 12).

**Example 12**

**Program the Address Pitch Adjustment Register to support a 16 color 640 x 480 virtual display on a 320 x 240 LCD panel; the Memory Interface is 16 bits.**

1. Initialize the S1D13503 registers for a 320 × 240 panel.
2. Determine whether the Address Pitch Adjustment Register refers to bytes or words.
   Since the Memory Interface is set to 16 bits, the Address Pitch Adjustment Register refers to words.
3. Determine the number of pixels per unit referred to by the Address Pitch Adjustment Register.
   The Address Pitch Adjustment Register refers to units of words, so find the number of pixels per word.

   16 colors → 4 bits per pixel

   4 bits per pixel → 2 pixels per byte

   Pixels per word = (Pixels per byte) × 2 = 2 × 2 = 4 pixels per word

4. Calculate the number of pixels on a horizontal scan line not visible.

   (Virtual display width in pixels) – (Panel width in pixels) = 640 – 320 = 320 hidden pixels

   Consequently on a screen update the S1D13503 will show the first 320 of 640 pixels, and then ignore the remaining 320 pixels in order to reach the next scan line.
5. Program the Address Pitch Adjustment Register

$$\frac{\text{Number of hidden horizontal pixels}}{\text{Pixels per word}} = \frac{320}{4} = 80 \text{ words} = 50\text{h words}$$

   Therefore AUX[0Dh] = 50h
6. To view the rest of the image refer to Section 5.5, *"Panning and Scrolling"* on page 38, keeping in mind that the horizontal width is 640 pixels, not 320.

## 5.2 Bitmaps and Text Displays

For the scope of this guide, a bitmap is a data structure which represents the image shown on the LCD. The bitmap includes the dimensions of the image, and the color or gray shade palette used to program the lookup table. Text is shown by creating a font, which in this example is a series of bitmaps, one bitmap per alphanumeric character.

**Example 13**

**Display the word "TEXT" on a 256 color 320 x 240 LCD panel; the Memory Interface is 16 bits.**

1. Define the font for the letters 'T', 'E', and 'X'.
   Each character is 8 × 8 pixels, with at least one horizontal and vertical side left blank for spacing.

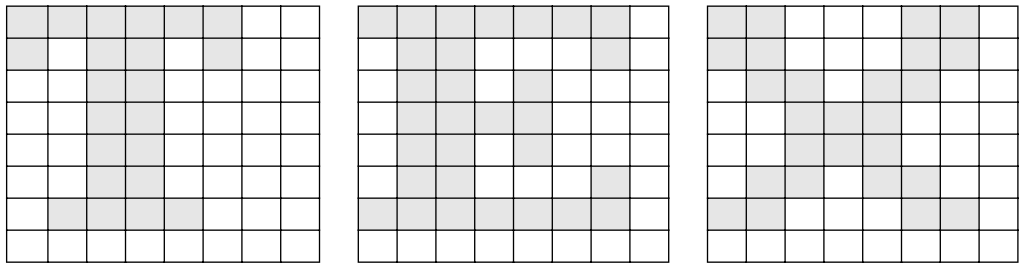


Figure 5-2 Font for the Message "TEXT"

2. Program the lookup table.
   See Example 3, "Initialize the Look-Up Table for 256 colors (Bank 0 Only)," on page 12.
3. Calculate the display memory map.
   See Figure 4-2, "Memory Map Example for 320 × 240 LCD Panel with 256 Colors," on page 26.
4. Write font to display memory.
   In a general purpose program the entire bitmapped font would be placed in an array. As characters are to be displayed, the program would choose the appropriate bitmap, select the proper position on the screen, and write to display memory. For this example assume that the program has already selected the proper bitmaps and the correct positions in display memory (there is a detailed programming example later in this guide; see Section 7.3, *"Advanced Functions"* on page 51).

   Each highlighted pixel in the text bitmap will be shown at maximum intensity, which is pixel value 0FFh. The text, for simplicity, will be shown in the upper left corner of the screen. When the program has completed writing the pixels for the word "TEXT", the display memory will have the data shown in Figure 5-3. In this figure the bytes are grouped within vertical lines.

| Offset (hex) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Offset (hex) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | F | F | F | F | F | F | 0 | 0 | F | F | F | F | F | F | F | 0 | F | F | 0 | 0 | 0 | F | F | 0 | F | F | F | F | F | F | 0 | 0 | 001F |
| | F | F | F | F | F | F | 0 | 0 | F | F | F | F | F | F | F | 0 | F | F | 0 | 0 | 0 | F | F | 0 | F | F | F | F | F | F | 0 | 0 | |
| 0140 | F | 0 | F | F | 0 | F | 0 | 0 | 0 | F | F | 0 | 0 | 0 | F | 0 | F | F | 0 | 0 | 0 | F | F | 0 | F | 0 | F | F | 0 | F | 0 | 0 | 015F |
| | F | 0 | F | F | 0 | F | 0 | 0 | 0 | F | F | 0 | 0 | 0 | F | 0 | F | F | 0 | 0 | 0 | F | F | 0 | F | 0 | F | F | 0 | F | 0 | 0 | |
| 0280 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 0 | F | F | 0 | F | 0 | 0 | 0 | 0 | F | F | 0 | F | F | 0 | 0 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 029F |
| | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 0 | F | F | 0 | F | 0 | 0 | 0 | 0 | F | F | 0 | F | F | 0 | 0 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | |
| 03C0 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 0 | F | F | F | F | 0 | 0 | 0 | 0 | 0 | F | F | F | 0 | 0 | 0 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 03DF |
| | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 0 | F | F | F | F | 0 | 0 | 0 | 0 | 0 | F | F | F | 0 | 0 | 0 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | |
| 0500 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 0 | F | F | 0 | F | 0 | 0 | 0 | 0 | 0 | F | F | F | 0 | 0 | 0 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 051F |
| | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 0 | F | F | 0 | F | 0 | 0 | 0 | 0 | 0 | F | F | F | 0 | 0 | 0 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | |
| 0640 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 0 | F | F | 0 | 0 | 0 | F | 0 | 0 | F | F | 0 | F | F | 0 | 0 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 065F |
| | 0 | 0 | F | F | 0 | 0 | 0 | 0 | 0 | F | F | 0 | 0 | 0 | F | 0 | 0 | F | F | 0 | F | F | 0 | 0 | 0 | 0 | F | F | 0 | 0 | 0 | 0 | |
| 0780 | 0 | F | F | F | F | 0 | 0 | 0 | F | F | F | F | F | F | F | 0 | F | F | 0 | 0 | 0 | F | F | 0 | 0 | 0 | F | F | F | F | 0 | 0 | 079F |
| | 0 | F | F | F | F | 0 | 0 | 0 | F | F | F | F | F | F | F | 0 | F | F | 0 | 0 | 0 | F | F | 0 | 0 | 0 | F | F | F | F | 0 | 0 | |
| 08C0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 08DF |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Figure 5-3 Display Memory Contents for Message "TEXT" in 256 Color Mode

## 5.3 Mapping of Registers

The S1D13503 has an internal set of 16-/8-bit read/write registers which configure it for various modes of operation. The registers can be accessed in two ways; Indexed Addressing and Direct Addressing.

**Note:** Refer to the *S1D13503 Hardware Functional Specification* for more information on the S1D13503 registers.

### Indexed Mapping

This method requires only two sequential I/O address locations starting from the base I/O address. The base I/O address is determined by the power-on state of the SRAM data lines VD[4 through 12]. See "*Summary of Configuration Options*" in the *S1D13503 Hardware Functional Specification*.

The S5U13503P00C Evaluation Board uses three sequential I/O addresses which are defined as Index Address, Index Data, and Memory Banking. To access registers using this method, an Index Address must be written to the first I/O address location allowing data to be written/read to/from the second I/O address.

The Memory Banking port is specific to the S5U13503P00C implementation and is used to select one of two 64K display memory banks; a read from this port selects bank 0, and a write to this port selects bank 1. Note that the values read from or written to the Memory Banking port are not important.

**Example 14**

**Write 12h to register 08h on the S5U13503P00C evaluation board; the base port address is 310h, and indexed port mapping is used.**

1. Write 08h to the index register
   The index register is at base port address + 0 = 310h.

```
MOV DX,310h
MOV AL,08h
OUT DX,AL
```

2. Write 12h to the data register
   The data register is at base port address + 1 = 311h.

```
MOV DX,311h
MOV AL,12h
OUT DX,AL
```

### Direct Mapping

This method of addressing requires 16 sequential I/O addresses starting from the base I/O address. The base I/O address is determined by the power-on state of the SRAM data lines VD[7 through 12]. See "*Summary of Configuration Options*" in the *S1D13503 Hardware Functional Specification*.

To access the internal 16 registers of the S1D13503, simply perform I/O read/write functions to the absolute address as defined in the previous paragraph.

There is no memory banking available in direct addressing mode.

**Example 15**

**Write 12h to register 08h on the S5U13503P00C evaluation board; the base port address is 310h, and direct port mapping is used.**

1. Calculate the port address for register 08h.

```
Port address = 310h + 8h = 318h
```

2. Write the value 12h to port address 318h.

```
MOV DX,318h
MOV AL,12h
OUT DX,AL
```

**Note:** The S5U13503P00C is normally configured for indexed mapping, not direct mapping. Refer to the *S5U13503P00C Evaluation Board User's Manual* for more information configuring the S5U13503P00C board for indexed or direct mapping.

## 5.4 Split Screen

This section describes how to create a split screen for both single and dual LCD panels. For single panel displays, the Screen 1 Display Line Count Registers are used. For dual panel displays, the Screen 2 Display Start Address Registers are used.

### Registers

**AUX[0A] Screen 1 Display Line Count Register (LSB)**
I/O address = 1010b, Read/Write

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Screen 1 Display Line Count | Screen 1 Display Line Count | Screen 1 Display Line Count | Screen 1 Display Line Count | Screen 1 Display Line Count | Screen 1 Display Line Count | Screen 1 Display Line Count | Screen 1 Display Line Count |

**AUX[0B] Screen 1 Display Line Count Register (MSB)**
I/O address = 1011b, Read/Write

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| n/a | n/a | n/a | n/a | n/a | n/a | Screen 1 Display Line Count | Screen 1 Display Line Count |

**AUX[0A] bits 7–0, AUX[0B] bits 1–0**

**Screen 1 Display Line Count Bits [9:0]**

These bits are the eight LSB of a 10-bit value used to determine the number of lines displayed for screen 1. The remaining lines will automatically display from the screen 2 display start address. The 10-bit value programmed is the number of display lines - 1.

This register is used to enable the split screen display feature (single panel only) where two different images can be displayed at the same time on one display.

For example; AUX[0A] = 20h for a 320 × 240 display system. The display will display 20h + 1 = 33 lines on the upper part of the screen as dictated by the screen 1 display start address registers (AUX[06] and AUX[07]), and 240 - 33 = 207 lines will be displayed on the lower part of the screen as dictated by the screen 2 display start address registers (AUX[08] and AUX[09]).

Two different images can be displayed when using a dual panel configuration by changing the screen 2 display start address. However, by using this method screen 2 is limited to the lower half of the display.

This register is ignored in dual panel mode.

**Note:** See *"Display Start Address Registers"* on page 25 for additional register descriptions.

### Description

A split screen is generally considered as the presentation of two different images on the screen. Image 1 is shown on the top half and image 2 is shown on the bottom half of the screen. The system is *always* in split screen mode, on a single panel image 2 is displayed off screen; on a dual panel image 2 becomes the lower half of the panel.

### Single Panel LCD

The following is the procedure to show a split screen image on a 16 color 320 × 240 single panel LCD. For this example the S5U13503P00C is used with the Memory Interface set to 16 bits (required for 128K of display memory). In addition, the two images shown on the split screen are each 320 × 240; only a portion of each image is shown.

1. Determine whether the Display Start Address Registers refer to bytes or words.
   Since the Memory Interface is set to 16 bits, the Display Start Address Registers refer to words. Note that when addresses refer to words, the image must be aligned in memory such that the beginning is found on a word boundary (the least significant bit of the memory address must be 0).

2. Calculate the number of bytes per scan line.

$$16 \text{ colors} \rightarrow 4 \text{ bits per pixel}$$
$$4 \text{ bits per pixel} \rightarrow 2 \text{ pixels per byte}$$
$$\text{Number of bytes per scan line} = \frac{\text{Pixels per scan line}}{\text{Pixels per byte}} = \frac{320}{2}$$
$$= 160 \text{ bytes per scan line} = 00A0h \text{ bytes per scan line}$$

3. Determine the display memory location for image 1.
   For simplicity, assign the beginning of display memory as the starting address of image 1 (see Figure 5-4). For the S5U13503P00C, this address is D000:0000h, bank 0.

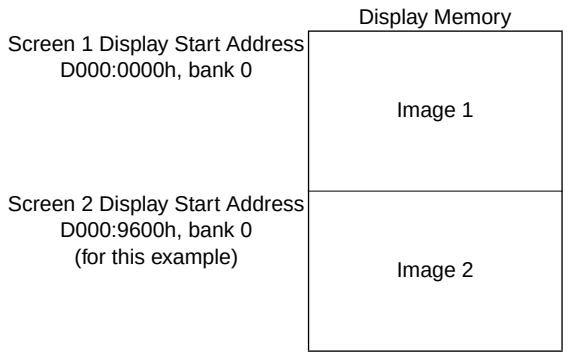



Figure 5-4 Memory Map for Split Screen

4. Program the Screen 1 Display Start Address Register to point to the beginning of image 1.
   Since image 1 is at the beginning of display memory, program the Screen 1 Display Start Address Register to 0000h.
   AUX[06h] = 00h
   AUX[07h] = 00h

5. Calculate the total number of bytes required for image

$$(\text{Bytes per scan line}) \times (\text{Number of scan lines for image 1}) = 160 \times 240$$
$$= 38400 \text{ bytes} = 9600h \text{ bytes}$$

6. Determine the display memory location for image 2.
   Place image 2 immediately after image 1 (see Figure 5-4). Assign the starting address for image 2 as follows:

$$\text{Image 2 address} = (\text{Base display memory address}) + (\text{Size of image 1})$$
$$= \{\text{D000:0000h, bank 0}\} + 0000\text{:}9600h$$
$$= \{\text{D000:9600h, bank 0}\}$$

   Note that if the image 2 address is larger than D000:FFFFh, then switch to bank 1, reset the segment to D000h, and keep the offset. For example, if the image 2 address were {D001:9200h, bank 0}, then this address must be changed to {D000:9200h, bank 1}.

7. Program the Screen 2 Display Start Address Register to point to the beginning of image 2. Image 2 is placed right after image 1, as shown below:

$$\text{Screen 2 display start address} = \text{Screen 1 display start address} + \frac{\text{Size of image 1 in bytes}}{\text{2 bytes per word}}$$

$$= 0000\text{h} + \frac{9600\text{h}}{2} = 4\text{B00h}$$

AUX[08h] = 00h
AUX[09h] = 4Bh

8. Program the Screen 1 Display Line Count Register.
The Display Line Count Register indicates how many lines of the first screen should be shown *minus 1*. By changing the line count, image 2 appears to move up or down the display.

   - If the line count is set to the maximum number of visible scan lines - 1, only image 1 is shown.

$$\text{Visible scan lines} - 1 = 240 - 1 = 239 = 00\text{EFh}$$

   AUX[0Ah] = LSB of (visible scan lines - 1) = EFh
   AUX[0Bh] = MSB of (visible scan lines - 1) = 00h

   - If the line count is set to 0, then the first scan line of image 1 is shown followed by the first part of image 2. It is not possible to show only image 2 by changing the line count. If only image 2 needs to be shown, reprogram the Screen 1 Display Start Address Registers to point to the beginning of image 2. Once both Screen 1 and 2 Display Start Address Registers point to the same image, the line count has no visible effect.
   AUX[0Ah] = 00h
   AUX[0Bh] = 00h

   - If the line count is set to 99, then the first 100 scan lines of image 1 are shown, following by the first part of image 2 (see Figure 5-5).
   AUX[0Ah] = 63h (99 decimal)
   AUX[0Bh] = 00h

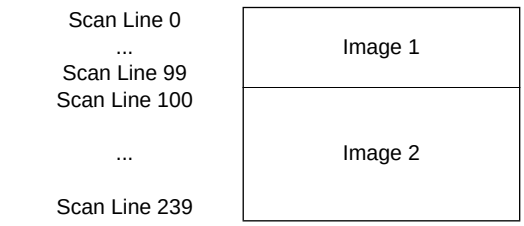



Figure 5-5 320 × 240 Single Panel for Split Screen

9. Write both image 1 and image 2 to their respective locations in display memory.

## *Dual Panel LCD*

The following is the procedure to show a split screen image on a 4 gray shade 640 × 480 dual panel LCD. For this example the S5U13503P00C is used with the Memory Interface set to 16 bits (required for 128K of display memory). In addition, the two images shown on the split screen are each 640 × 240.

1. Determine whether the Display Start Address Registers refer to bytes or words.
Since the Memory Interface is set to 16 bits, the Display Start Address Registers refer to words. Note that when addresses refer to words, the image must be aligned in memory such that the beginning is found on a word boundary (the least significant bit of the memory address must be 0).

2. Calculate the number of bytes per scan line.

$$4 \text{ gray shades} \rightarrow 2 \text{ bits per pixel}$$
$$2 \text{ bits per pixel} \rightarrow 4 \text{ pixels per byte}$$
$$\text{Number of bytes per scan line} = \frac{\text{Pixels per scan line}}{\text{Pixels per byte}} = \frac{640}{4}$$
$$= 160 \text{ bytes per scan line} = 00A0h \text{ bytes per scan line}$$

3. Determine the display memory location for image 1.
   For simplicity, assign the beginning of display memory as the starting address of image 1 (see Figure 5-4). For the S5U13503P00C, this address is D000:0000h, bank 0.

4. Program the Screen 1 Display Start Address Register to point to the beginning of image 1.
   Since image 1 is at the beginning of display memory, program the Screen 1 Display Start Address Register to 0000h.
   AUX[06h] = 00h
   AUX[07h] = 00h

5. Calculate the total number of bytes required for image 1.

$$(\text{Bytes per scan line}) \times (\text{Number of scan lines for image 1}) = 160 \times 240$$
$$= 38400 \text{ bytes} = 9600h \text{ bytes}$$

6. Determine the display memory location for image 2.
   Place image 2 immediately after image 1 (see Figure 5-4). Assign the starting address for image 2 as follows:

$$\text{Image 2 address} = (\text{Base display memory address}) + (\text{Size of image 1})$$
$$= \{D000:0000h, \text{ bank } 0\} + 0000:9600h$$
$$= \{D000:9600h, \text{ bank } 0\}$$

   Note that if the image 2 address is larger than D000:FFFFh, then switch to bank 1, reset the segment to D000h, and keep the offset. For example, if the image 2 address were {D001:9200h, bank 0}, then this address must be changed to {D000:9200h, bank 1}.

7. Program the Screen 2 Display Start Address Register to point to the beginning of image 2.
   Image 2 is placed right after image 1, as shown below:

$$\text{Screen 2 display start address} = \text{Screen 1 display start address} + \frac{\text{Size of image 1 in bytes}}{\text{2 bytes per word}}$$
$$= 0000h + \frac{9600h}{2} = 4B00h$$

   AUX[08h] = 00h
   AUX[09h] = 4Bh

8. Write both image 1 and image 2 to their respective locations in display memory.

### Notes

When using a dual panel, the Screen 1 Display Line Count Register is ignored by the S1D13503. Once the two Display Start Address Registers are programmed, the top panel will show the beginning of image 1, and the bottom panel will show the beginning of image 2 (see Figure 5-6).


Screen 1 Display Line Count is ignored;
Image 1 always has half the total number of scan lines (240 in this example).

Figure 5-6 640 × 480 Dual Panel for Split Screen

Each image can be scrolled or panned by appropriate programming of the respective Display Start Address Registers. The following are some examples:

- To scroll image 1 up, the Screen 1 Start Address Register must point to the following scan line.

$$\text{Screen 1 display start address} = \text{Screen 1 display start address} + \frac{\text{Number of bytes per scan line}}{\text{2 bytes per word}}$$

  AUX[06h] = LSB of Screen 1 Display Start Address
  AUX[07h] = MSB of Screen 1 Display Start Address

- To scroll image 2 down, the Screen 2 Start Address Register must point to the previous scan line.

$$\text{Screen 2 display start address} = \text{Screen 2 display start address} - \frac{\text{Number of bytes per scan line}}{\text{2 bytes per word}}$$

  AUX[08h] = LSB of Screen 2 Display Start Address
  AUX[09h] = MSB of Screen 2 Display Start Address

- To pan image 1 to the right by a group of pixels, the Screen 1 Start Address Register must be increased by 1.

$$\text{Screen 1 display start address} = \text{Screen 1 display start address} + 1$$

  AUX[06h] = LSB of Screen 1 Display Start Address
  AUX[07h] = MSB of Screen 1 Display Start Address
  See *"Panning Right and Left"* on page 38 for more information.

- To pan image 2 to the left by a group of pixels, the Screen 2 Start Address Register must be decreased by 1.

$$\text{Screen 2 display start address} = \text{Screen 2 display start address} - 1$$

  AUX[08h] = LSB of Screen 2 Display Start Address
  AUX[09h] = MSB of Screen 2 Display Start Address
  See *"Panning Right and Left"* on page 38 for more information.

### Displaying a Single Image on a Dual Panel

The following is the procedure to show a single image on a dual panel LCD. In this procedure the single image is broken into two smaller images; the first half of the image is placed on the top panel and the second half is placed on the bottom panel. For this example the S5U13503P00C is used with a 4 gray shade 640 × 480 dual panel LCD; the Memory Interface is set to 16 bits to support 128K of display memory.

1. Determine whether the Display Start Address Registers refer to bytes or words.
   Since the Memory Interface is set to 16 bits, the Display Start Address Registers refer to words. Note that when addresses refer to words, the image must be aligned in memory such that the beginning is found on a word boundary (the least significant bit of the memory address must be 0).

2. Calculate the number of bytes per scan line.

$$4 \text{ gray shades} \rightarrow 2 \text{ bits per pixel}$$

$$2 \text{ bits per pixel} \rightarrow 4 \text{ pixels per byte}$$

$$\text{Number of bytes per scan line} = \frac{\text{Pixels per scan line}}{\text{Pixels per byte}} = \frac{640}{4}$$

$$= 160 \text{ bytes per scan line} = 00A0\text{h bytes per scan line}$$

3. Determine the display memory location for the first half of the image.
   For simplicity, assign the beginning of display memory as the starting address of the image's first half (see Figure 5-7). For the S5U13503P00C, this address is D000:0000h, bank 0.

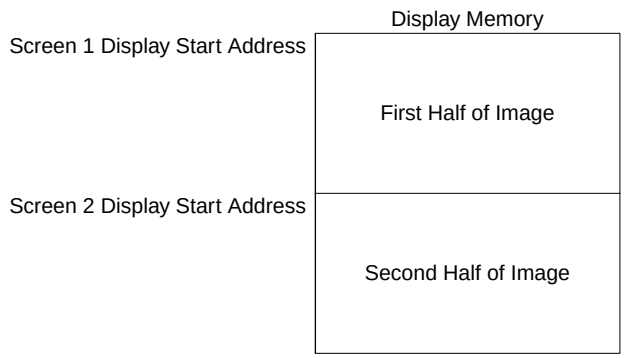



Figure 5-7 Memory Map for a Dual Panel Showing a Single Image

4. Program the Screen 1 Display Start Address Register to point to the beginning of the first half of the image.
   Since the first half is at the beginning of display memory, program the Screen 1 Display Start Address Register to 0000h.
   AUX[06h] = 00h
   AUX[07h] = 00h

5. Determine the size of the image's first half.

$$\text{Vertical size of first half of image} = \text{Vertical size of panel 1}$$

$$= \frac{\text{Number of scan lines in display}}{2} = \frac{480}{2} = 240 \text{ scan lines}$$

$$\text{Size} = \frac{\text{Display width in pixels}}{\text{Pixels per byte}} \times (\text{Number of scan lines in first half of image})$$

$$= \frac{640}{4} \times 240 = 38400 \text{ bytes} = 9600\text{h bytes}$$

6. Determine the display memory location for the second half of the image.
   Place the second half of the image immediately after the first half (see Figure 5-7). Assign the starting address for the second half as follows:

$$\begin{aligned}\text{Address of second half of image} &= (\text{Base display memory address}) + (\text{Size of first half of image})\\ &= \{\text{D000:0000h, bank 0}\} + \text{0000:9600h}\\ &= \{\text{D000:9600h, bank 0}\}\end{aligned}$$

   Note that if the address of the second half of the image is larger than D000:FFFFh, then switch to bank 1, reset the segment to D000h, and keep the offset. For example, if the address of the second half of the image were {D001:9200h, bank 0}, then this address must be changed to {D000:9200h, bank 1}.

7. Program the Screen 2 Display Start Address Register to point to the beginning of the second half of the image.
   The second half of the image is placed right after the first half, as shown below:

$$\text{Screen 2 display start address} = \text{Screen 1 display start address} + \frac{\text{Size of first half of image in bytes}}{\text{2 bytes per word}}$$

$$= \text{0000h} + \frac{\text{9600h}}{2} = \text{4B00h}$$

   AUX[08h] = 00h
   AUX[09h] = 4Bh

8. Write both the first and second halves of the image to their respective locations in display memory.

## 5.5 Panning and Scrolling

Panning and scrolling are typically used to show an image which is too large to be shown completely on an LCD panel. Although the image is stored entirely in display memory, only a small portion is actually visible on the LCD panel. This visible portion is called the *viewport*; the user moves this viewport over different portions of the image by panning and scrolling. *Panning* moves the viewport right or left. *Scrolling* moves the viewport up or down.

### *Initialization*

To pan and scroll over a large image, the S1D13503 registers must first be initialized and the image written to display memory. To do so, initialize the registers as described in Section 2, *"Initializing the S1D13503"* on page 2, but with the following exception: the Address Pitch Adjustment Register in the S1D13503 must be set to create a virtual display; see Section 5.1, *"Virtual Displays"* on page 27 for more information.

### *Panning Right and Left*

To pan to the right, increase the value in the Screen 1 Display Start Address Register. To pan to the left, decrease the value in the Screen 1 Display Start Address Register.

Note that the S1D13503 can pan right or left by either 1, 2, 4, 8, or 16 pixels. This is because the Screen 1 Display Start Address Register refers to either bytes or words (see *"S5U13503P00C Evaluation Board Display Memory"* on page 24), and a byte can represent 1, 2, 4, or 8 pixels, and so a word can represent 2, 4, 8, or 16 pixels; see Table 5-1 below:

Table 5-1 Smallest Number of Pixels for Panning

| Memory Interface | Colors/ Gray Levels | Pixels per Byte | Smallest Number of Pixels for Panning |
|---|---|---|---|
| 8 bits | 2 | 8 | 8 |
| | 4 | 4 | 4 |
| | 16 | 2 | 2 |
| | 256 | 1 | 1 |
| 16 bits | 2 | 8 | 16 |
| | 4 | 4 | 8 |
| | 16 | 2 | 4 |
| | 256 | 1 | 2 |

### *Scrolling Up and Down*

To scroll up, increase the value in the Screen 1 Display Start Address Register by the number of bytes in one *virtual* scan line. To scroll down, decrease the value in the Screen 1 Display Start Address Register by the number of bytes in one *virtual* scan line.

A virtual scan line is in reference to a virtual display, in which an image larger than the physical size of the LCD is stored. The number of bytes in a virtual scan line is the number of bytes required to store one horizontal line of pixels in the virtual image.

**Example 16**

**Scroll down one line for a 16 gray shade 640 x 200 virtual image using a 320 x 240 single panel LCD. The Memory Interface is set to 16 bits to support 128K of display memory. Also describe how to scroll in a dual panel LCD.**

1. Calculate the number of bytes in a virtual scan line.

$$\frac{\text{Number of horizontal pixels in virtual image}}{\text{Number of pixels per word}} = \frac{\text{640 pixels per scan line}}{\text{2 pixels per byte}} = \text{320 bytes per scan line}$$

2. Add the number of *words* in a virtual scan line to the Screen 1 Display Start Address Register. In this example the Screen 1 Display Start Address points to the beginning of the image.

$$\text{Screen 1 display start address} = \text{Screen 1 display start address} + \frac{\text{Number of bytes in a virtual scan line}}{\text{2 bytes per word}}$$

$$= 0000\text{h} + \frac{320}{2}$$

$$= 00A0\text{h}$$

3. Program the Screen 1 Display Start Address.
   AUX[06h] = A0h
   AUX[07h] = 00h
4. This step is for dual panels only.
   Add the number of *words* in a virtual scan line to the Screen 2 Display Start Address Register. In this example the Screen 2 Display Start Address has previously been initialized as described in *"Displaying a Single Image on a Dual Panel"* on page 36.

$$\text{Screen 2 display start address} = \text{Screen 2 display start address} + \frac{\text{Number of bytes in a virtual scan line}}{\text{2 bytes per word}}$$

5. This step is for dual panels only.
   Program the Screen 2 Display Start Address.
   AUX[08h] = least significant byte of "Screen 2 Display Start Address"
   AUX[09h] = most significant byte of "Screen 2 Display Start Address"

## 5.6 Power Saving

The following section introduces the power saving capabilities of the S1D13503. A detailed description of the Power Save Register is provided, followed by a description of the power save modes.

### Registers

Register bits discussed in this section are highlighted.

| **AUX[03] Mode Register 1**<br>I/O address = 0011b, Read/Write | | | | | | | |
|---|---|---|---|---|---|---|---|
| PS<br>Bit 1 | PS<br>Bit 0 | LCD Signal<br>State | LUT<br>Bypass | LCD Data<br>Width Bit 1 | BW /<br>256 colors | Color Mode | Line Byte<br>Count Bit 8 |

**bits 7–6 PS Bits [1:0]**

Selects the Power Save Modes as shown in the following table. The PS bits [1:0] go low on RESET.

Table 5-2 Power Save Mode Selection

| PS1 | PS0 | Mode Activated |
|---|---|---|
| 0 | 0 | Normal Operation |
| 0 | 1 | Power Save Mode 1 |
| 1 | 0 | Power Save Mode 2 |
| 1 | 1 | Reserved |

For more details refer to "*Power Save Modes*" in the *S1D13503 Hardware Functional Specification*.

### Power Save Modes

Two software-controlled Power Save Modes have been incorporated into the S1D13503 to accommodate the important need for power reduction in the hand-held devices market. These modes can be enabled by setting the 2 Power Save bits (AUX[03h] bits 7–6).

The various settings are:

Table 5-3 Power Save Mode Selection

| Bit 7 | Bit 6 | Mode Activated |
|---|---|---|
| 0 | 0 | Normal Operation |
| 0 | 1 | Power Save Mode 1 |
| 1 | 0 | Power Save Mode 2 |
| 1 | 1 | Reserved |

#### Power Save Mode 1

Power Save Mode 1 would typically be used when power savings are required and display memory accesses may occur. The disadvantage is that since the oscillator is running, this mode consumes more power that Power Save Mode 2.

#### Power Save Mode 2

Power Save Mode 2 is typically used when display memory accesses would not occur.

#### Power Save Mode Function Summary

Table 5-4 Power Save Mode Function Summary

| Function | Power Save Mode (PSM) | | | |
|---|---|---|---|---|
| | Normal (Active) | PSM1 | | PSM2 |
| | | State 1 | State 2 | |
| Display Active? | Yes | No | No | No |
| I/O Access Possible? | Yes | Yes | Yes | Yes |
| Memory Access Possible? | Yes | Yes | No | No |
| Sequence Controller Running? | Yes | No | No | No |
| Internal Oscillator Disabled? | No | No | No | Yes |

**Notes:** 1. When programming the PS bits perform a read/modify/write operation so as not to destroy any other data in the register.
2. Refer to the programming example in *"Advanced Functions"* on page 51.

### Programming to Enter Power Down Mode

If the LCDENB pin is used to control an external LCDBIAS power supply, the following sequence is recommended to prevent damage to the panel. Panel damage can occur if the LCDBIAS is present without active panel sync signals. Note the LCDENB pin is controlled by AUX[01h] bit 4 (LCDE).

1. Write '0' to bit 7 of AUX[01h] to turn off the display.
2. Write to bit 4 of AUX[01h] with value 'x' as appropriate to disable the specific power supply design. For the S5U13503P00C, write '0' to disable the power supply.
3. Wait until the LCDBIAS power supply reaches zero volts. This delay time is dependent upon the specific power supply design, as well as the display's electrical characteristics. For the S5U13503P00C, this time is about 0.5 seconds.
4. Enter power saving mode by writing the appropriate bits 7–6 of AUX[03h].

### Programming to Exit Power Down Mode

When the LCDENB pin is used to control an external LCDBIAS power supply, the following sequence is recommended to exit power down mode. Note the LCDENB pin is controlled by AUX[01h] bit 4 (LCDE).

1. Exit power saving mode by writing 00b to bits 7–6 of AUX[03h].
2. Write to bit 4 of AUX[01h] with value 'x' as appropriate to enable the specific power supply design. For the S5U13503P00C, write '1' to enable the power supply. Note that no delay is required before applying power.
3. Write '1' to bit 7 of AUX[01h] to turn on the display.

# 6 IDENTIFYING THE S1D13503

To identify the LCD controller upon power up / reset, perform the following steps:

1. Power up LCD controller.
2. Read AUX[0Eh], bits 5–4. Refer to Table 6-1 below to decode chip ID.

Table 6-1 ID-Bit Usage

| | Chip | AUX[0E] | |
|---|---|---|---|
| | | Bit 5 | Bit 4 |
| Power On or RESET | S1D13503 | 0 | 0 |
| | reserved | 0 | 1 |
| | SED1352F$_{0B}$ /F$_{1B}$/D$_{0B}$ | 1 | 0 |
| | SED1352F$_{0A}$ /F$_{1A}$/D$_{0A}$ | 1 | 1 |

**Note:** If the registers have already been initialized after power up, the ID bits in AUX[0Eh] cannot be used since these bits are also used for the RGB index. It is recommended to always store the chip ID immediately after power up and before any register initialization.

# 7 PROGRAMMING THE S1D13503

The purpose of this section is to show how to program the S1D13503 exercising the specific capabilities of this chip. A series of functions written in 'C' will be presented, each illustrating a basic feature of the S1D13503. These functions are written for the S5U13503P00C evaluation board, and are combined under a menu-driven program called 13503DEMO.EXE.

**Note:** The sample code will not run on a display larger than 320 × 240, and will use either 256 colors or 16 gray shades in most of the examples.

This program accepts the following command line options:

**13503DEMO t=n x=n y=n d=n i=n p=n [f=n] [/?]**

where: **t** = **SINGLE** | **DUAL**
**x** = horizontal panel size in pixels from 1 to 320 (decimal)
**y** = vertical panel size in pixels from 1 to 240 (decimal)
**d** = **COLOR | MONO**
**i** = **4 | 8** (4-bit or 8-bit interface to panel)
**p** = **300** | **310**...**360** | **370** (port address in hex) (indexed I/O addressing selected by default)
**f** = **1 | 2** (format for color 8-bit panel interface)
**/?** = show this help screen

For example, if there is a 320 × 240 color single panel LCD, 8-bit interface, format 2, with a port address of 310h, type

13503DEMO t=SINGLE x=320 y=240 d=COLOR i=8 p=310 f=2

When 13503DEMO is started, output will be sent to the standard output device. This output will present a menu of numbered options:

```
S1D13503F00A/S5U13503P00C DEMO
PROGRAM
Press 1 to read registers
Press 2 to show color/gray shade bar
Press 3 to show split screen
Press 4 to show panning and scrolling
Press 5 to start power saving
Press ESC to quit
```

Figure 7-1 Display for 13503DEMO.EXE

## *7.1 Main Loop Code*

```
//-----------------------------------------------------------------------
//
// FUNCTION: main()
//
// DESCRIPTION: Start of demo program.
//
// INPUTS: Command line arguments.
// RETURN VALUE: None.
//
//-----------------------------------------------------------------------

void main(char argc, char **argv)
{
int ch;

CheckArguments(argc, argv);
printf("Initializing\n");
Initialize();
SetDisplay(OFF);
ClearLCDScreen();

switch (GetID(PanelPortAddr))
   {
   case ID_1352FOA:
      printf("Detected SED1352F0A.\n\n");
      Quit();
      break;

   case ID_1352FOB:
      printf("Detected SED1352F0B.\n\n");
      Quit();
      break;

   case ID_13503:
      printf("Detected S1D13503F00A.\n");
      break;

   default:
      printf("ERROR: Could not detect chip.\n\n");
      Quit();
      break;
   }

ShowMenu();

while ((ch = getch()) != ESC)
   {
   switch (ch)
      {
      case '1':
         ShowRegisters();
         break;

      case '2':
         GrayShadeBars();
         break;

      case '3':
         SplitScreen();
         break;

      case '4':
         PanScroll();
```

```
            break;

         case '5':
            PowerSaving();
            break;

         case ESC:
            exit(0);
         }
      }
}
```

## 7.2 Initialization Code

```
//-------------------------------------------------------------------------
//
// FUNCTION: Initialize()
//
// DESCRIPTION: Intialize S1D13503 registers.
//
// INPUTS: This function looks at the followingl global variables to
//         determine the appropriate register settings:
//            PanelX, PanelY, PanelType
//
// OUTPUTS: The following global variables are changed:
//            PanelGrayLevel, BytesPerScanLine
//
//-------------------------------------------------------------------------

void Initialize(void)
{
static unsigned int val, val2;
static unsigned int x;


if (PanelD == PANEL_MONO)
   PanelGrayLevel = 16;
else
   PanelGrayLevel = 256;

//--------------------------------------

//
// Mode Register:
//   Display = OFF
//   Panel = SINGLE
//   Mask XSCL = NOT MASKED
//   LCDE = NOT ENABLED
//   Gray Shade/Color = 16 Gray Shades (bit is ignored for 256 colors)
//   LCD Data Width = 8 bit data transfer
//   Memory Interface = 16 bits
//   RAMS = Addressing for 8Kx8 SRAM
//
val = 0x0C;

if (Interface == 4)
   val &= 0xfb;   // Clear AUX[01] bit 2 so that Memory Interface = 4 bits

if (PanelType == TYPE_DUAL)
   val |= 0x40;   // Set panel type to DUAL

WriteRegister(1, val);   // Write to Mode Register

//--------------------------------------

//
// Line Byte/Word Count Register
//
// Bits 0-7 are in AUX[2], Bit 8 is in AUX[3].
//
// Because the Memory Interface is set to 16 bits, the
// Line Byte/Word Count Register counts in words.
// To calculate the Line Byte Count for different numbers of
// gray shades/colors, use the following formula:
//
//            BitsPerPixel
//       ----------------------  x Horizontal Resolution  - 1
```

```
//       Memory Interface Width
//
switch (PanelGrayLevel)
   {
   case 2:
      val = (PanelX / 16) - 1;         // For black and white mode
      break;

   case 4:
      val = (PanelX / 8) - 1;          // For 4 gray shades/colors
      break;

   case 16:
      val = (PanelX / 4) - 1;          // For 16 gray shades/colors
      break;

   case 256:
      val = (PanelX / 2) - 1;          // For 256 colors
      break;
   }

WriteRegister(2, val & 0xff);          // Line Byte/Word Count Register

val2 = (val >> 8) & 0x01;

if (PanelD == PANEL_COLOR)
   {
   val2 |= 0x06;        // Select color mode and 256 colors

   if ((Interface == 8) && (PanelF == 2))
      val2 |= 0x08;     // Select format 2
   }

WriteRegister(3, val2);   // Mode Register 1


//
// BytesPerScanLine is a global variable
//
switch (PanelGrayLevel)
   {
   case 2:
      BytesPerScanLine = (PanelX / 8);
      break;

   case 4:
      BytesPerScanLine = (PanelX / 4);
      break;

   case 16:
      BytesPerScanLine = (PanelX / 2);
      break;

   case 256:
      BytesPerScanLine = PanelX;
      break;
   }

//--------------------------------------

//
// Total Display Line Count Register
// Screen 1 Display Line Count Register
//
```

```
// To show a full image on Screen 1, copy the Total Display Line Count
// into the Screen 1 Display Line Count.
//

//
// Old programs had previously assumed that all panels smaller
// than 400 lines use a 4 bit interface. However, newer panels
// which are less than 400 lines may use an 8 bit interface.
// Consequently this program must be told which interface to use.
//
// Set the Mask XSCL bit to MASKED (1) when using a 4 bit interface.
//
if (Interface == 4)
   {
   val = ReadRegister(1);
   val &= 0xfb;   // Set LCD Data Width to 4 bit data transfer
   val |= 0x20;   // Set Mask XSCL to MASKED
   WriteRegister(1, val);  // Write to Mode Register; LCD Data Width = 4 bits
   }

val = PanelY;

//
// A dual panel LCD will, of course, have two panels.  Each panel will
// show either the top or bottom half of the image, which is half of the
// vertical resolution.
//
if (PanelType == TYPE_DUAL)
   val /= 2;

--val;

WriteRegister(4, val & 0xff);    // Write to Total Display Line Count Reg
WriteRegister(0x0a, val & 0xff); // Write to Screen 1 Display Line Count Reg
WriteRegister(5, (val >> 8) & 0x03); // Total Disp Line Cnt (MSB)/WF Count Reg
WriteRegister(0x0b, (val >> 8) & 0x03);  // Scrn 1 Disp Line Count Reg (MSB)

//--------------------------------------

//
// Set Screen 1 Display Start Address to beginning of video memory
//
WriteRegister(6, 0);  // Write to Screen 1 Display Start Address Register
WriteRegister(7, 0);

//--------------------------------------

//
// Screen 2 Display Start Address Register
//
// If using a dual panel, the Screen 2 Display Start Address must point
// to the second half of the image in video memory.
//
if (PanelType == TYPE_DUAL)
   {
   val = (unsigned int) ((ReadRegister(3) & 0x01) << 8) | ReadRegister(2);
   ++val;

   val *= (PanelY / 2);
   WriteRegister(8, val & 0xff);
   WriteRegister(9, val >> 8);
   }
else
   {
```

```
   //
   // On a single panel, Screen 1 was programmed to show all of its
   // lines.  Consequently Screen 2 will not be seen, and so the
   // Screen 2 Display Start Address will have no observable effect.
   // For convenience, set the screen 2 address to 0.
   //
   WriteRegister(8, 0);
   WriteRegister(9, 0);
   }

//--------------------------------------

//
// Set Horizontal Non-Display Period to 0 to use fixed default non-display
// period
//
WriteRegister(0x0c, 0);

//--------------------------------------

//
// Set Address Pitch Adjustment to 0
//
WriteRegister(0x0d, 0);  // Write to Address Pitch Adjustment Register

//--------------------------------------

//
// Update Lookup Table for 16 gray shades/ 256 colors
//
if (PanelD == PANEL_MONO)
   {
   for (x = 0; x < 16; ++x)
      {
      WriteRegister(0x0e, x);
      WriteRegister(0x0f, MonoLUT16[x]);
      }
   }
else
   {
   for (x = 0; x < 16; ++x)
      {
      WriteRegister(0x0e, x);    // Auto-increment mode selected

      WriteRegister(0x0f, ColorLUT256Red[x]);
      WriteRegister(0x0f, ColorLUT256Green[x]);
      WriteRegister(0x0f, ColorLUT256Blue[x]);
      }
   }

//--------------------------------------

//
// Now that system is initialized, set DISPLAY ON and enable LCDE
//
val = ReadRegister(1);
val |= 0x90;   // DISPLAY ON, LCDE enabled
WriteRegister(1, val);
}


//-------------------------------------------------------------------------
//
// GetID()
```

```
//
// This function returns the Chip ID.
//
//---------------------------------------------------------------------------

static unsigned char GetID(int PortAddr)
{
static unsigned char ChipID;

ChipID = ID_NOT_DETECTED;


//
// If the chip was just powered up, and no registers have been initialized,
// then use the following code:
//
outp(PortAddr, 0x0e);

switch (inp(PortAddr+1) & 0x30)
   {
   case 0x00:
      ChipID = ID_13503;
      break;

   case 0x20:
      ChipID = ID_1352F0B;
      break;

   case 0x30:
      ChipID = ID_1352F0A;
      break;

   default:
      ChipID = ID_NOT_DETECTED;
      break;
   }

return(ChipID);
}
```

## 7.3 Advanced Functions

```
#define VIRTUAL_X    (360L)
#define VIRTUAL_Y    (360L)

//-------------------------------------------------------------------------
//
// FUNCTION: ShowRegisters()
//
// DESCRIPTION: Shows the contents of the S1D13503 registers.
//
// INPUTS: None.
// RETURN VALUE: None.
//
//-------------------------------------------------------------------------

void ShowRegisters(void)
{
static unsigned char x;
static unsigned char red, green, blue;

printf("S1D13503 Registers: ");

for (x = 0; x < 16; ++x)
   printf("%02X ", ReadRegister(x));

printf("\nS1D13503 Lookup Table: ");

for (x = 0; x < 16; ++x)
   {
   WriteRegister(0x0e, x);

   red = ReadRegister(0x0f);
   green = ReadRegister(0x0f);
   blue = ReadRegister(0x0f);

   if (x % 7 == 0)
      printf("\n");

   printf("(%02X,%02X,%02X) ", red, green, blue);
   }

ShowMenu();
}


//-------------------------------------------------------------------------
//
// FUNCTION: GrayShadeBars()
//
// DESCRIPTION: Displays a series of vertical bars, each with a
//              different color/gray shade.
//              For color displays, bars are shown for 4, 16, and 256 colors.
//              For monochrome displays, bars are shown for black and white,
//              4, and 16 gray shades.
//
// INPUTS: None.
//
// RETURN VALUE: None.
//
//-------------------------------------------------------------------------

void GrayShadeBars(void)
{
static unsigned int val, val2, x;
```

```
static unsigned char _far *pVideo;
static char Gray4[] =  "Vertical Bars at 4 Gray Shades";
static char Color4[] = "Vertical Bars at 4 Colors";
static char Gray16[] = "Vertical Bars at 16 Gray Shades";
static char Color16[] = "Vertical Bars at 16 Colors";

static char *str;


printf("Displaying Vertical Bars\n");

Initialize();

SetDisplay(OFF);
ClearLCDScreen();



//
// Access memory banks
//
FP_SEG(pVideo) = 0xd000;
FP_OFF(pVideo) = 0x0000;

//--------------------------------------

if (PanelD == PANEL_MONO)
   {
   //
   // Select black and white mode
   //
   val = ReadRegister(3);
   val |= 0x04;            // Set AUX[03] bit 2
   val &= 0xfd;            // Clear AUX[03] bit 1
   WriteRegister(3, val);


   //
   // Update Line Byte/Word Count register for black and white.
   //
   // Since black and white has 8 pixels per byte, there
   // are ((x horizontal pixels)/8) bytes per scan line.  This means that
   // there are ((x horizontal pixels)/16) words per scan line.
   //
   // Since the Memory Interface is set to 16 bits, the Line Byte/Word Count
   // refers to words.
   //
   val = (PanelX / 16) - 1;
   BytesPerScanLine = (PanelX / 8);

   WriteRegister(2, val & 0xff);         // Line Byte Count Register

   val2 = ReadRegister(3);
   val2 &= 0xfe;                         // Clear bit 0
   val2 |= (val >> 8) & 0x01;
   WriteRegister(3, val2);               // Mode Register 1

   PanelGrayLevel = 2;
   ShowVerticalBars(pVideo, 0);

   //
   // Show text.  The lightest gray shade is set to PanelGrayLevel-1.
   //
   ShowText(pVideo, BANK0, "Vertical Bars for Black and White", PanelGrayLevel-1);
```

```
   SetDisplay(ON);
   Delay(2000);
   }

//--------------------------------------

SetDisplay(OFF);
ClearLCDScreen();

//
// Select 4 gray shades/colors
//
if (PanelD == PANEL_MONO)
   {
   val = ReadRegister(1);
   val &= 0xf7;            // Clear AUX[01] bit 3
   WriteRegister(1, val);

   val = ReadRegister(3);
   val &= 0xf9;            // Clear AUX[03] bits 1 and 2
   WriteRegister(3, val);

   //
   // Update Lookup Table for 4 gray shades
   //
   for (x = 0; x < 16; ++x)
      {
      WriteRegister(0x0e, x);
      WriteRegister(0x0f, MonoLUT4[x]);
      }

   str = Gray4;
   }
else   // 4 colors
   {
   val = ReadRegister(1);
   val &= 0xf7;            // Clear AUX[01] bit 3
   WriteRegister(1, val);

   val = ReadRegister(3);
   val &= 0xfb;            // Clear AUX[03] bit 2
   val |= 0x02;            // Set AUX[03] bit 1
   WriteRegister(3, val);

   //
   // Update Lookup Table for 4 colors
   //
   for (x = 0; x < 16; ++x)
      {
      WriteRegister(0x0e, x);

      WriteRegister(0x0f, ColorLUT4Red[x]);
      WriteRegister(0x0f, ColorLUT4Green[x]);
      WriteRegister(0x0f, ColorLUT4Blue[x]);
      }

   str = Color4;
   }

//
// Update Line Byte/Word Count register for 4 colors/gray shades
//
// Since 4 colors/gray shades corresponds to 4 pixels per byte, there
// are ((x horizontal pixels)/4) bytes per scan line.  This means that
```

```
// there are ((x horizontal pixels)/8) words per scan line.
//
// Since the Memory Interface is set to 16 bits, the Line Byte/Word Count
// refers to words.
//
val = (PanelX / 8) - 1;
BytesPerScanLine = (PanelX / 4);

WriteRegister(2, val & 0xff);         // Line Byte Count Register

val2 = ReadRegister(3);
val2 &= 0xfe;                         // Clear bit 0
val2 |= (val >> 8) & 0x01;
WriteRegister(3, val2);               // Mode Register 1

PanelGrayLevel = 4;
ShowVerticalBars(pVideo, 0);

//
// Show text.  The lightest color/gray shade is set to PanelGrayLevel-1.
//
ShowText(pVideo, BANK0, str, PanelGrayLevel-1);
ShowText(pVideo + BytesPerScanLine*8, BANK0, "BANK: 0", PanelGrayLevel-1);
SetDisplay(ON);
Delay(2000);

val = ReadRegister(0x0e);
val &= 0x3f;
val |= 0x40;
WriteRegister(0x0e, val);
ShowVerticalBars(pVideo, 0);

ShowText(pVideo, BANK0, str, PanelGrayLevel-1);
ShowText(pVideo + BytesPerScanLine*8, BANK0, "BANK: 1", PanelGrayLevel-1);
Delay(2000);

val &= 0x3f;
val |= 0x80;
WriteRegister(0x0e, val);
ShowVerticalBars(pVideo, 0);
ShowText(pVideo, BANK0, str, PanelGrayLevel-1);
ShowText(pVideo + BytesPerScanLine*8, BANK0, "BANK: 2", PanelGrayLevel-1);
Delay(2000);

val |= 0xc0;
WriteRegister(0x0e, val);
ShowVerticalBars(pVideo, 0);
ShowText(pVideo, BANK0, str, PanelGrayLevel-1);
ShowText(pVideo + BytesPerScanLine*8, BANK0, "BANK: 3", PanelGrayLevel-1);
Delay(2000);

//-------------------------------------

SetDisplay(OFF);
ClearLCDScreen();

//
// Select 16 colors/gray shades
//
if (PanelD == PANEL_MONO)
   {
   val = ReadRegister(1);
   val |= 0x08;            // Set AUX[01] bit 3
   WriteRegister(1, val);
```

```
   val = ReadRegister(3);
   val &= 0xf9;            // Clear AUX[03] bits 1 and 2
   WriteRegister(3, val);

   //
   // Update Lookup Table for 16 gray shades
   //
   for (x = 0; x < 16; ++x)
      {
      WriteRegister(0x0e, x);
      WriteRegister(0x0f, MonoLUT16[x]);
      }

   str = Gray16;
   }
else   // 16 colors
   {
   val = ReadRegister(1);
   val |= 0x08;            // Set AUX[01] bit 3
   WriteRegister(1, val);

   val = ReadRegister(3);
   val &= 0xfb;            // Clear AUX[03] bit 2
   val |= 0x02;            // Set AUX[03] bit 1
   WriteRegister(3, val);

   //
   // Update Lookup Table for 16 colors
   //
   for (x = 0; x < 16; ++x)
      {
      WriteRegister(0x0e, x);

      WriteRegister(0x0f, ColorLUT16Red[x]);
      WriteRegister(0x0f, ColorLUT16Green[x]);
      WriteRegister(0x0f, ColorLUT16Blue[x]);
      }

   str = Color16;
   }

//
// Update Line Byte Count register for 16 colors/gray shades
//
// Since 16 colors/gray shades corresponds to 2 pixels per byte, there
// are ((x horizontal pixels)/2) bytes per scan line.  This means that
// there are ((x horizontal pixels)/4) words per scan line.
//
// Since the Memory Interface is set to 16 bits, the Line Byte/Word Count
// refers to words.
//
val = (PanelX / 4) - 1;
BytesPerScanLine = (PanelX / 2);

WriteRegister(2, val & 0xff);         // Line Byte Count Register

val2 = ReadRegister(3);
val2 &= 0xfe;                         // Clear bit 0
val2 |= (val >> 8) & 0x01;
WriteRegister(3, val2);               // Mode Register 1

PanelGrayLevel = 16;
ShowVerticalBars(pVideo, 0);
```

```
//
// Show text.  The lightest color/gray shade is set to PanelGrayLevel-1.
//
ShowText(pVideo, BANK0, str, PanelGrayLevel-1);
SetDisplay(ON);
Delay(2000);

//-------------------------------------

if (PanelD == PANEL_COLOR)
   {
   SetDisplay(OFF);
   ClearLCDScreen();

   //
   // Select 256 colors
   //
   val = ReadRegister(3);
   val |= 0x06;            // Set AUX[03] bits 1 and 2
   WriteRegister(3, val);


   //
   // Update Lookup Table for 256 colors
   //
   for (x = 0; x < 16; ++x)
      {
      WriteRegister(0x0e, x);

      WriteRegister(0x0f, ColorLUT256Red[x]);
      WriteRegister(0x0f, ColorLUT256Green[x]);
      WriteRegister(0x0f, ColorLUT256Blue[x]);
      }

   //
   // Update Line Byte/Word Count register for 256 colors
   //
   // Since 256 colors have one pixel per byte, there
   // are (x horizontal pixels) bytes per scan line.  This means that
   // there are ((x horizontal pixels)/2) words per scan line.
   //
   // Since the Memory Interface is set to 16 bits, the Line Byte/Word Count
   // refers to words.
   //
   val = (PanelX / 2) - 1;
   BytesPerScanLine = PanelX;

   WriteRegister(2, val & 0xff);         // Line Byte Count Register

   val2 = ReadRegister(3);
   val2 &= 0xfe;                         // Clear bit 0
   val2 |= (val >> 8) & 0x01;
   WriteRegister(3, val2);               // Mode Register 1

   PanelGrayLevel = 256;
   ShowVerticalBars(pVideo, 0);

   //
   // Show text.  The lightest color is set to PanelGrayLevel-1.
   //
   ShowText(pVideo, BANK0, "Horizontal/Vertical Bars at 256 Colors", PanelGrayLevel-
1);
```

```
   SetDisplay(ON);
   Delay(2000);
   }
else
   SetDisplay(ON);

ShowMenu();
}

//-------------------------------------------------------------------------
//
// ShowText()
//
// DESCRIPTION: Writes text to the LCD panel.  Text must only contain
//              the letters A-Z, and the space character.  All other
//              characters are replaced by spaces.
//
// NOTES: It is assumed that a pixel set to a value of 0 represents the
//        background color (black).
//
//-------------------------------------------------------------------------

void ShowText(unsigned char _far *pVideoStart, unsigned char bank, char *str, int
color)
{
static const unsigned char *pFont;
static unsigned char _far *pVideoFirstColumn;
static unsigned char _far *pVideo;
static unsigned char ch;
static unsigned int y, val, Video;
static unsigned int count;

//
// Each letter in the font is 8 x 8 bits
//
#define MAX_FONT    97

static const unsigned char font[MAX_FONT][8] =
   { { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 },   // (blank)
     { 0x30, 0x78, 0x78, 0x30, 0x30, 0x00, 0x30, 0x00 },   // !
     { 0x6C, 0x6C, 0x6C, 0x00, 0x00, 0x00, 0x00, 0x00 },   // "
     { 0x6C, 0x6C, 0xFE, 0x6C, 0xFE, 0x6C, 0x6C, 0x00 },   // #
     { 0x30, 0x7C, 0xC0, 0x78, 0x0C, 0xF8, 0x30, 0x00 },   // $
     { 0x00, 0xC6, 0xCC, 0x18, 0x30, 0x66, 0xC6, 0x00 },   // %
     { 0x38, 0x6C, 0x38, 0x76, 0xDC, 0xCC, 0x76, 0x00 },   // &
     { 0x60, 0x60, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00 },   // '
     { 0x18, 0x30, 0x60, 0x60, 0x60, 0x30, 0x18, 0x00 },   // (
     { 0x60, 0x30, 0x18, 0x18, 0x18, 0x30, 0x60, 0x00 },   // )
     { 0x00, 0x66, 0x3C, 0xFF, 0x3C, 0x66, 0x00, 0x00 },   // *
     { 0x00, 0x30, 0x30, 0xFC, 0x30, 0x30, 0x00, 0x00 },   // +
     { 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x60 },   // ,
     { 0x00, 0x00, 0x00, 0xFC, 0x00, 0x00, 0x00, 0x00 },   // -
     { 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x00 },   // .
     { 0x06, 0x0C, 0x18, 0x30, 0x60, 0xC0, 0x80, 0x00 },   // /
     { 0x7C, 0xC6, 0xCE, 0xDE, 0xF6, 0xE6, 0x7C, 0x00 },   // 0
     { 0x30, 0x70, 0x30, 0x30, 0x30, 0x30, 0xFC, 0x00 },   // 1
     { 0x78, 0xCC, 0x0C, 0x38, 0x60, 0xCC, 0xFC, 0x00 },   // 2
     { 0x78, 0xCC, 0x0C, 0x38, 0x0C, 0xCC, 0x78, 0x00 },   // 3
     { 0x1C, 0x3C, 0x6C, 0xCC, 0xFE, 0x0C, 0x1E, 0x00 },   // 4
     { 0xFC, 0xC0, 0xF8, 0x0C, 0x0C, 0xCC, 0x78, 0x00 },   // 5
     { 0x38, 0x60, 0xC0, 0xF8, 0xCC, 0xCC, 0x78, 0x00 },   // 6
     { 0xFC, 0xCC, 0x0C, 0x18, 0x30, 0x30, 0x30, 0x00 },   // 7
     { 0x78, 0xCC, 0xCC, 0x78, 0xCC, 0xCC, 0x78, 0x00 },   // 8
     { 0x78, 0xCC, 0xCC, 0x7C, 0x0C, 0x18, 0x70, 0x00 },   // 9
```

```
{ 0x00, 0x30, 0x30, 0x00, 0x00, 0x30, 0x30, 0x00 },   // :
{ 0x00, 0x30, 0x30, 0x00, 0x00, 0x30, 0x30, 0x60 },   // ;
{ 0x18, 0x30, 0x60, 0xC0, 0x60, 0x30, 0x18, 0x00 },   // <
{ 0x00, 0x00, 0xFC, 0x00, 0x00, 0xFC, 0x00, 0x00 },   // =
{ 0x60, 0x30, 0x18, 0x0C, 0x18, 0x30, 0x60, 0x00 },   // >
{ 0x78, 0xCC, 0x0C, 0x18, 0x30, 0x00, 0x30, 0x00 },   // ?
{ 0x7C, 0xC6, 0xDE, 0xDE, 0xDE, 0xC0, 0x78, 0x00 },   // @
{ 0x30, 0x78, 0xCC, 0xCC, 0xFC, 0xCC, 0xCC, 0x00 },   // A
{ 0xFC, 0x66, 0x66, 0x7C, 0x66, 0x66, 0xFC, 0x00 },   // B
{ 0x3C, 0x66, 0xC0, 0xC0, 0xC0, 0x66, 0x3C, 0x00 },   // C
{ 0xF8, 0x6C, 0x66, 0x66, 0x66, 0x6C, 0xF8, 0x00 },   // D
{ 0xFE, 0x62, 0x68, 0x78, 0x68, 0x62, 0xFE, 0x00 },   // E
{ 0xFE, 0x62, 0x68, 0x78, 0x68, 0x60, 0xF0, 0x00 },   // F
{ 0x3C, 0x66, 0xC0, 0xC0, 0xCE, 0x66, 0x3E, 0x00 },   // G
{ 0xCC, 0xCC, 0xCC, 0xFC, 0xCC, 0xCC, 0xCC, 0x00 },   // H
{ 0x78, 0x30, 0x30, 0x30, 0x30, 0x30, 0x78, 0x00 },   // I
{ 0x1E, 0x0C, 0x0C, 0x0C, 0xCC, 0xCC, 0x78, 0x00 },   // J
{ 0xE6, 0x66, 0x6C, 0x78, 0x6C, 0x66, 0xE6, 0x00 },   // K
{ 0xF0, 0x60, 0x60, 0x60, 0x62, 0x66, 0xFE, 0x00 },   // L
{ 0xC6, 0xEE, 0xFE, 0xFE, 0xD6, 0xC6, 0xC6, 0x00 },   // M
{ 0xC6, 0xE6, 0xF6, 0xDE, 0xCE, 0xC6, 0xC6, 0x00 },   // N
{ 0x38, 0x6C, 0xC6, 0xC6, 0xC6, 0x6C, 0x38, 0x00 },   // O
{ 0xFC, 0x66, 0x66, 0x7C, 0x60, 0x60, 0xF0, 0x00 },   // P
{ 0x78, 0xCC, 0xCC, 0xCC, 0xDC, 0x78, 0x1C, 0x00 },   // Q
{ 0xFC, 0x66, 0x66, 0x7C, 0x6C, 0x66, 0xE6, 0x00 },   // R
{ 0x78, 0xCC, 0xE0, 0x70, 0x1C, 0xCC, 0x78, 0x00 },   // S
{ 0xFC, 0xB4, 0x30, 0x30, 0x30, 0x30, 0x78, 0x00 },   // T
{ 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xFC, 0x00 },   // U
{ 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0x78, 0x30, 0x00 },   // V
{ 0xC6, 0xC6, 0xC6, 0xD6, 0xFE, 0xEE, 0xC6, 0x00 },   // W
{ 0xC6, 0xC6, 0x6C, 0x38, 0x38, 0x6C, 0xC6, 0x00 },   // X
{ 0xCC, 0xCC, 0xCC, 0x78, 0x30, 0x30, 0x78, 0x00 },   // Y
{ 0xFE, 0xC6, 0x8C, 0x18, 0x32, 0x66, 0xFE, 0x00 },   // Z
{ 0x78, 0x60, 0x60, 0x60, 0x60, 0x60, 0x78, 0x00 },   // [
{ 0xC0, 0x60, 0x30, 0x18, 0x0C, 0x06, 0x02, 0x00 },   // (backslash)
{ 0x78, 0x18, 0x18, 0x18, 0x18, 0x18, 0x78, 0x00 },   // ]
{ 0x10, 0x38, 0x6C, 0xC6, 0x00, 0x00, 0x00, 0x00 },   // ^
{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF },   // _
{ 0x30, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00 },   // ’
{ 0x00, 0x00, 0x78, 0x0C, 0x7C, 0xCC, 0x76, 0x00 },   // a
{ 0xE0, 0x60, 0x60, 0x7C, 0x66, 0x66, 0xDC, 0x00 },   // b
{ 0x00, 0x00, 0x78, 0xCC, 0xC0, 0xCC, 0x78, 0x00 },   // c
{ 0x1C, 0x0C, 0x0C, 0x7C, 0xCC, 0xCC, 0x76, 0x00 },   // d
{ 0x00, 0x00, 0x78, 0xCC, 0xFC, 0xC0, 0x78, 0x00 },   // e
{ 0x38, 0x6C, 0x60, 0xF0, 0x60, 0x60, 0xF0, 0x00 },   // f
{ 0x00, 0x00, 0x76, 0xCC, 0xCC, 0x7C, 0x0C, 0xF8 },   // g
{ 0xE0, 0x60, 0x6C, 0x76, 0x66, 0x66, 0xE6, 0x00 },   // h
{ 0x30, 0x00, 0x70, 0x30, 0x30, 0x30, 0x78, 0x00 },   // i
{ 0x0C, 0x00, 0x0C, 0x0C, 0x0C, 0xCC, 0xCC, 0x78 },   // j
{ 0xE0, 0x60, 0x66, 0x6C, 0x78, 0x6C, 0xE6, 0x00 },   // k
{ 0x70, 0x30, 0x30, 0x30, 0x30, 0x30, 0x78, 0x00 },   // l
{ 0x00, 0x00, 0xCC, 0xFE, 0xFE, 0xD6, 0xC6, 0x00 },   // m
{ 0x00, 0x00, 0xF8, 0xCC, 0xCC, 0xCC, 0xCC, 0x00 },   // n
{ 0x00, 0x00, 0x78, 0xCC, 0xCC, 0xCC, 0x78, 0x00 },   // o
{ 0x00, 0x00, 0xDC, 0x66, 0x66, 0x7C, 0x60, 0xF0 },   // p
{ 0x00, 0x00, 0x76, 0xCC, 0xCC, 0x7C, 0x0C, 0x1E },   // q
{ 0x00, 0x00, 0xDC, 0x76, 0x66, 0x60, 0xF0, 0x00 },   // r
{ 0x00, 0x00, 0x7C, 0xC0, 0x78, 0x0C, 0xF8, 0x00 },   // s
{ 0x10, 0x30, 0x7C, 0x30, 0x30, 0x34, 0x18, 0x00 },   // t
{ 0x00, 0x00, 0xCC, 0xCC, 0xCC, 0xCC, 0x76, 0x00 },   // u
{ 0x00, 0x00, 0xCC, 0xCC, 0xCC, 0x78, 0x30, 0x00 },   // v
{ 0x00, 0x00, 0xC6, 0xD6, 0xFE, 0xFE, 0x6C, 0x00 },   // w
{ 0x00, 0x00, 0xC6, 0x6C, 0x38, 0x6C, 0xC6, 0x00 },   // x
{ 0x00, 0x00, 0xCC, 0xCC, 0xCC, 0x7C, 0x0C, 0xF8 },   // y
```

```
     { 0x00, 0x00, 0xFC, 0x98, 0x30, 0x64, 0xFC, 0x00 },   // z
     { 0x1C, 0x30, 0x30, 0xE0, 0x30, 0x30, 0x1C, 0x00 },   // {
     { 0x18, 0x18, 0x18, 0x00, 0x18, 0x18, 0x18, 0x00 },   // |
     { 0xE0, 0x30, 0x30, 0x1C, 0x30, 0x30, 0xE0, 0x00 },   // }
     { 0x76, 0xDC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 },   // ~
     { 0x00, 0x10, 0x38, 0x6C, 0xC6, 0xC6, 0xFE, 0x00 },   // 127
     { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } };   // block char


pVideoFirstColumn = pVideoStart;
pVideo = pVideoFirstColumn;

//
// Select Memory Bank by reading or writing to port.
//
if (bank == 1)
   outp(PanelPortAddr+2, 0);
else
   inp(PanelPortAddr+2);

switch (PanelGrayLevel)
   {
   case 2:
      //
      // If there are 2 gray levels, there are 8 pixels/byte
      //
      color &= 0x01;

      while (*str != 0)
         {
         ch = *str++;

         if ((ch < ' ') || (ch - ' ' > MAX_FONT-1))
            ch = '.';

         pFont = &font[ch - ' '][0];

         for (y = 0; y < 8; ++y)
            {
            pVideo = pVideoFirstColumn;
            Video = 0;

            val = *pFont++;

            //
            // Since there are 2 gray shades, each bit in the font will be
            // represented in video memory as a one bit pixel.
            //
            if (val & 0x80)
               Video |= (color << 7);

            if (val & 0x40)
               Video |= (color << 6);

            if (val & 0x20)
               Video |= (color << 5);

            if (val & 0x10)
               Video |= (color << 4);

            if (val & 0x08)
               Video |= (color << 3);

            if (val & 0x04)
```

```
            Video |= (color << 2);

         if (val & 0x02)
            Video |= (color << 1);

         if (val & 0x01)
            Video |= color;

         *pVideo++ = (unsigned char) Video;
         CheckBank(pVideo, &bank);

         pVideoFirstColumn += BytesPerScanLine;
         }

      ++pVideoStart;   // Point to next character
      pVideoFirstColumn = pVideoStart;
      }
   break;

case 4:
   //
   // If there are 4 colors/gray levels, there are 4 pixels/byte
   //
   color &= 0x03;

   while (*str != 0)
      {
      ch = *str++;

      if ((ch < ' ') || (ch - ' ' > MAX_FONT-1))
         ch = '.';

      pFont = &font[ch - ' '][0];

      for (y = 0; y < 8; ++y)
         {
         pVideo = pVideoFirstColumn;
         Video = 0;

         val = *pFont++;

         //
         // Since there are 4 colors/gray shades, each bit in the font
         // will be represented in video memory as a two bit pixel.
         //
         if (val & 0x80)
            Video |= (color << 6);

         if (val & 0x40)
            Video |= (color << 4);

         if (val & 0x20)
            Video |= (color << 2);

         if (val & 0x10)
            Video |= color;

         *pVideo++ = (unsigned char) Video;
         CheckBank(pVideo, &bank);
         Video = 0;

         if (val & 0x08)
            Video |= (color << 6);
```

```
            if (val & 0x04)
               Video |= (color << 4);

            if (val & 0x02)
               Video |= (color << 2);

            if (val & 0x01)
               Video |= color;

            *pVideo++ = (unsigned char) Video;
            CheckBank(pVideo, &bank);

            pVideoFirstColumn += BytesPerScanLine;
            }

         pVideoStart += 2;   // Point to next character
         pVideoFirstColumn = pVideoStart;
         }
      break;

   case 16:
      color &= 0x0f;

      while (*str != 0)
         {
         ch = *str++;

         if ((ch < ' ') || (ch - ' ' > MAX_FONT-1))
            ch = '.';

         pFont = &font[ch - ' '][0];

         for (y = 0; y < 8; ++y)
            {
            pVideo = pVideoFirstColumn;
            Video = 0;

            val = *pFont++;

            //
            // Since there are 16 colors/gray shades, each bit in the font
            // will be represented in video memory as a four bit pixel.
            //
            if (val & 0x80)
               Video |= (color << 4);

            if (val & 0x40)
               Video |= color;

            *pVideo++ = (unsigned char) Video;
            CheckBank(pVideo, &bank);
            Video = 0;

            if (val & 0x20)
               Video |= (color << 4);

            if (val & 0x10)
               Video |= color;

            *pVideo++ = (unsigned char) Video;
            CheckBank(pVideo, &bank);
            Video = 0;

            if (val & 0x08)
```

```
            Video |= (color << 4);

         if (val & 0x04)
            Video |= color;

         *pVideo++ = (unsigned char) Video;
         CheckBank(pVideo, &bank);
         Video = 0;

         if (val & 0x02)
            Video |= (color << 4);

         if (val & 0x01)
            Video |= color;

         *pVideo++ = (unsigned char) Video;
         CheckBank(pVideo, &bank);

         pVideoFirstColumn += BytesPerScanLine;
         }

      pVideoStart += 4;  // Point to next character
      pVideoFirstColumn = pVideoStart;
      }
   break;

case 256:
   while (*str != 0)
      {
      ch = *str++;

      if ((ch < ' ') || (ch - ' ' > MAX_FONT-1))
         ch = '.';

      pFont = &font[ch - ' '][0];

      for (y = 0; y < 8; ++y)
         {
         pVideo = pVideoFirstColumn;
         Video = 0;

         val = *pFont++;

         //
         // Since there are 256 colors, each bit in the font will be
         // represented in video memory as an 8 bit pixel.
         //
         for (count = 0; count < 8; ++count)
            {
            if (val & 0x80)
               Video = color;
            else
               Video = 0;

            *pVideo++ = (unsigned char) Video;
            CheckBank(pVideo, &bank);
            val <<= 1;
            }

         pVideoFirstColumn += BytesPerScanLine;
         }

      pVideoStart += 8;  // Point to next character
      pVideoFirstColumn = pVideoStart;
```

```
         }
      break;
   }
}

//-------------------------------------------------------------------------
//
// FUNCTION: SplitScreen()
//
// DESCRIPTION: Show split screen.
//
// INPUTS: None.
// RETURN VALUE: None.
//
//-------------------------------------------------------------------------

void SplitScreen(void)
{
static unsigned char _far *pVideoImage1;
static unsigned char _far *pVideoImage2;
static unsigned long ImageSize;
static unsigned int OriginalLineCount;
static unsigned int val;
static int MinLineCount;
static unsigned int MaxVirtualScanLines;
static unsigned char Image2Bank;

printf("Showing Split Screen\n");

Initialize();

SetDisplay(OFF);
ClearLCDScreen();


//
// Access memory banks
//
FP_SEG(pVideoImage1) = 0xd000;
FP_OFF(pVideoImage1) = 0x0000;


switch (PanelGrayLevel)
   {
   case 2:
      BytesPerScanLine = (PanelX / 8);
      break;

   case 4:
      BytesPerScanLine = (PanelX / 4);
      break;

   case 16:
      BytesPerScanLine = (PanelX / 2);
      break;

   case 256:
      BytesPerScanLine = PanelX;
      break;
   }

ShowVerticalBars(pVideoImage1, 0);
```

```
//
// Calculate starting video memory location for image 2 by finding the
// last location of image 1
//
ImageSize = (unsigned long) BytesPerScanLine * PanelY;

//
// Because the image size is limited to a maximum of 320 x 240, and there
// is 128K of video memory, there is enough memory available.
//
FP_SEG(pVideoImage2) = 0xd000;
FP_OFF(pVideoImage2) = (unsigned int) (ImageSize & 0xffff);

if (ImageSize & 0xffff0000)
   Image2Bank = BANK1;
else
   Image2Bank = BANK0;

ShowHorizontalBars(pVideoImage2, Image2Bank);

//
// Show text.  The lightest color/gray shade is set to PanelGrayLevel-1.
//
ShowText(pVideoImage1, BANK0, "SPLIT SCREEN IMAGE ONE", PanelGrayLevel-1);
ShowText(pVideoImage2, Image2Bank, "SPLIT SCREEN IMAGE TWO", PanelGrayLevel-1);


//
// Set Screen 2 Display Start Address register to point to Image 2
//
// Adjust ImageSize to represent the size in words, not bytes.
// This is because the Memory Interface is set to 16 bits.
//
val = (unsigned int) (ImageSize / 2);

WriteRegister(8, (unsigned int) val & 0xff);
WriteRegister(9, (unsigned int) val >> 8);


SetDisplay(ON);


//
// If this is a dual panel, then the split screen has just been shown.
// Otherwise, set up the Screen 1 Display Line Count register for single
// panels.
//
if (PanelType == TYPE_SINGLE)
   {
   OriginalLineCount =
      (unsigned int) ((ReadRegister(0x0b) & 0x03) << 8) | ReadRegister(0x0a);


   // Only for 128K of memory
   MaxVirtualScanLines = (unsigned int)
      ((unsigned long) 0x20000 / BytesPerScanLine);

   MinLineCount = OriginalLineCount -
                  (MaxVirtualScanLines - OriginalLineCount) + 1;

   if (MinLineCount < 0)
      MinLineCount = 0;
```

```
   //
   // Scroll image 2 down
   //
   for (val = MinLineCount; val < OriginalLineCount; val += 1)
      {
      WriteRegister(0x0a, val & 0xff);         // Total Display Line Count
      WriteRegister(0x0b, (val >> 8) & 0x03);  // Total Disp Line Cnt/WF Count

      Delay(DELAY_SHORT);
      }

   //
   // Scroll image 2 up
   //
   for (val = OriginalLineCount; val > (unsigned int) MinLineCount; val -= 1)
      {
      WriteRegister(0x0a, val & 0xff);         // Total Display Line Count
      WriteRegister(0x0b, (val >> 8) & 0x03);  // Total Disp Line Cnt/WF Count

      Delay(DELAY_SHORT);
      }

   val = MinLineCount;
   WriteRegister(0x0a, val & 0xff);            // Total Display Line Count Reg
   WriteRegister(0x0b, (val >> 8) & 0x03);     // Total Disp Line Cnt/WF Count

   Delay(500);
   }

ShowMenu();
}
```

---

```
void SetStartAddress(int x, int y)
{
int addr;

switch (PanelGrayLevel)
   {
   case 16:
      addr = (unsigned int) ((x/2 + (VIRTUAL_X/2) * y)/2);
      break;

   case 256:
      addr = (unsigned int) ((x + VIRTUAL_X * y)/2);
      break;
   }

WriteRegister(6, addr & 0xff);
WriteRegister(7, addr >> 8);
}
```

---

```
void PanScroll(void)
{
static unsigned int x, y;
static unsigned int MaxX, MaxY;
static unsigned int val, pitch;
static unsigned char _far *pVideo;
static unsigned char bank, color;


printf("Showing Panning and Scrolling\n");
```

```
Initialize();

SetDisplay(OFF);
ClearLCDScreen();


switch (PanelGrayLevel)
   {
   case 16:
      pitch = (unsigned int) (((VIRTUAL_X / 2) - BytesPerScanLine) / 2);
      BytesPerScanLine = (VIRTUAL_X / 2);
      break;

   case 256:
      pitch = (unsigned int) ((VIRTUAL_X - BytesPerScanLine) / 2);
      BytesPerScanLine = VIRTUAL_X;
      break;
   }

WriteRegister(0x0d, pitch);


//
// Access memory banks
//
FP_SEG(pVideo) = 0xd000;
FP_OFF(pVideo) = 0x0000;


//
// Display random blocks of data.  To do so, a text character will be used.
// This character sets all pixels in a character region, so a block is
// shown at the specified gray shade.
//

// Seed the random number generator with current time
srand((unsigned) time(NULL));

for (x = 0; x < 300; ++x)
   {
   if (((rand() * 2L) / RAND_MAX) == 1)
      bank = BANK0;
   else
      bank = BANK1;

   FP_OFF(pVideo) = (unsigned int) ((rand() * 0xffffL) / RAND_MAX);
   val = rand() % 50;

   switch (PanelGrayLevel)
      {
      case 16:
         color = (unsigned char) (rand() % 16);
         break;

      case 256:
         color = (unsigned char) (rand() % 256);
         break;
      }

   // The last character in the font table is a solid block character.
   ShowText(pVideo, bank, "\x80", color);
   }
```

```
ShowBorders();

//
// Move virtual display from (0, 0) to (MaxX, 0)
//
MaxX = (unsigned int) (VIRTUAL_X - PanelX);
MaxY = (unsigned int) (VIRTUAL_Y - PanelY);

SetDisplay(ON);

for (x = 0; x <= MaxX; ++x)
   {
   SetStartAddress(x, 0);
   Delay(DELAY_SHORT);
   }

for (y = 0; y <= MaxY; ++y)
   {
   SetStartAddress(MaxX, y);
   Delay(DELAY_SHORT);
   }

for (x = MaxX; x > 0; --x)
   {
   SetStartAddress(x, MaxY);
   Delay(DELAY_SHORT);
   }

for (y = MaxY; y > 0; --y)
   {
   SetStartAddress(0, y);
   Delay(DELAY_SHORT);
   }

SetStartAddress(0, 0);

ShowMenu();
}

//-------------------------------------------------------------------------
//
// FUNCTION: PowerSaving()
//
// DESCRIPTION: Starts power saving mode 2.
//
// INPUTS: None.
// RETURN VALUE: None.
//
//-------------------------------------------------------------------------

void PowerSaving(void)
{
static unsigned int val;

printf("Starting Power Saving\n");

//
// The following are the steps to enter a power save mode.
//

//
// Step 1: Turn off display
//
val = ReadRegister(1);
```

```
val &= 0x7f;
WriteRegister(1, val);

//
// Step 2: Disable LCDE (turn off LCD power supply).
//         For the S5U13503P00C, set LCDE bit to 0.
//
val = ReadRegister(1);
val &= 0xef;
WriteRegister(1, val);

//
// Step 2: Wait for LCD power supply to drop to zero volts
//         For the S5U13503P00C, wait about a half second.
//
Delay(500);

//
// Step 3: Enter Power Save Mode
//
val = ReadRegister(3);
val &= 0x3f;
val |= 0x80;
WriteRegister(3, val);  // Set power saving mode 2

printf("Press any key to cancel power saving\n");
getch();


//
// The following are the steps to exit a power save mode.
//

//
// Step 1: Exit Power Save Mode
//
val = ReadRegister(3);
val &= 0x3f;
WriteRegister(3, val);  // Cancel power saving mode 2

//
// Step 2: Enable LCDE (turn on LCD power supply).
//         For the S5U13503P00C, set LCDE bit to 1.
//
val = ReadRegister(1);
val |= 0x10;
WriteRegister(1, val);

//
// Step 3: Turn on display.
//
val = ReadRegister(1);
val |= 0x80;
WriteRegister(1, val);


ShowMenu();
}
```

# 8 GLOSSARY

| | |
|---|---|
| **13503** | The S1D13503 LCD controller chip. |
| **bank** | In reference to display memory banking, a bank is a 64K byte block of display memory. Bank 0 represents the first 64K bytes of display memory, and bank 1 represents the second 64K bytes. |
| **color** | A specific combination of red, green, and blue intensities. |
| **display memory** | Memory in which an image is stored for display by the S1D13503. |
| **gray shade** | A specific combination of white and black colors. For example, a lighter gray shade has more white than black. |
| **LCD** | Liquid Crystal Display. The display device used by the S1D13503. |
| **LCD controller** | The device used to control the LCD display. The S1D13503 is an LCD controller. |
| **LUT** | Look-Up Table, or palette. The LUT treats the value of a pixel as an index into an array of colors or gray shades. |
| **panel** | The circuitry and viewable area of an LCD display which supports a single image. LCD displays may have one or two panels. |
| **panning** | The right or left movement of the viewport in a virtual display. |
| **pixel** | Picture Element. A pixel is seen as a dot on the display, and can be shown using one of several different colors or gray shades. Combining pixels in a group creates an image. |
| **power saving** | A means of reducing the power consumption of the S1D13503. |
| **register** | A memory storage location to control a peripheral, such as the S1D13503. |
| **scrolling** | The up and down movement of the viewport in a virtual display. |
| **S1D13503** | The 13503 chip. |
| **S5U13503P00C** | The evaluation board for the S1D13503F00A. The S5U13503P00C is an ISA board for a PC-compatible computer. |
| **viewport** | The visible portion of a virtual display. |
| **virtual display** | An image, which is stored in display memory, that is larger than what the LCD display can show. A virtual display supports panning and scrolling. |

# S1D13503 REGISTER SUMMARY

| AUX[01] Mode Register 0: I/O address = 0001b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| DISP | PANEL | Mask XSCL | LCDE | Gray Shade / Color | LCD Data Width Bit 0 | Memory Interface | RAMS |

| AUX[02] Line Byte Count Register (LSB): I/O address = 0010b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Line Byte Count (low byte) | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

| AUX[03] Mode Register 1: I/O address = 0011b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Power Save Mode | | LCD Signal State | LUT Bypass | LCD Data Width Bit 1 | BW / 256 Colors | Color Mode | Line Byte Count Bit 8 |
| Bit 1 | Bit 0 | | | | | | |

| AUX[04] Total Display Line Count Register (LSB): I/O address = 0100b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Total Display Line Count (low byte) | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

| AUX[05] Total Display Line Count Register (MSB) and WF Count Register: I/O address = 0101b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| WF Count | | | | | | Total Display Line Count | |
| Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Bit 9 | Bit 8 |

| AUX[06] Screen 1 Display Start Address Register (LSB): I/O address = 0110b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Screen 1 Display Start Address (low byte) | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

| AUX[07] Screen 1 Display Start Address Register (MSB): I/O address = 0111b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Screen 1 Display Start Address (high byte) | | | | | | | |
| Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |

| AUX[08] Screen 2 Display Start Address Register (LSB): I/O address = 1000b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Screen 2 Display Start Address (low byte) | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

| AUX[09] Screen 2 Display Start Address Register (MSB): I/O address = 1001b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Screen 2 Display Start Address (high byte) | | | | | | | |
| Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |

| AUX[0A] Screen 1 Display Line Count Register (LSB): I/O address = 1010b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Screen 1 Display Line Count (low byte) | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

| AUX[0B] Screen 1 Display Line Count Register (MSB): I/O address = 1011b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| n/a[1] | n/a | n/a | n/a | n/a | n/a | Screen 1 Disp Line Count | |
| | | | | | | Bit 9 | Bit 8 |

| AUX[0C] Horizontal Non-Display Period Register: I/O address = 1100b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Horizontal Non-Display Period | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

| AUX[0D] Address Pitch Adjustment Register: I/O address = 1101b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Address Pitch Adjustment | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

| AUX[0E] Look-Up Table Address Register: I/O address = 1110b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Green Bank Select | | ID[2] / RGB Index | | Look-Up Table (Palette) Address | | | |
| Bit 1 | Bit 0 | Bit 1 | Bit 0 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

| AUX[0F] Look-Up Table Data Register: I/O address = 1111b, RW | | | | | | | |
|---|---|---|---|---|---|---|---|
| Red Bank Select | | Blue Bank Select | | Look-Up Table (Palette) Data | | | |
| Bit 1 | Bit 0 | Bit 1 | Bit 0 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

**Notes:** 1. n/a bits should be written 0.

2. These bits are used to identify the S1D13503 at power on / RESET. If these bits read 00b at Power On / Reset the device is an S1D13503. If this bit reads 10b at Power On / Reset the device is an SED1352F$_{0B}$/F$_{1B}$/D$_{0B}$. If this bit reads 11b at Power On / Reset the device is an SED1352F$_{0A}$/F$_{1A}$/D$_{0A}$.

# S1D13503 Series
# Dot Matrix Graphic LCD Controller

## Utilities

# UTILITIES

## Table of Contents

# 1 13503SHOW.EXE DISPLAY UTILITY

13503SHOW is a utility used to load and display GIF images. It can also be used to demonstrate the split screen capabilities of the S1D13503 by loading two images and vertically scrolling one image.

## Program Requirements

| | |
|---|---|
| **Video Controller** | : S1D13503 |
| **Display Type** | : Up to 640 × 480 LCD |
| **BIOS** | : Seiko Epson 13503BIOS version 1.xx or later |
| **DOS Program** | : Yes |
| **DOS Version** | : 3.0 or greater |
| **Windows Program** | : No |
| **Windows DOS Box** | : Yes |
| **Windows DOS Full Screen** | : Yes |
| **OS/2 DOS Full Screen** | : Yes |

## Installation

Copy the file **13503show.exe** from the distribution disk to a directory on your hard drive that is in your DOS path.

## Usage

13503SHOW is invoked from the DOS command line as follows.

**13503show [file1.gif] [file2.gif] [/i] [/k] [/v] [/?]**

Where:

| | |
|---|---|
| file1.gif | is the first screen image to be displayed. |
| file2.gif | is the second screen image to be displayed. |
| /i | inverts all displayed images (show as negative) - used for some monochrome panels (works in monochrome mode only). |
| /k | exit the program and keep the image on the display - useful in batch file execution such as demonstrations. |
| /v | verbose mode - useful to determine GIF information if it is not known. |
| /d | leave the display on while loading image - useful for animation. |
| /? | produces the usage message. |

*Examples:*

13503show

with no arguments will run the program in split screen mode. This will display two predefined images, with screen one displaying horizontal bars and screen two displaying vertical bars. Screen two may be scrolled up and down using the arrow, page up, page down, home and end keys.

13503show file.gif

displays the named GIF image.

13503show file1.gif file2.gif

displays the two named GIF images in a split screen. Screen two may be scrolled up and down using the arrow, page up, page down, home and end keys.

Pressing the ESC key will terminate the program.

## *Comments*

- 13503SHOW requires 13503BIOS.COM to be loaded prior to running.
- Split screen viewing is limited on dual panels. The view port is fixed in place at the top left of the bottom LCD panel. Panning and scrolling is still possible within the screen 2 view port.
- The size of screen two is determined by available memory and number of colors/gray shades. If there is insufficient memory for screen two 13503SHOW will not accept the two image files and will generate an error message.
- When loading two GIF images, it may take several seconds of apparent inactivity to load the second image into memory.
- The GIF format must be 2, 16 or 256 color, non-interlaced GIF89a format.
- 13503SHOW will clear the screen when the ESC key is pressed unless the /k switch is used in the command line.
- The file is loaded into the program at its image color depth (i.e., a 256 color image is initially displayed in 256 color mode, a 16 color image is initially displayed in 16 color mode).

## *Program Messages*

**ERROR: This program requires BIOS13503 to be loaded!**
The program 13503BIOS.COM must be run before 13503SHOW. Load 13503BIOS.COM and rerun 13503SHOW.EXE.

**File "*filename*" not found or cannot be opened for reading.**
The GIF file you are trying to display is not in your DOS path or not on your system.

**File is not GIF89a format.**
The GIF file contains an invalid format. 13503SHOW only supports GIF89a format.

**Insufficient video memory for second image.**
There is not enough video memory available to store both images.

# 2 13503VIRT.EXE DISPLAY UTILITY

13503VIRT.EXE demonstrates the virtual panning capabilities of the S1D13503. Two images larger than the display resolution are loaded in display memory. 13503VIRT.EXE will then display, in a split screen, a portion of each complete image while providing panning capabilities using the arrow keys for navigation.

## *Program Requirements*

| | |
|---|---|
| **Video Controller** | : S1D13503 |
| **Display Type** | : Up to 640 × 480 LCD |
| **BIOS** | : Seiko Epson 13503BIOS version 1.xx or later |
| **DOS Program** | : Yes |
| **DOS Version** | : 3.0 or greater |
| **Windows Program** | : No |
| **Windows DOS Box** | : Yes |
| **Windows DOS Full Screen** | : Yes |
| **OS/2 DOS Full Screen** | : Yes |

## *Installation*

Copy the file **13503virt.exe** from the distribution disk to a directory on your hard drive that is in your DOS path.

## *Usage*

13503VIRT is invoked from the DOS command line as follows.

**13503virt g=n [/a] [/k] [/?]**

Where: g is the number of gray shades/colors: **2, 4, 16** or **256**.
- /a automatically pan and scroll the image - useful for demonstrations.
- /k exit the program and keep the image on the display - useful in batch file execution for demonstrations.
- /? produces a usage message.

The program draws a test pattern of two images on the display. The user can navigate throughout either image using the numeric keypad. Use the arrow keys to pan and scroll the screen, Home to go to the top left, PG UP to go to the top right, End to go to the bottom left, Pg Dn to go to the bottom right, and 5 to go to the center of the image. Pressing Ctrl while using an arrow key steps the scroll or pan in smaller increments. Press the Num Lock key to allow navigation in the second image.

Holding down the Shift key while pressing either the up or down arrow will move the split up or down.

Pressing the ESC key terminates the program.

## *Comment*

13503VIRT requires 13503BIOS.COM to be loaded prior to running.

## *Program Message*

**ERROR: This program requires 13503BIOS to be loaded!**
The program 13503BIOS.COM must be run before 13503VIRT.EXE. Load 13503BIOS.COM and then re-run 13503VIRT.EXE.

# 3 13503BIOS.COM UTILITY

13503BIOS is a Terminate and Stay Resident (TSR) program which replaces and/or supplements the PC video interrupt INT 10h. This program provides text, scroll, and cursor functionality when no VGA BIOS is present. Although the S1D13503 is not a VGA or EGA compatible controller, this program is supplied to give the user a familiar prompt. Within limits 13503BIOS simulates a VGA BIOS and will allow standard output functions to work. DOS programs such as Edlin, Format, Debug, and internal commands such as Copy, Ren, Mkdir, etc., should work. However, complex programs such as Edit, Qbasic, and Scandisk will not work. The standard output functions are handled by the VGA BIOS, if present.

## *Program Requirements*

| | |
|---|---|
| **Video Controller** | : S1D13503 |
| **Display Type** | : Up to 640 × 480 LCD |
| **BIOS** | : None or any VGA |
| **DOS Program** | : Yes |
| **DOS Version** | : 3.0 or greater |
| **Windows Program** | : No |
| **Windows DOS Box** | : Yes |
| **Windows DOS Full Screen** | : Yes |
| **OS/2 DOS Full Screen** | : Yes |

## *Installation*

Copy the files **13503bios.com** and **13503bios.ini** from the distribution disk to a directory on your hard drive that is in your DOS path.

## *Usage*

13503BIOS.COM is run from the DOS command line.

The file 13503bios.ini is the initialization file for 13503bios.com and must reside in the same directory as 13503bios.com. This file contains the default run parameters for 13503bios.com. These parameters may be changed within the initialization file or for one time usage on the command line as follows:

**13503bios t=n x=n y=n d=n g=n i=n m=n p=n [f=n] [/?]**

Where:
- t is panel drive type: **single** or **dual**.
- x is the horizontal panel size in pixels (decimal).
- y is the vertical panel size in lines (decimal).
- d is panel type: **color** or **mono**.
- g is the number of colors/gray shades: **2**, **4**, **16** or **256**.
- i is the panel interface data width: **4**, **8** or **16** bits.
- p is the port address in hex: **300**|**310**...**360**|**370**.
- m is the memory size in K bytes: **64** or **128**.
- f is the 8-bit color panel format: **1** or **2**.
- /? produces a usage message.

Note that the port address must be the same as the physical address set on the S5U13503P00C evaluation board.

*Example:*

```
13503BIOS t=single x=320 y=240 d=color g=256 i=8 p=310 m=128 f=2
```

## *Comments*

- 13503BIOS can be used in conjunction with a Monochrome Display Adapter (mono) card. The standard DOS command MODE MONO will switch to the monochrome card and the DOS command MODE CO80 will switch to the LCD panel.
- 13503BIOS emulates mode 3, but any program that attempts to write directly to video memory, bypassing the video BIOS, will not display correctly.
- 13503BIOS can be used in conjunction with a VGA BIOS. In this case all TTY output will be displayed on the VGA monitor.
- When the S1D13503 video memory is specified as 64K bytes, the S1D13503 video memory will reside at D000h to DFFFh. For 128K bytes of S1D13503 video memory, the memory will reside at C000h to DFFFh.

## *Program Messages*

**ERROR: Panels greater than 640 pixels not supported.**
More than 640 horizontal pixels has been specified for the panel in the command line.

**ERROR: Panels greater than 480 lines not supported.**
More than 480 vertical lines has been specified for the panel in the command line.

**ERROR: Invalid port specified.**
The port address (p) must be specified in the format 3x0 in the command line. The range is 300h to 370h in 10h increments.

**ERROR: Not enough memory for panel.**
The panel specified is too large to run in 16 gray shades mode. Select 4 gray shades instead.

**ERROR: Video memory and VGA BIOS memory conflict.**
Both the S1D13503 video memory and the VGA BIOS are trying to use the memory at location C000h to CFFFh.

**ERROR: Only 64k or 128k memory allowed.**
An invalid value has been specified for memory size (m) on the command line.

# 4 13503MODE.EXE DISPLAY UTILITY

13503MODE is a menu driven display utility for the S1D13503 which demonstrates the color /gray shades as well as available palettes. For 128K bytes of display memory either 4, 16 or 256 colors/ gray shades are available.

## *Program Requirements*

| | |
|---|---|
| **Video Controller** | : S1D13503 |
| **Display Type** | : Up to 640 × 480 LCD |
| **BIOS** | : Seiko Epson 13503BIOS version 1.xx or later |
| **DOS Program** | : Yes |
| **DOS Version** | : 3.0 or greater |
| **Windows Program** | : No |
| **Windows DOS Box** | : Yes |
| **Windows DOS Full Screen** | : Yes |
| **OS/2 DOS Full Screen** | : Yes |

## *Installation*

Copy the file **13503mode.exe** from the distribution disk to a directory on your hard drive that is in your DOS path.

## *Usage*

13503MODE is invoked from the DOS command line as follows.

**13503mode g=n [/?] [/d] [/k]**

Where: g is the number of colors /gray shades: **2, 4, 16** or **256**.

/? produces a usage message.

/d inhibits display writes on startup - useful for examaning the LUT of a previously loaded image.

/k exit the program and keep the image on the display - useful for batch file execution for demonstrations.

13503MODE displays a default color/gray shade pattern as a series of vertical or horizontal bars. The pattern, number of colors/gray shades and current palette may be modified by the user when possible. Instructions to modify these options appear when available.

An image other than the default one may be used as follows:

1. run **13503bios.com** if it is not already loaded
2. load an image into the video buffer with 13503show.exe
   **13503show file.gif /k**
3. load **13503mode /d**

The Look-Up Table (LUT) of the image file displayed may now be manipulated by the user.

Pressing the ESC key terminates the program and restores the original 13503BIOS settings.

## *Comment*

13503MODE requires 13503BIOS.COM to be loaded prior to running.

## *Program Message*

**ERROR: This program requires 13503BIOS to be loaded!**
The program 13503BIOS.COM must be run before 13503MODE. Load 13503BIOS.COM and then re-run 13503MODE.EXE.

# 5 13503PD.EXE POWER DOWN UTILITY

13503PD is an OEM utility program for setting power down modes in the S1D13503 LCD Display Controller that supports the SOLLEX Super VGA Standard video BIOS extensions. It provides a simple method for setting power modes during power consumption testing.

## *Program Requirements*

| | |
|---|---|
| **Video Controller** | : S1D13503 |
| **Display Type** | : Up to 640 × 480 LCD |
| **BIOS** | : Seiko Epson 13503BIOS version 1.xx or later |
| **DOS Program** | : Yes |
| **DOS Version** | : 3.0 or greater |
| **Windows Program** | : No |
| **Windows DOS Box** | : Yes |
| **Windows DOS Full Screen** | : Yes |
| **OS/2 DOS Full Screen** | : Yes |

## *Installation*

Copy the file **13503pd.exe** from the distribution disk to a directory on your hard drive that is in your DOS path.

## *Usage*

13503PD is run from the DOS command line as follows:

**13503pd ModeNumber**

Where: ModeNumber is a decimal number (0, 1, or 2) for the desired power down mode.

*Example*: typing the following command line activates power down mode 2:

```
13503pd 2 <ENTER>
```

Output from the program can be redirected to an external DOS device such as a terminal attached to the serial port such as COM1 as shown below:

```
13503pd 2 > com1 <ENTER>
```

Striking any key will set mode state 0 (no power down).

## *Comments*

- 13503PD.EXE requires 13503BIOS.COM to be loaded prior to running.
- The following power modes are supported:

| | |
|---|---|
| Mode 0 | Mode 0 operates at full power. |
| Mode 1 or 2 | S1D13503 will engage power down mode 1 or 2. The S1D13503 Look-Up Table will be disabled and all LCD signals are forced low. |

## *Program Messages*

**Power Down Mode xx is set.**
The power down mode xx has been set. This message may not be visible if the active display controller is the S1D13503.

**ERROR: Cannot set power mode xx!**
13503PD.EXE cannot set the power down mode requested - either 13503BIOS.COM is not loaded or the power down mode number exceeds 2.

**ERROR: This program requires 13503BIOS to be loaded!**
The program 13503BIOS.COM must be run before 13503PD. Load 13503BIOS and re-run 13503PD.EXE.

# 6 13503READ.EXE DIAGNOSTIC UTILITY

13503READ is an OEM utility program which enables the user to read the S1D13503 register contents. It is a useful utility for OEMs wishing to submit a problem report for the video controller. If run with 13503BIOS loaded, it will try to interpret the BIOS settings.

## *Program Requirements*

| | |
|---|---|
| **Video Controller** | : S1D13503 |
| **Display Type** | : Up to 640 × 480 LCD |
| **BIOS** | : Seiko Epson 13503BIOS.COM (optional) |
| **DOS Program** | : Yes |
| **DOS Version** | : 3.0 or greater |
| **Windows Program** | : No |
| **Windows DOS Box** | : Yes |
| **Windows DOS Full Screen** | : Yes |
| **OS/2 DOS Full Screen** | : Yes |

**Note:** 13503READ uses “stdout” calls and may be redirected to a file or piped to a DOS filter such as MORE.COM.

## *Installation*

Copy the file **13503read.exe** from the installation diskette to a directory on your hard drive which is in the DOS path.

## *Usage*

From DOS prompt, type the following:

**13503read [p=n] [/?]**

Where: 13503read without any argument will read the S1D13503 registers, including the palettes.

| | |
|---|---|
| p | is the S1D13503 port address in hex (e.g. 310). |
| /? | produces a usage message. |

*Example:* to generate a report, simply type

13503read [port] > report.txt

and the information which 13503READ obtains will be stored in the file report.txt.

## *Comments*

- It is not necessary to specify a port address if 13503BIOS has previously been loaded.
- 13503READ will search for 13503BIOS.COM. If this program is found the port address reported by 13503BIOS will be used. If the port address is specified on the 13503READ command line the two port addresses are compared and if different an error message is generated.
- 13503READ will accept any port address, however, the S5U13503P00C can only be configured to an address in the range of 300h through 370h.

## *Program Messages*

**ERROR: 13503 registers not responding at port address [port].**
13503READ has not found an S1D13503 at the port address specified. Check the command line port setting for 13503BIOS and/or 13503READ to ensure it is correct and re-run the program.

**ERROR: 13503READ requires a port address.**
13503READ has not detected 13503BIOS.COM to obtain the port address and no port address was specified on the command line. Either specify a port address on the 13503READ command line or run 13503BIOS.COM prior to running 13503READ.

**ERROR: 13503BIOS reports a port address of [port], which is different from the specified port address of [port].**
The port address entered for 13503READ is different than the one entered for 13503BIOS.COM. Specify the same port address on the 13503READ command line as the one in 13503BIOS.COM and the physical address of the S5U13503P00C evaluation board and re-run the program.

**WARNING: 13503BIOS state is out of sync with S1D13503 registers.**
One or more of the following command line items reported by 13503BIOS does not match the values found in the S1D13503 registers; horizontal panel size, vertical panel size, number of colors/gray shades, or panel type (single or dual).

# S1D13503 Series
## Dot Matrix Graphic LCD Controller

## S5U13503P00C
## Evaluation Board
## User's Manual

# S5U13503P00C EVALUATION BOARD USER'S MANUAL

## Table of Contents

## List of Figures



## List of Tables

# 1 S5U13503P00C Rev 1.0 Evaluation Board

This manual reflects the use of the S5U13503P00C Rev 1.0 evaluation board in conjunction with the S1D13503F00A LCD Controller. All appropriate components are surface-mount to reduce cost and minimize board space.

## 1.1 Features

- 100 pin QFP5 package
- SMD technology for all appropriate devices
- 4/8-bit Monochrome STN LCD display support
- 4/8/16-bit Color STN LCD display support
- 8/16-bit ISA Bus support
- 5V operation
- Two terminal crystal support (up to 25.0MHz)
- Oscillator support
- 16-bit wide, 128K bytes SRAM support
- Configuration Options
- Support for Software Power Save modes
- On-board adjustable LCD BIAS negative power supply
- On-board adjustable LCD BIAS positive power supply
- CPU/Bus Interface Header strips for Non-ISA Bus support

# 2 INSTALLATION AND CONFIGURATION

The S1D13503F00A uses the display memory data lines (VD[15:0]) as configuration inputs which are read on power-up. For the purpose of this design, most of these configuration inputs have been factory set and therefore are not configurable. An eight position DIP switch is provided for the selection of the following:

Table 2-1 Configuration DIP Switch Settings

| Switch | Signal | Closed | Open |
|---|---|---|---|
| SW1-1 | VD0 | 16-bit ISA Bus interface | 8-bit ISA Bus interface |
| SW1-2 | VD1 | Direct-mapping I/O | Indexed I/O |
| SW1-3 | VD2 | M68K CPU interface | ISA Bus interface |
| SW1-4 | VD3 | Byte-swap high and low data bytes | No byte-swap |
| SW1-5 | VD7 | I/O mapping address bit 4 | See Table 2-2, "I/O Mapping Example" |
| SW1-6 | VD8 | I/O mapping address bit 5 | |
| SW1-7 | VD9 | I/O mapping address bit 6 | |
| SW1-8 | – | Reserved | Reserved |

**Notes:** • The polarity of the Configuration Dip Switches is Closed = "1" or "high", Open = "0" or "low".

• VD[15:0] have internal pull-down resistors and therefore external pull-up resistors are only required if the configuration option requires a "1" state on power-up.

### Factory set fixed options on this board are:

- 16-bit display memory interface.
- All 128K bytes of video memory is available at memory segment $D with software selecting one of two 64K memory banks (See *"SRAM Support"* on page 7).
- This board is pre-set to use indexed I/O with address $03y0 (0000 0011 0yyy 000x), where x is don't care and yyy can be configured with dip-switch SW1-5 through SW1-7. The factory setting of yyy = 001, i.e., I/O address = $0310 and $0311.

Direct-mapping I/O is only available for Non-ISA Bus support. When using direct-mapped I/O, the I/O address is $03yx (0000 0011 0yyy xxxx), where x is don't care and yyy can be configured with dip-switch SW1-5 through SW1-7. If yyy = 001, then the I/O address for AUX[00] = $0310, I/O address for AUX[01] = $0311, I/O address for AUX[02] = $0312 and so on. (See *"Non-ISA Bus Support"* on page 7.)

Table 2-2 I/O Mapping Example

| | Bit 6 | Bit 5 | Bit 4 |
|---|---|---|---|
| I/O Mapping Address (Hex) | 0 | 0 | 1 |

Table 2-3 Decoding Jumper Setting

| | Description | 1-2 | 2-3 |
|---|---|---|---|
| JP1 | Set to the same polarity as SW1-1 (VD0) | 1 | 0 |
| JP2 | Set to the same polarity as SW1-5 (VD7) | 1 | 0 |
| JP3 | Set to the same polarity as SW1-6 (VD8) | 1 | 0 |
| JP4 | Set to the same polarity as SW1-7 (VD9) | 1 | 0 |
| JP5 | XSCL2 clock for Passive Color 8-bit single 640 × 480 LCD Panel (Format 1) (see Functional Specification) | NC | XSCL2 |

**Note:** These jumpers are necessary for the external ISA Bus decode logic.

### Hard-Wired Configuration Inputs

For ISA bus support options, external 10K ohm pull-up resistors have been assembled, and are connected to signal lines VD11, VD12, VD14 and VD15 (R6, R5, R4 and R3 respectively).

For Non-ISA bus support (see page 7), the following signal lines may require the 10K ohm pull-up resistors installed:

VD4 (R18), VD5 (R19), VD6 (R20), VD10 (R21) and/or VD13 (R17)

## LCD Signal Connector Pinout

Table 2-4 LCD Signal Connector J1 Pinout

| S1D13503 Pin Name | LCD Connector Pin No. | Color STN LCD | | | | | Mono STN LCD | |
|---|---|---|---|---|---|---|---|---|
| | | 16-bit Single/Dual | 8-bit Dual | 8-bit Single (Format 1[#1]) AUX[03] bit 3 = 0, AUX[01] bit 2 = 1 | 8-bit Single (Format 2[#1]) AUX[03] bit 3 = 1, AUX[01] bit 2 = 1 | 4-bit | 8-bit | 4-bit |
| LD0 | 1 | LD0 | LD0 | LD0 | LD0 | | LD0 | |
| LD1 | 3 | LD1 | LD1 | LD1 | LD1 | | LD1 | |
| LD2 | 5 | LD2 | LD2 | LD2 | LD2 | | LD2 | |
| LD3 | 7 | LD3 | LD3 | LD3 | LD3 | | LD3 | |
| UD0 | 9 | UD0 | UD0 | UD0 | UD0 | UD0 | UD0 | UD0 |
| UD1 | 11 | UD1 | UD1 | UD1 | UD1 | UD1 | UD1 | UD1 |
| UD2 | 13 | UD2 | UD2 | UD2 | UD2 | UD2 | UD2 | UD2 |
| UD3 | 15 | UD3 | UD3 | UD3 | UD3 | UD3 | UD3 | UD3 |
| LD4 | 17 | LD4 | | | | | | |
| LD5 | 19 | LD5 | | | | | | |
| LD6 | 21 | LD6 | | | | | | |
| LD7 | 23 | LD7 | | | | | | |
| UD4 | 25 | UD4 | | | | | | |
| UD5 | 27 | UD5 | | | | | | |
| UD6 | 29 | UD6 | | | | | | |
| UD7 | 31 | UD7 | | | | | | |
| XSCL | 33 | XSCL | XSCL | XSCL | XSCL | XSCL | XSCL | XSCL |
| WF/XSCL2 | 35 | | | XSCL2 | | | | |
| LP | 37 | LP | LP | LP | LP | LP | LP | LP |
| YD | 39 | YD | YD | YD | YD | YD | YD | YD |
| GRND | 2–26 (even pins) | GRND | GRND | GRND | GRND | GRND | GRND | GRND |
| N/C | 28 | | | | | | | |
| VLCD | 30 | | | | | | VLCD | VLCD |
| VCC | 32 | +5V | +5V | +5V | +5V | +5V | +5V | +5V |
| +12V | 34 | +12V | +12V | +12V | +12V | +12V | +12V | +12V |
| VDDH | 36 | VDDH | VDDH | VDDH | VDDH | VDDH | VDDH | VDDH |
| WF/XSCL2 | 38 | WF | WF | | WF | WF | WF | WF |
| LCDENB | 40 | /LCDPWR | /LCDPWR | /LCDPWR | /LCDPWR | /LCDPWR | /LCDPWR | /LCDPWR |

#1 See Section 7.4 of the *S1D13503 Hardware Functional Specification* for details.

## CPU / BUS Interface Connector Pinouts

Table 2-5 CPU/BUS Connector H1 Pinout

| Connector Pin No. | CPU/BUS Pin Name | Comments |
|---|---|---|
| 1 | SD0 | Connected to DB0 of the S1D13503 |
| 2 | SD1 | Connected to DB1 of the S1D13503 |
| 3 | SD2 | Connected to DB2 of the S1D13503 |
| 4 | SD3 | Connected to DB3 of the S1D13503 |
| 5 | GND | Ground |
| 6 | GND | Ground |
| 7 | SD4 | Connected to DB4 of the S1D13503 |
| 8 | SD5 | Connected to DB5 of the S1D13503 |
| 9 | SD6 | Connected to DB6 of the S1D13503 |
| 10 | SD7 | Connected to DB7 of the S1D13503 |
| 11 | GND | Ground |
| 12 | GND | Ground |
| 13 | SD8 | Connected to DB8 of the S1D13503 |
| 14 | SD9 | Connected to DB9 of the S1D13503 |
| 15 | SD10 | Connected to DB10 of the S1D13503 |
| 16 | SD11 | Connected to DB11 of the S1D13503 |
| 17 | GND | Ground |
| 18 | GND | Ground |
| 19 | SD12 | Connected to DB12 of the S1D13503 |
| 20 | SD13 | Connected to DB13 of the S1D13503 |
| 21 | SD14 | Connected to DB14 of the S1D13503 |
| 22 | SD15 | Connected to DB15 of the S1D13503 |
| 23 | RESET | Connected to the RESET signal of the S1D13503 |
| 24 | GND | Ground |
| 25 | GND | Ground |
| 26 | GND | Ground |
| 27 | +12V | 12 volt supply |
| 28 | +12V | 12 volt supply |
| 29 | /SBHE | Connected to the BHE# signal of the S1D13503 |
| 30 | IOCHRDY | Connected to the READY signal of the S1D13503 |
| 31 | /IOSC | Connected to the IOCS# signal of the S1D13503 |
| 32 | /MEMCS | Connected to the MEMCS# signal of the S1D13503 |

Table 2-6 CPU/BUS Connector H2 Pinout

| Connector Pin No. | CPU/BUS Pin Name | Comments |
|---|---|---|
| 1 | SA0 | Connected to AB0 of the S1D13503 |
| 2 | SA1 | Connected to AB1 of the S1D13503 |
| 3 | SA2 | Connected to AB2 of the S1D13503 |
| 4 | SA3 | Connected to AB3 of the S1D13503 |
| 5 | SA4 | Connected to AB4 of the S1D13503 |
| 6 | SA5 | Connected to AB5 of the S1D13503 |
| 7 | SA6 | Connected to AB6 of the S1D13503 |
| 8 | SA7 | Connected to AB7 of the S1D13503 |
| 9 | GND | Ground |
| 10 | GND | Ground |
| 11 | SA8 | Connected to AB8 of the S1D13503 |
| 12 | SA9 | Connected to AB9 of the S1D13503 |
| 13 | SA10 | Connected to AB10 of the S1D13503 |
| 14 | SA11 | Connected to AB11 of the S1D13503 |
| 15 | SA12 | Connected to AB12 of the S1D13503 |
| 16 | SA13 | Connected to AB13 of the S1D13503 |
| 17 | GND | Ground |
| 18 | GND | Ground |
| 19 | SA14 | Connected to AB14 of the S1D13503 |
| 20 | SA15 | Connected to AB15 of the S1D13503 |
| 21 | SA16 | Connected to U2 pin 20 |
| 22 | SA17 | Connected to AB17 of the S1D13503 |
| 23 | SA18 | Connected to AB18 of the S1D13503 |
| 24 | SA19 | Connected to AB19 of the S1D13503 |
| 25 | GND | Ground |
| 26 | GND | Ground |
| 27 | +5V | 5 volt supply |
| 28 | +5V | 5 volt supply |
| 29 | /IOW | Connected to the IOW# signal of the S1D13503 |
| 30 | /IOR | Connected to the IOR# signal of the S1D13503 |
| 31 | /SMEMW | Connected to the MEMW# signal of the S1D13503 |
| 32 | /SMEMR | Connected to the MEMR# signal of the S1D13503 |

# 3 TECHNICAL DESCRIPTION

## 3.1 ISA Bus Support

This board directly supports the 8/16-bit ISA Bus with Indexed I/O via a standard AT edge connector. Only those configuration resistors needed for ISA Bus support have been assembled, refer to *"Hard-Wired Configuration Inputs"* on page 2 for configuration details. External logic has been added to provide signals which the S1D13503F00A does not directly support. See Application Note S18A-G-003-xx for details.

This board is pre-set to use indexed I/O with base address 000 0011 0yyy 000x, where x is don't care and yyy can be configured through dip-switch SW1-7 to SW1-5. The factory setting of yyy = 001, i.e., I/O address = $0310 and $0311. The display memory bank address is described in *"SRAM Support"* on page 7.

*Example:*

```
I/O write $310 01   :set index = 1
I/O read $311       :read contents of AUX[1]
I/O write $310 05   :set index = 5
I/O write $311 07   :write 07 to AUX[5]
```

This board has been designed to operate as a stand-alone card or in conjunction with either a VGA or a monochrome display adapter card.

### With VGA

When the VGA display adapter used is an ISA or VL bus with an 8-bit BIOS EPROM (normally just one ROM on the adapter card) the S5U13503P00C must be configured as follows:

| | |
|---|---|
| SW1-1 open | : 8-bit operation, necessary to prevent MEMCS16# conflict when reading VGA BIOS |
| SW1-2 to 4 open | : for ISA bus support with indexed I/O |
| SW1-5 to 7 | : set as desired |
| JP1 2-3 shorted | : to reflect SW1-1 polarity |
| JP2 to JP4 | : to reflect SW1-5 to 7 polarity |
| JP5 | : set as required for panel |

When the ISA or VL bus VGA video adapter has a 16-bit BIOS EPROM (normally two ROMs on the adapter card) either the 16-bit ISA bus interface or the 8-bit ISA bus interface may be used on the S5U13503P00C.

When using the S5U13503P00C in conjunction with a PCI bus VGA display adapter either the 16-bit ISA bus interface or the 8-bit ISA bus interface may be used on the S5U13503P00C.

### With Monochrome

When using the S5U13503P00C in conjunction with a monochrome display adapter either the 16-bit ISA bus interface or the 8-bit ISA bus interface may be used on the S5U13503P00C.

### Stand-Alone

The S5U13503P00C can be used as a stand-alone video adapter. When used as a stand-alone video adapter the BIOS setup program for the computer must support and have "No Video" selected as the video adapter. The 13503BIOS.COM utility program can be used with the evaluation board to simulate a standard video BIOS, thus providing text and cursor functionality. See the 13503BIOS.COM Utility manual, S18A-B-003-01 for details.

## 3.2 Non-ISA Bus Support

This evaluation board was specifically designed to support the standard 8/16-bit ISA bus. However, as the S1D13503F00A does support other bus interfaces, header strips have been provided containing all necessary I/O pins. (See Table 2-1, "Configuration DIP Switch Settings," on page 2, *"Hard-Wired Configuration Inputs"* on page 2, and *"CPU/Bus Interface Header Strips"* on page 9, for details.)

When using the header strips to provide the bus interface observe the following:

1. All I/O signals on the ISA bus card edge must be isolated from the ISA Bus (do not plug the card into a computer). Voltage lines are provided on the header strips.
2. U2, a TIBPAL22V10 PAL, is currently used to provide the S1D13503F00A IOCS# (pin 23) and MEMCS# (pin 22) input signals for ISA bus use. This functionality must now be provided externally as U2 must be removed.
3. Linear addressing of the entire 128K bytes of video RAM is available. Due to the memory banking method used for ISA bus support, U2 must be removed and H2, pin 21, must be physically connected to U2, pin20, in order to provide SA16 to U1.
4. If it becomes necessary / desirable to change the configuration information associated with VD[15:0], additional 10K Ohm pull-up resistors can be added to those affected VD lines as there are place holders available on the PCB.

## 3.3 SRAM Support

The S5U13503P00C board supports 16-bit wide, 128K byte SRAM. In order for the S5U13503P00C to operate in conjunction with a VGA card and not cause memory space conflicts, all 128K bytes of memory is available through two 64K byte banks. The first 64K bank is selected by *reading* from the base I/O mapping address + 2 (address $312 if the I/O address is $310) and the second 64K bank is selected by *writing* to I/O address + 2 (address $312 if the I/O address is $310). The display memory banks reside at the 64K byte memory segment $D.

```
I/O read $312    :select memory bank 0
I/O write $312   :select memory bank 1
```

## 3.4 Monochrome LCD Support

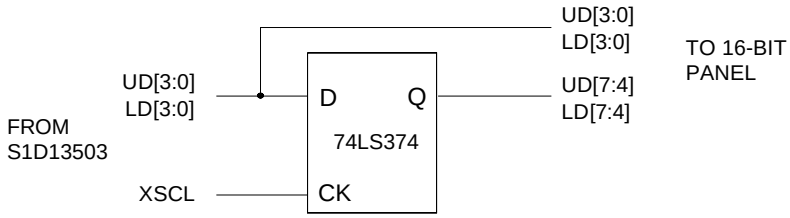
The S1D13503F00A directly supports 4/8-bit Dual and Single monochrome LCD panels. All the necessary signals are provided on the 40-pin ribbon cable header. The interface signals are alternated with grounds on the cable to reduce cross talk and noise related problems.

Refer to Table 2-4, "LCD Signal Connector J1 Pinout," on page 3 for specific settings.

## 3.5 Color LCD Support

The S5U13503P00C directly supports 4/8/16-bit Dual and Single color LCD panels. All the necessary signals are provided on the 40-pin ribbon cable header. The interface signals are alternated with grounds on the cable to reduce cross talk and noise related problems.

To facilitate interfacing a 16-bit panel to the S1D13503F00A, the following external circuit is implemented on-board:




This circuit provides 16-bit color panel support by latching the 8 bits of output data from the S1D13503F00A to provide 16 bits of data on the next clock. Refer to Table 2-4, “LCD Signal Connector J1 Pinout,” on page 3 for specific settings.

## 3.6 Power Save Modes

The S1D13503F00A supports two software Power Save Modes. The utility program 13503PD.EXE is supplied to control these software modes. The software modes are controlled by directly writing the S1D13503F00A associated internal registers.

## 3.7 Adjustable LCD Panel Negative Power Supply

The majority of Monochrome LCD panels require a negative power supply to provide between -18 V and -23 V ($I_{out}$ = 45 mA). For ease of implementation, such a power supply has been provided as an integral part of this design. The signal VLCD can be adjusted by R11 (100K potentiometer) to provide an output voltage from -14 V to -23 V and enabled/disabled by the control signal LCDENB.

**Note:** LCDENB is directly controlled by register AUX[01], bit 4, of the S1D13503F00A. The VLCD power supply used on the S5U13503P00C requires a logic “1” to disable it. As the signal LCDENB is a logic “0” at power-up, it is inverted by external logic to disable VLCD and prevent damaging the panel connected to the S5U13503P00C.

Determine the panel’s specific power requirements and set the potentiometer accordingly before connecting the panel.

## 3.8 Adjustable LCD Panel Positive Power Supply

The majority of LCD Passive Color panels and most single Monochrome 640 × 480 STN LCD panels require a positive power supply to provide between +23 V and +40 V ($I_{out}$ = 45 mA). For ease of implementation, such a power supply has been provided as an integral part of this design. The signal VDDH can be adjusted by R8 (100K potentiometer) to provide an output voltage from +23 V to +40 V and enabled/disabled by the control signal LCDENB.

**Note:** LCDENB is directly controlled by register AUX[01], bit 4, of the S1D13503F00A. The VDDH power supply used on the S5U13503P00C requires a logic “1” to disable it. As the signal LCDENB is a logic “0” at power-up, it is inverted by external logic to disable VDDH and prevent damaging the panel connected to the S5U13503P00C.

Determine the panel’s specific power requirements and set the potentiometer accordingly before connecting the panel.

## 3.9 Crystal Support

The input crystal frequency may be up to 25.0 MHz depending on the specific panel size and frame rate desired.

Refer to Section 9.3 of the *Hardware Functional Specification* for further details.

## 3.10 Oscillator Support

The input oscillator frequency used may be up to 25.0 MHz, depending on the specific panel size and frame rate desired.

Refer to Section 9.3 of the *Hardware Functional Specification* for further details.

**Note:** When the oscillator package is used capacitors C7, C8 and resistor R16 must be removed.

## 3.11 CPU/Bus Interface Header Strips

All of the CPU/Bus interface pins of S1D13503F00A, with the exception of SA16, are connected to the header strips H1 and H2 for easy interface to a CPU/Bus other than the ISA bus.

Refer to Table 2-5, "CPU/BUS Connector H1 Pinout," on page 4 and Table 2-6, "CPU/BUS Connector H2 Pinout," on page 5 for specific settings.

**Note:** These headers only provide the CPU/Bus interface signals from S1D13503F00A, when MC68000 interface is selected (SW1-3 closed), external decoding logic MUST be used to access the S1D13503F00A.

## 3.12 Schematic Notes

This evaluation board may have been modified and therefore the following schematics may not reflect the actual implementation. Please request updated information before starting any hardware design.

# *APPENDIX A PARTS LIST*

| Item # | Qty/Board | Designation | Part Value | Description |
|---|---|---|---|---|
| 1 | 13 | C11–C23 | 0.01µF | 0.01µF, 1206 pckg |
| 2 | 2 | C9 –C10 | 10µF | 10µF / 25V Tantalum D-SIZE |
| 3 | 2 | C7–C8 | 7pF | 7pF, 1206 pckg **(This Parts isn't assembled)** |
| 4 | 3 | C2–C4 | 10µF / 63V | Electrolytic / Radial (LXF63VB10RM5X11LL) |
| 5 | 3 | C1, C5, C6 | 56µF/35V | LXF35VB56RM6X11LL |
| 6 | 2 | H1, H2 | CON32A | 32-pin Dual Row Header |
| 7 | 5 | JP1–JP5 | Header 3 | 3-pin single Row Header |
| 8 | 1 | J1 | CON40A | Shrouded Header 40 pin Dual-row center-key PTH |
| 9 | 1 | L1 | 1µH | Dale Inductor IM-4-1.0µH PTH |
| 10 | 2 | L2–L3 | Ferrite Bead | Fair-rite 2743001111 |
| 11 | 1 | Q1 | 2N3906 | PNP Signal Transistor TO-92 PTH |
| 12 | 1 | Q2 | 2N3903 | NPN Signal Transistor TO-92 PTH |
| 13 | 3 | R2, R13, R14 | 1K | 1K Ohm/1206/5% |
| 14 | 2 | R12, R15 | 100K | 100K Ohm/1206/5% |
| 15 | 4 | R3–R6 | 10K | 10K Ohm/1206/5% |
| 16 | 1 | R7 | 470K | 470K Ohm/1206/5% |
| 17 | 1 | R8 | 200K | 200K Ohm Trim POT Spectrol 63S204T607 |
| 18 | 1 | R9 | 10K | 10K 10-pin SIP, Part No. 4610X-101-103 |
| 19 | 1 | R10 | 14K | 14K Ohm/1206/5% |
| 20 | 1 | R11 | 100K | 100K Ohm Trim POT Spectrol 63S104T607 |
| 21 | 1 | R16 | 2M | 2M Ohm/1206/5% **(This Parts isn't assembled)** |
| 22 | 1 | R1 | 0 Ohm | 0 Ohm/1206/1% |
| 23 | 5 | R17–R21 | 10K | 10K Ohm/1206/5% |
| 24 | 1 | S1 | SW-DIP-8 | DIP Switch 8-position |
| 25 | 1 | U1 | S1D13503F00A | QFP5-100-S2 |
| 26 | 1 | U2 | TIBPAL22V10 | Texas Ins. PAL, Socketed |
| 27 | 2 | U3, U4 | 74LS688 | DW020 SMT Package |
| 28 | 1 | U5 | 74LS09 | D014 SMT Package |
| 29 | 2 | U6, U7 | | SRM20100LTM-70 128K × 8 SRAM |
| 30 | 1 | U8 | RD-0412 | XENTEK - Positive Power Supply |
| 31 | 1 | U9 | EPN001 | XENTEK - Negative Power Supply |
| 32 | 1 | U10 | OSC-14 | 14-pin Socket for 25.0MHz, 12.0MHz, and 6.0MHz 14-pin Oscillators **(This is only Socket)** |
| 33 | 1 | U11 | 74LS374 | DW020 SMT Package |
| 34 | 1 | Y1 | 25.175MHz | Crystal **(This Parts isn't assembled)** |

# APPENDIX B S5U13503P00C REV. 1.0 SCHEMATIC DIAGRAMS

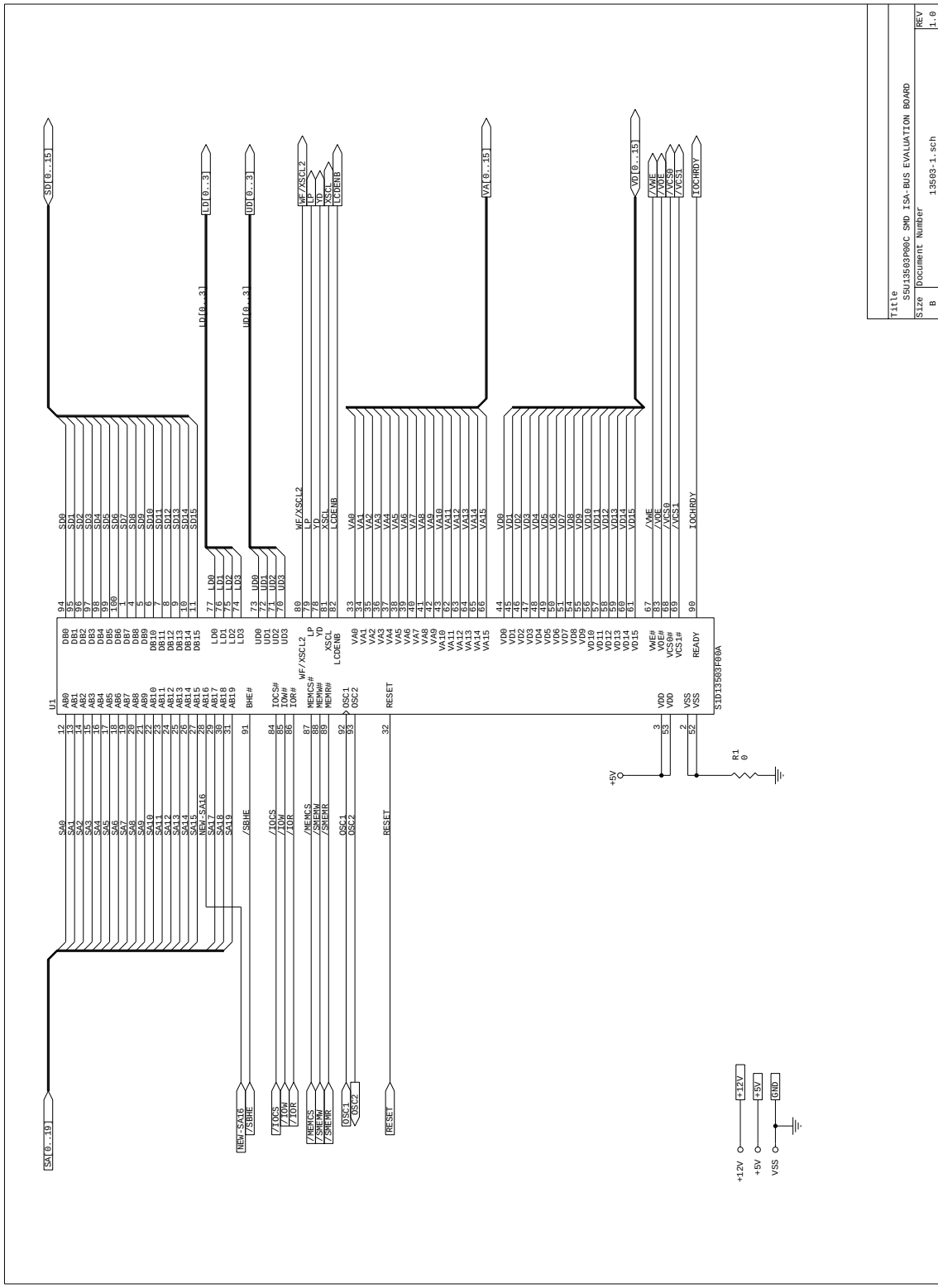



Figure 1 S5U13503P00C Rev. 1.0 Schematic Diagram (1 of 7)

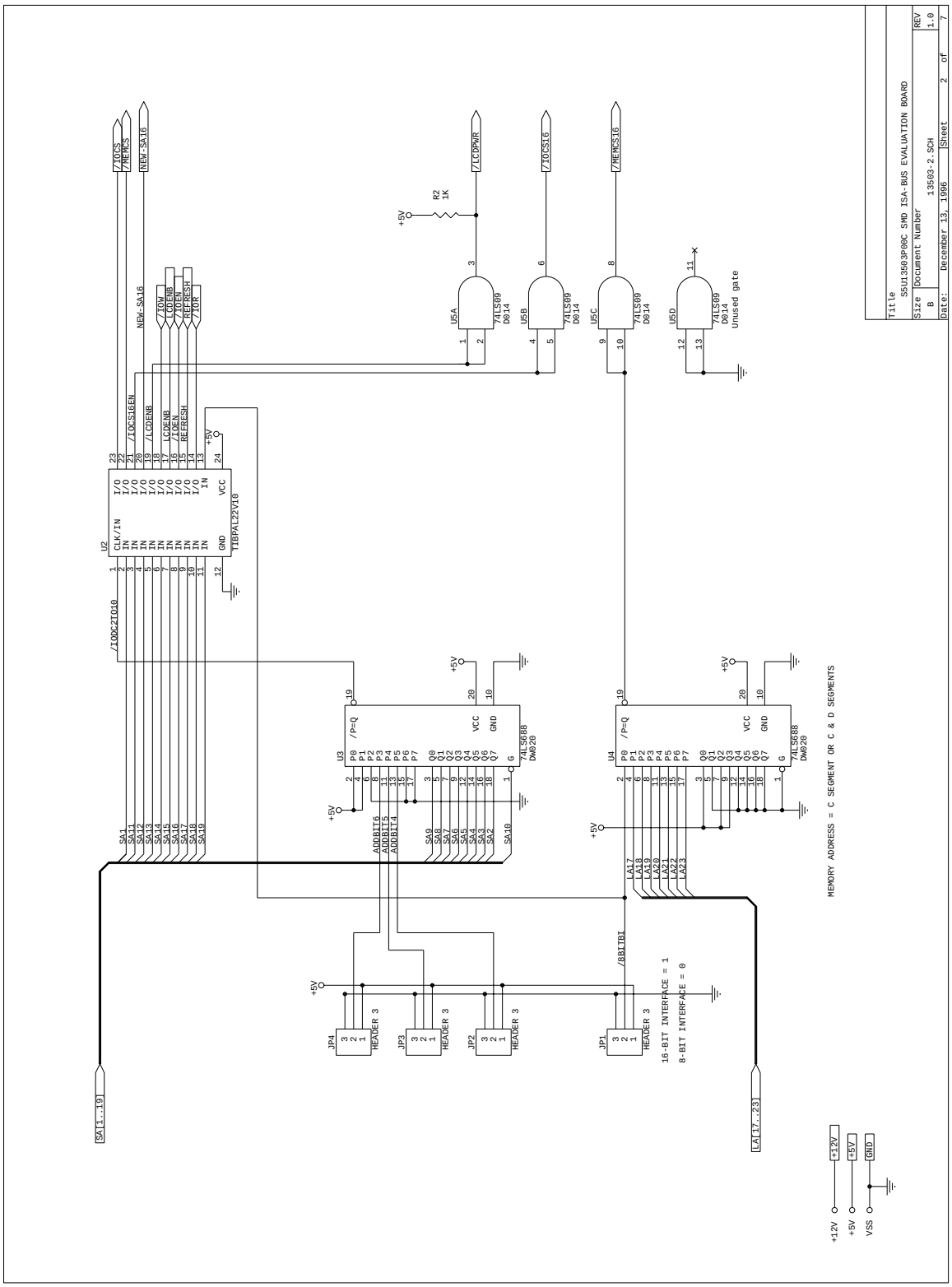



Figure 2 S5U13503P00C Rev. 1.0 Schematic Diagram (2 of 7)

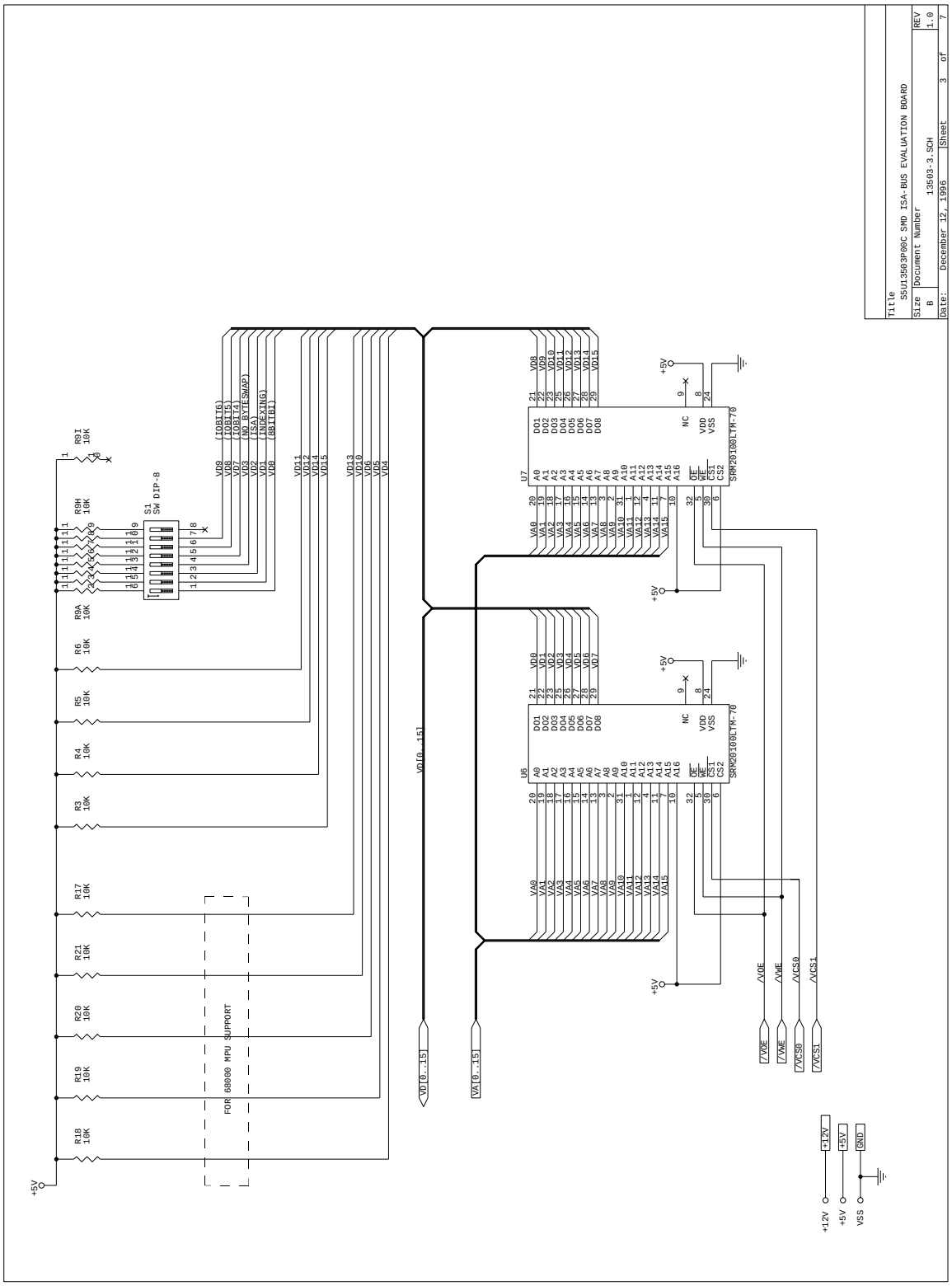



Figure 3 S5U13503P00C Rev. 1.0 Schematic Diagram (3 of 7)

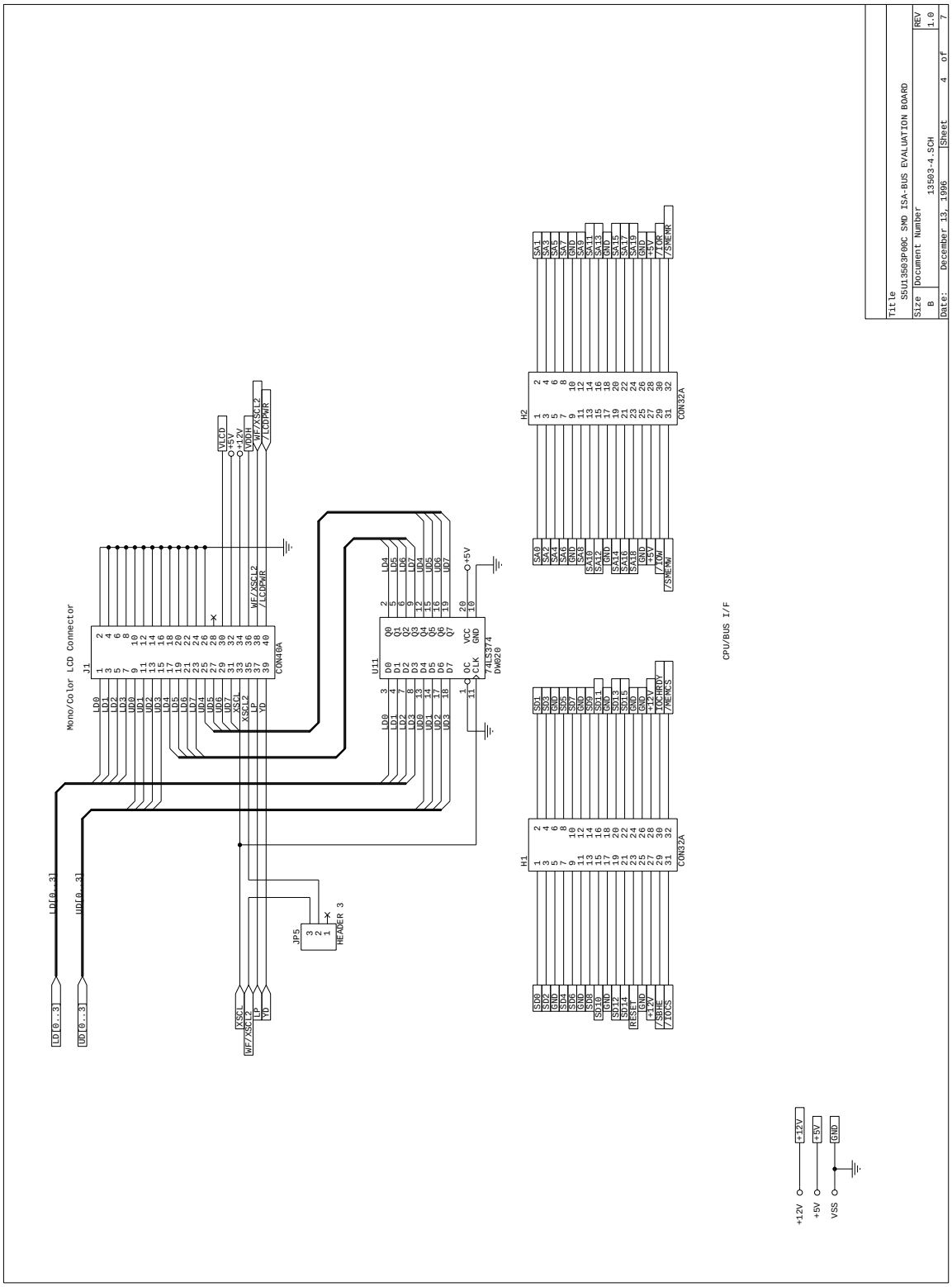



Figure 4 S5U13503P00C Rev. 1.0 Schematic Diagram (4 of 7)

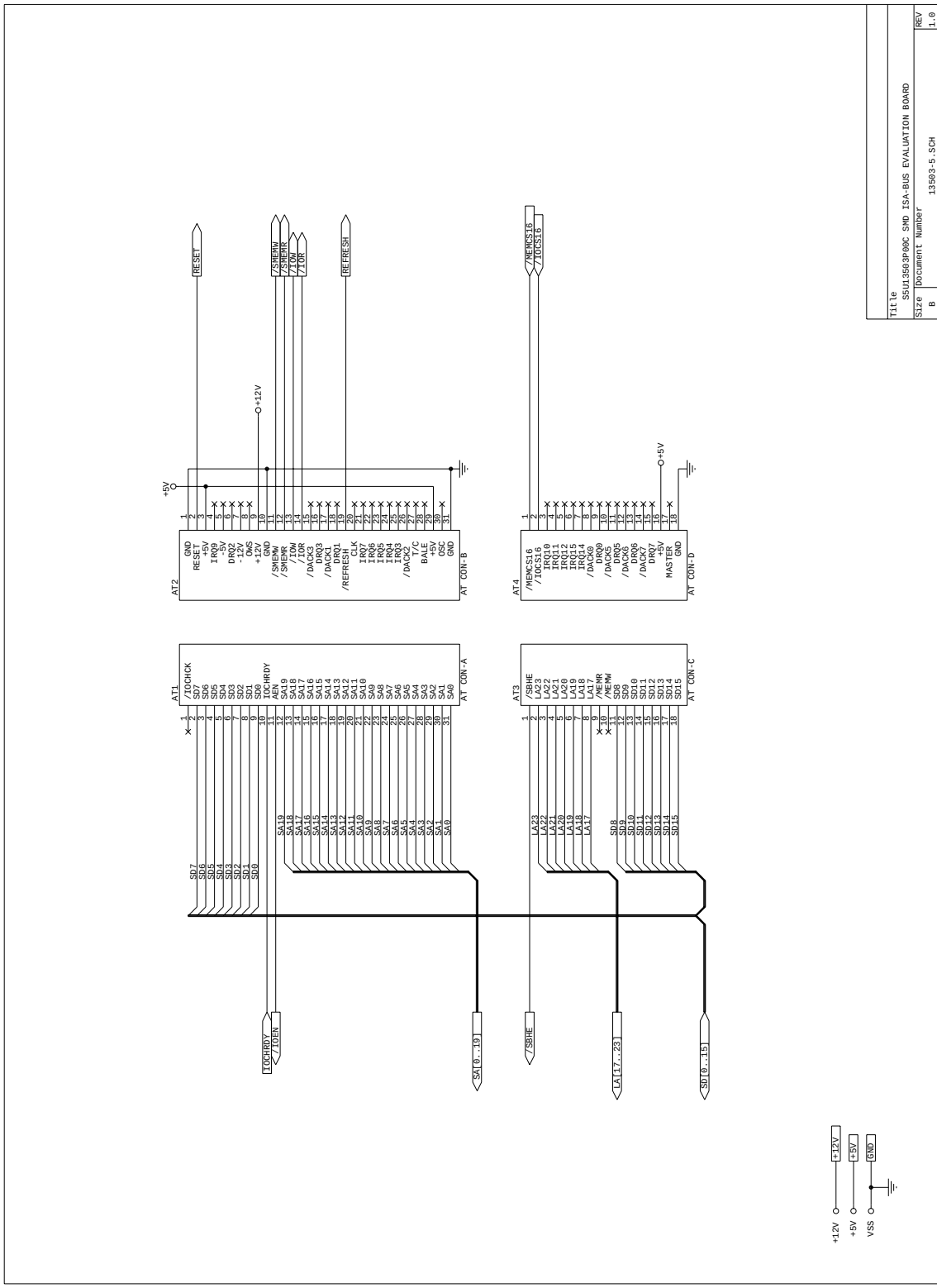



Figure 5 S5U13503P00C Rev. 1.0 Schematic Diagram (5 of 7)

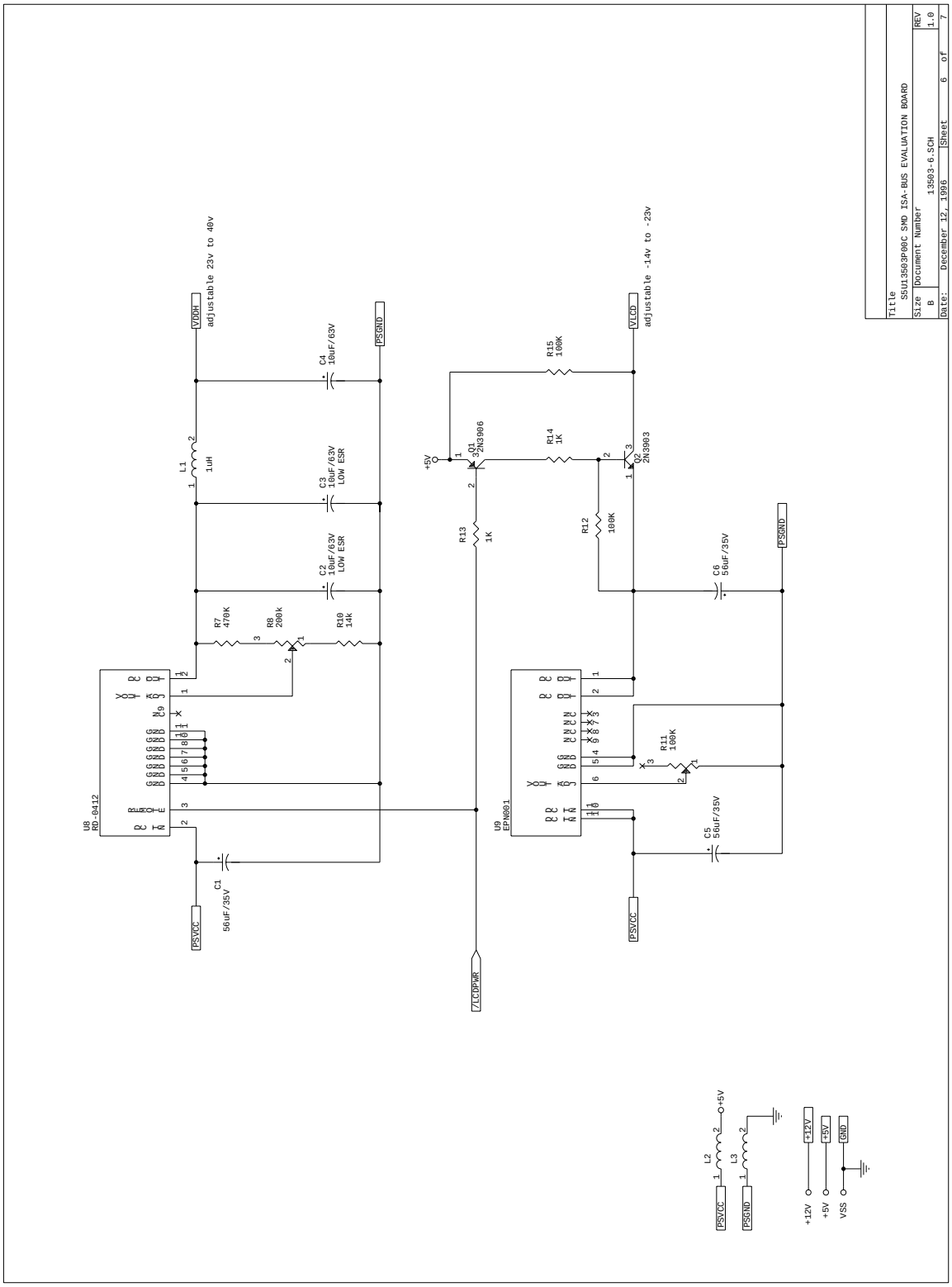



Figure 6 S5U13503P00C Rev. 1.0 Schematic Diagram (6 of 7)

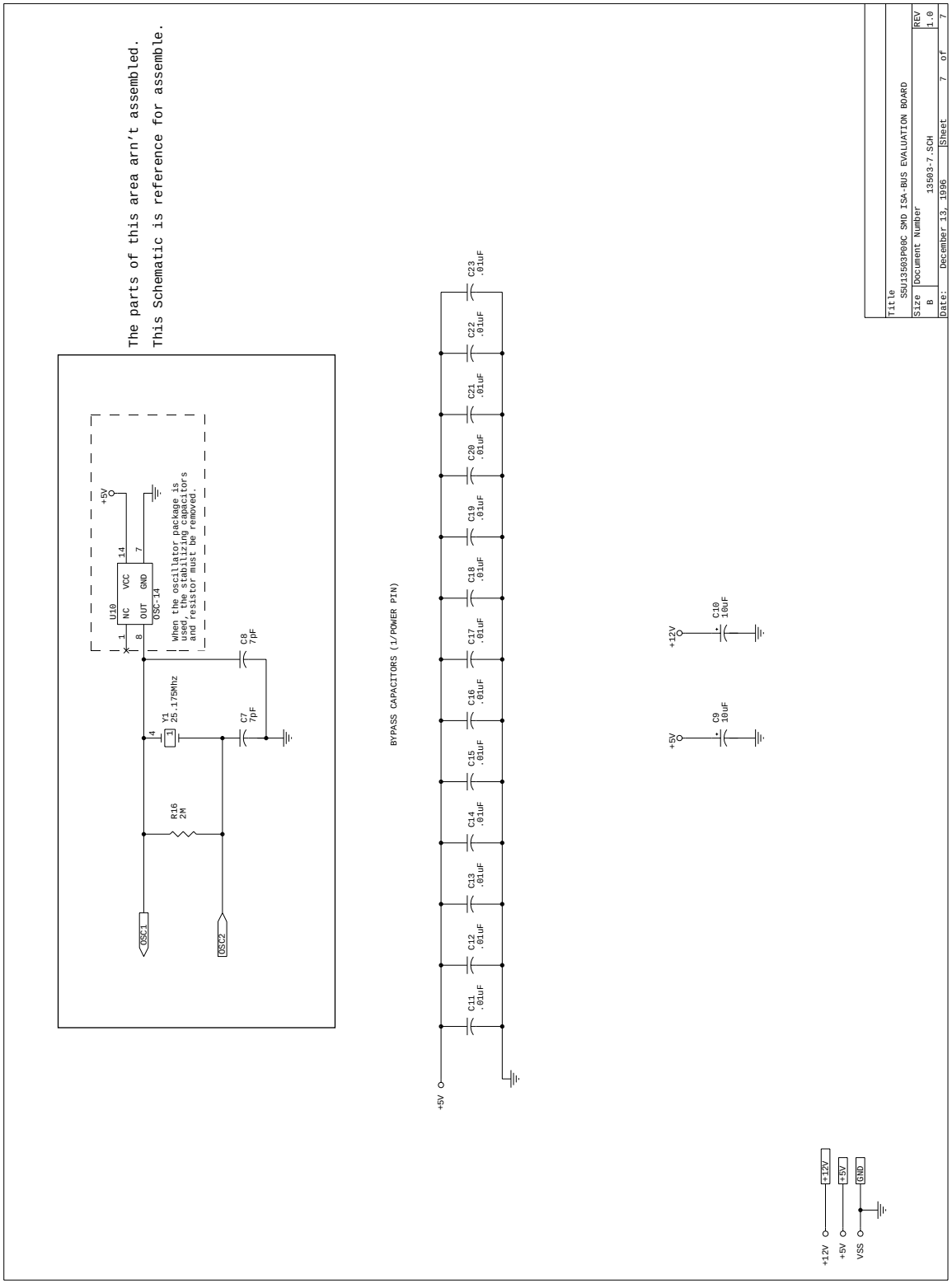



Figure 7 S5U13503P00C Rev. 1.0 Schematic Diagram (7 of 7)

**THIS PAGE IS BLANK.**

# S1D13503 Series
## Dot Matrix Graphic LCD Controller
## Application Notes

# APPLICATION NOTES

## Table of Contents

## List of Figures



## List of Tables

# 1 ISA BUS INTERFACE CONSIDERATIONS

## 1.1 Introduction

The S1D13503 is a general purpose LCD controller capable of interfacing to a variety of microprocessors. In some cases this interface is accomplished through the use of minimal external circuitry. This application note describes the interface between the S1D13503 and the ISA Bus both 8 and 16-bit implementations.

### Reference Material

Refer to the *S1D13503 Hardware Functional Specification* for complete AC timing details.

This document makes no attempts to describe the operation of the ISA Bus, please refer to the appropriate ISA Bus documentation for complete information.

## *1.2 16-Bit ISA Bus Interface*

For the purpose of the example shown below, the following conditions apply:

1. Indexed I/O with addresses 0310h and 0311h (see Configuration Options)
2. 128K bytes of display memory occupying $C and $D segments (see Configuration Options)

**Note:** This memory configuration will conflict with a VGA card installed on the same bus, therefore either a serial terminal or monochrome display adapter is recommended as the primary console.

This section provides the necessary logic equations and settings to complete the interface between the S1D13503 and the 16-bit ISA Bus.

**Note:** A PAL was used instead of discrete logic to reduce external component count.

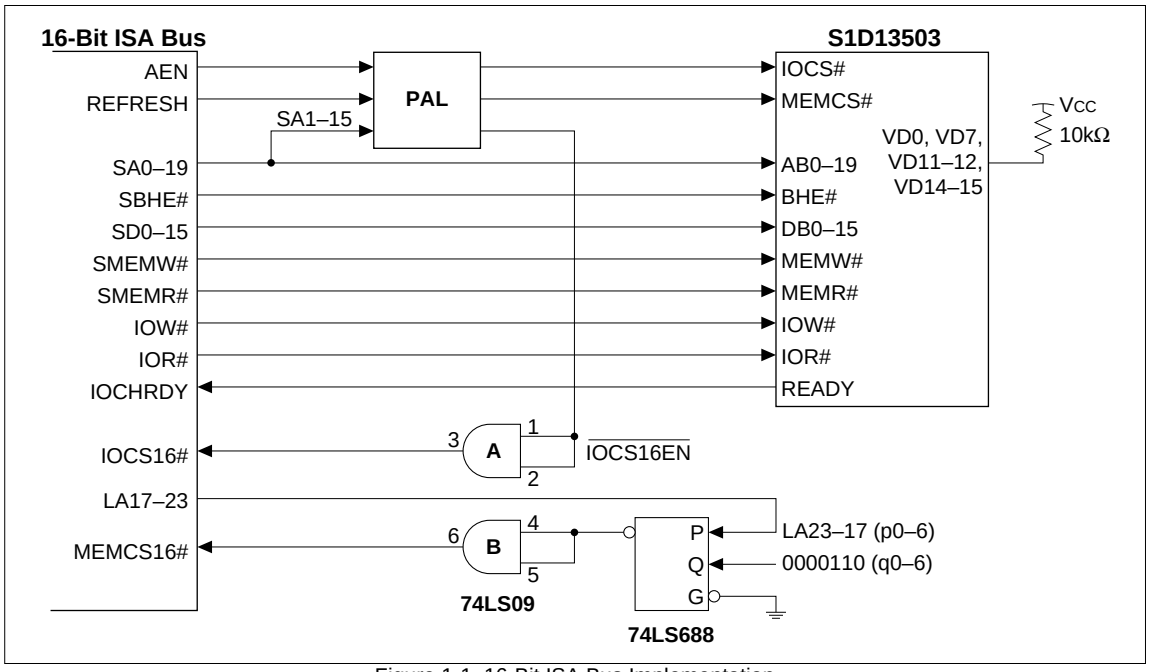



Figure 1-1 16-Bit ISA Bus Implementation

### *PAL Equations*

The PAL is programmed with the following equations:

A '!' placed before a signal name indicates a logic '0' state.
A '&' indicates a logic 'AND' function.

1. As stated above, the default I/O address is from 0310h to 0311h. The S1D13503 provides internal decoding of address bits A0 to A9, therefore minimal external circuitry is necessary to provide signals IOCS# and IOCS16#.

   IOCS# is required by the S1D13503 to indicate a valid I/O cycle. In an ISA bus environment, valid I/O decoding must include addresses A15 to A0. As A0 to A9 are decoded internally, the equation must only guarantee that addresses A10 to A15 must all be '0' and AEN must also be '0'.

   IOCS# = !(!AEN & !A15 & !A14 & !A13 & !A12 & !A11 & !A10)

2. As the S1D13503 is capable of 16-bit I/O access, the IOCS16# bus signal must be driven externally to indicate such a cycle. As stated in the ISA specification, the IOCS16# is a straight address decode without qualification.

   IOCS16EN# = !(!IOCS# & A9 & A8 & !A7 & !A6 & !A5 & A4 & !A3 & !A2 & !A1)

3. With 128K bytes of display memory and A17 to A19 decoded internally to S1D13503; MEMCS# = !REFRESH

**Note:** The MSBs of the address (A23:A20) need not be externally decoded if using SMEMW# and SMEMR# as they will only assert on addresses < 1MB.

## *Additional Discrete Logic Description*

1. As shown in Figure 1-1, the 74LS688 is configured as a memory decoder with valid addresses between 0C0000h and 0DFFFFh. This provides the MEMCS16# signal allowing for 16-bit memory cycles. As stated in the ISA specification, the MEMCS16# is a straight address decode without qualification.
2. The 74LS09 is used simply to provide the Open-Collector outputs necessary for the IOCS16# and MEMCS16# signals.

## *S1D13503 Default Setup*

### Configuration Options

The S1D13503 latches the state of the SRAM data bus during RESET to determine the power-on configuration. The chip has internal pull-down resistors and therefore external pull-ups are only necessary when requiring a '1' state, see below.

| | | |
|---|---|---|
| 1. | VD15–VD13 = 110 | memory decoding for locations $C and $D segments |
| 2. | VD12–VD4 = 110001000 | I/O decoding for locations 0310h and 0311h (1100010000b–1100010001b) |
| 3. | VD3 = 0 | No byte swap of high and low bytes |
| 4. | VD2 = 0 | ISA Bus interface, i.e. non- MC68K interface |
| 5. | VD1 = 0 | Indexed I/O |
| 6. | VD0 = 1 | 16-bit bus interface |

Where 1 = pull-up with a 10K resistor; 0 = no pull-up resistor

### Register Setting

All register settings are completely programmable with the following exceptions;

- Memory Interface, AUX[1] bit 1 = 0 for 16-bit memory interface.

**Note:** This bit is forced = 0 when 16-bit CPU Interface is selected through VD0 on power-up.

- RAMS, AUX[1] bit 0, this bit is ignored in 16-bit memory configurations.

All other registers are dependent on display type, resolution, color and mode of operation, see *S1D13503 Hardware Functional Specification* for details.

## *1.3 8-Bit ISA Bus Interface*

For the purpose of the example shown below, the following conditions apply:

1. Indexed I/O with partial decoding, i.e. address lines A10 to A15 are not decoded for I/O cycles

**Note:** Partial decoding is quite safe on most ISA Bus systems as I/O addresses above 03FFh are rarely used.

2. I/O addresses are 0300h and 0301h (xxxxxx1100000000b and xxxxxx1100000001b)
3. 64K bytes of display memory occupying $A segment

**Notes:** • The 74LS00 is simply used to detect the $B segment and invalidate the MEMCS# input.

• This memory configuration will conflict with a VGA card installed on the same bus, therefore either a serial terminal or monochrome display adapter is recommended as the primary console.

This section provides the necessary settings to complete the interface between the S1D13503 and the 8-bit ISA Bus. Since I/O addresses are partially decoded, there is no need to use a PAL for decoding.

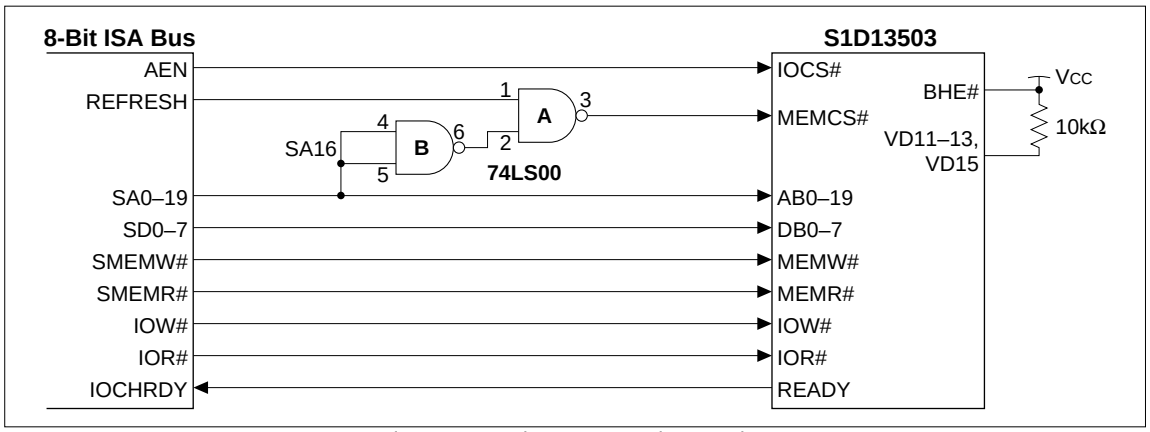



Figure 1-2 8-Bit ISA Bus Implementation

### *S1D13503 Default Setup*

#### Configuration Options

The S1D13503 latches the state of the SRAM data bus during RESET to determine the power-on configuration. The chip has internal pull-down resistors and therefore external pull-ups are only necessary when requiring a '1' state, see below.

| | | |
|---|---|---|
| 1. | VD15–VD13 = 101 | memory decoding for locations $A segment |
| 2. | VD12–VD4 = 110000000 | I/O decoding for locations 1100000000b–1100000001b |
| 3. | VD3 = 0 | No byte swap of high and low bytes |
| 4. | VD2 = 0 | ISA Bus interface, i.e. non- MC68K interface |
| 5. | VD1 = 0 | Indexed I/O |
| 6. | VD0 = 0 | 8-bit bus interface |

Where 1 = pull-up with a 10K resistor; 0 = no pull-up resistor

#### Register Setting

All register settings are completely programmable and are dependent on display type, resolution, color and mode of operation, see *S1D13503 Hardware Functional Specification* for details.

# 2 MC68340 INTERFACE CONSIDERATIONS

## 2.1 Introduction

The S1D13503 is a general purpose LCD controller capable of interfacing to a variety of microprocessors. This interface is accomplished through the use of minimal external circuitry. This application note describes the interface between the S1D13503 and the 16-bit MC68340 microcontroller.

### Reference Material

Refer to the *S1D13503 Hardware Functional Specification* for complete AC timing details.

This document makes no attempts to describe the operation of the MC68340 microcontroller, please refer to the appropriate MC68340 documentation for this information.

## 2.2 MC68340 MPU Interface

The following sections provide the necessary settings and equations to complete the interface between the S1D13503 and the MC68340 microcontroller.

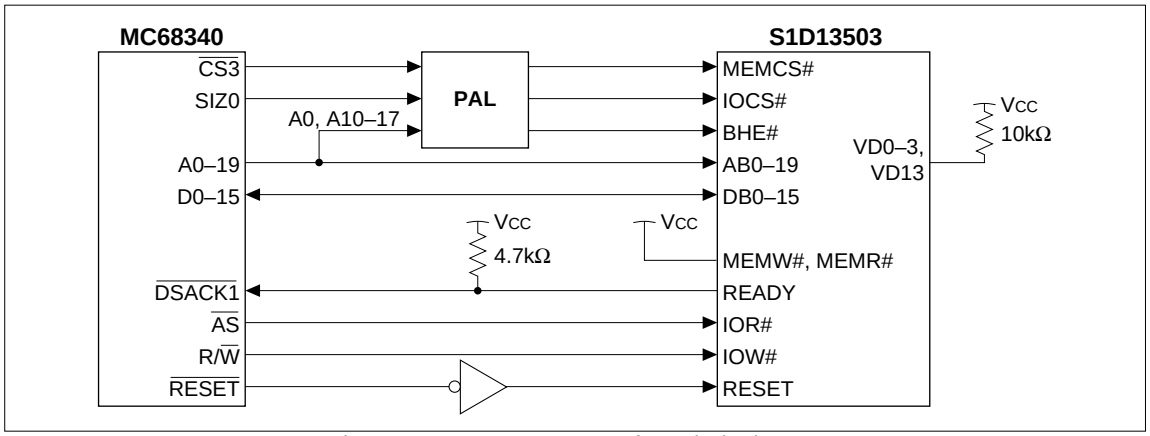



Figure 2-1 MC68340 MPU Interface Block Diagram

### MC68340 Setup

For the purpose of this example, the following conditions apply:

The internal chip select signal $\overline{CS3}$ of the MC68340, along with external $\overline{DSACK1}$ response, is employed to access the S1D13503. Direct mapping of the I/O with starting address at 00000000h, and 128K bytes of display memory with starting address 00020000h are also used.

1. $\overline{CS3}$ with 256K byte block size - starting address at 00000000h and ending address at 0003FFFFh
2. External $\overline{DSACK1}$ response - 16-bit port
3. Don't care Function Codes and with CPU space access
4. Both read and write accesses are allowed

Settings for the Address Mask register and Base Address register for the above conditions are:

058h–05Bh = 0003FFFFh Address Mask register

05Ch–05Fh = 000000F5h Base Address register

### PAL Equations

The PAL is programmed with the following equations:

1. With direct-mapping I/O occupying locations from 00000000h to 0000000Fh and A4 to A9 decoded internally to S1D13503;
   IOCS# = !(!$\overline{CS3}$ & !A17 & !A16 & !A15 & !A14 & !A13 & !A12 & !A11 & !A10)
2. With memory locations from 00020000h to 0003FFFFh and A17 to A19 decoded internally to S1D13503;
   MEMCS# = $\overline{CS3}$
3. BHE# becomes valid for two conditions:
   - 16-bit or 32-bit cycle, i.e., SIZ0 = 0
   - 8-bit cycle with odd byte access, i.e., SIZ0 = 1 and A0 = 1;

   BHE# = SIZ0 & !A0

## *S1D13503 Default Setup*

### Configuration Options

1. VD15–VD13 = 001 memory decoding for locations 20000h–3FFFFh
2. VD12–VD4 = 000000xxx I/O decoding for locations 0000000000b–0000001111b
3. VD3 = 1 byte swap of high and low bytes
4. VD2 = 1 MC68K interface
5. VD1 = 1 direct-mapping I/O
6. VD0 = 1 16-bit bus interface

Where x = don't care; 1 = pull-up with a 10K resistor; 0 = no pull-up resistor

**Note:** The states of these data pins are internally latched during RESET.

### Register Setting

AUX[1] bit 1 = 0 for 16-bit memory interface (must be 16-bit with a 16-bit bus).

# 3 LCD PANEL OPTIONS / MEMORY REQUIREMENTS

## 3.1 Introduction

The S1D13503 is a highly configurable general purpose LCD controller. The LCD panel frame-rate, resolution, and number of colors / gray shades all determine the memory and input clock requirements. This application note describes the equations used to determine the various parameters. An example resolution and desired frame-rate will be selected and used to determine the remaining variables.

### *Reference Material*

Refer to the *S1D13503 Hardware Functional Specification* for complete AC timing details.

## 3.2 Configuration Equations

This application note will follow one example through all the required calculations, for a complete description of all formula and associated parameters refer to the *S1D13503 Hardware Functional Specification*.

### Example

LCD Panel Resolution: $320 \times 240$

LCD Panel Configuration: 8-bit, Single Panel, Single Drive Panel

LCD Colors: 256

Desired Frame-rate: 70Hz

S1D13503 Operating Voltage: 3.3V

#### Input Clock Requirement Calculation

For a frame rate of 70Hz, the input clock (or pixel clock) frequency can be calculated as following,

$$fosc = \text{Input clock}$$

$$fosc = \text{FrameRate} \times (\text{NumberOfHorizontalPixels} + PHNDP + DHNDP) \times (\text{NumberOfVerticalLines} + 4)$$

Where DHNDP is Default Horizontal Non-Display Period in term of pixels:

DHNDP = 16 pixels in gray shade display modes, and

DHNDP = 32 pixels in BW display mode and in color display modes.

Where PHNDP is Programmable Horizontal Non-Display Period in term of pixels:

PHNDP = 0 pixel when AUX[0C] = 0, and

$$PHNDP = \frac{(AUX[0C] + 1) \times (\text{MemoryInterfaceWidth})}{(\text{BitsPerPixel})}$$ pixels when AUX[0C] not equal to zero.

**Note:** For this example we will use DHNDP = 32, PHNDP = 0

Therefore;

$fosc = 70 \times (320 + 32) \times (240 + 4)$

$fosc = 6.0MHz$

### SRAM Size and Access Time Requirements

#### SRAM Size

$$\text{Memory Size (bytes)} = \frac{(\text{HorizontalPixels}) \times (\text{VerticalLines}) \times (\text{BitsPerPixel})}{8}$$

i.e., 256 colors = 8 bits / pixel, therefore 1 byte (8 bits) = 1 pixel

Therefore:

Memory size (bytes) = $(320 \times 240) \times 8 / 8$

Memory size (bytes) = 76.8K bytes.

**Note:** For a detailed description of the memory size requirement, see Section 9.4 of the *S1D13503 Hardware Functional Specification*.

### SRAM Access Time

To support 256 color modes the S1D13503 must be configured to support a 16-bit data path into display memory (SRAM).

For 16-bit display memory interface the required SRAM access time must be;

SRAM Access time ≤ 1/fOSC - 40ns. (3.3V specification)

Therefore using a 6.0MHz input clock;

SRAM access time must be ≤ 127ns.

**Note:** For detail description of the SRAM access time, see Section 9.2 of the *S1D13503 Hardware Functional Specification*.

## *3.3 Conclusions*

To support a 320 × 240 256 color panel at 70Hz refresh, you require a 6.0MHz input clock and 76.8K bytes of 127ns SRAM.

## *3.4 Implementation*

### *16-Bit Display Memory Interface*

Since 76.8K bytes with at least 127ns access time SRAM is required, one 64K × 16 byte SRAM with 120ns access time will be used for this example.

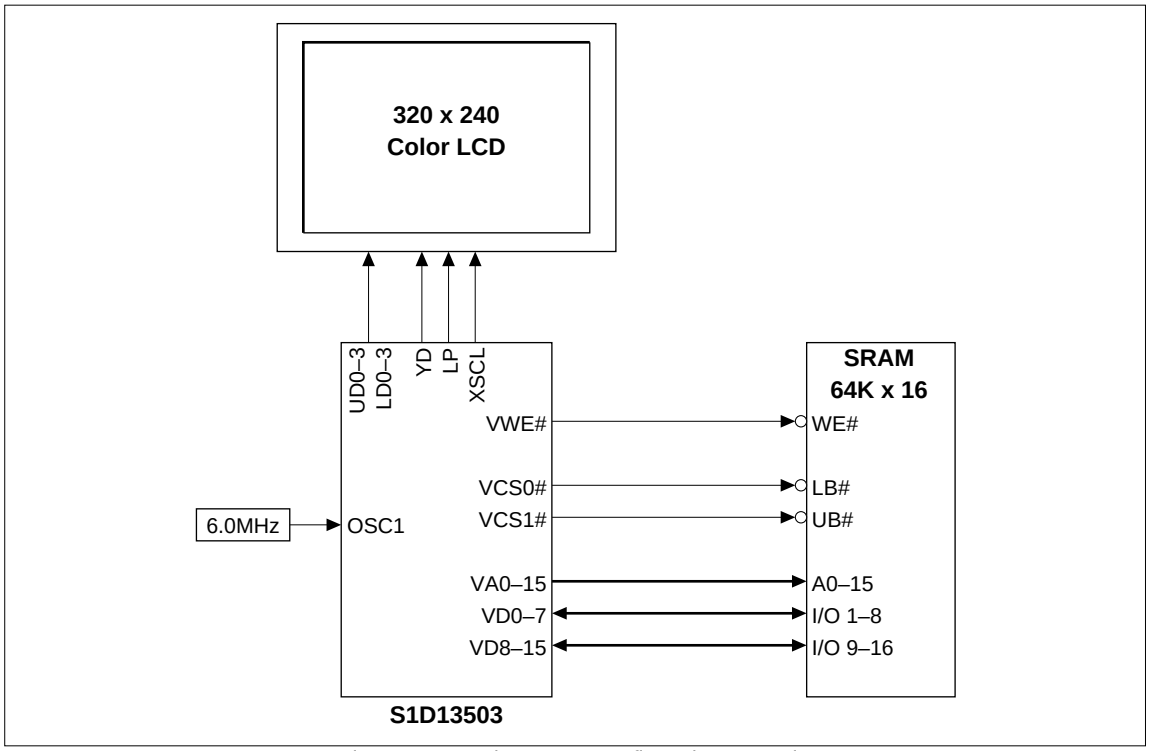



Figure 3-1 16-Bit Memory Configuration Example

### Configuration Options

VD0 = pull-up (with a 10K resistor) for 16-bit bus interface.

Other option settings are not related to this implementation.

### Register Settings

| | |
|---|---|
| AUX[0] = 0000 0000 | not in test mode |
| AUX[1] = 1011 100x | 8-bit single panel, 256 color, 16-bit display memory interface |
| AUX[2] = 1001 1111 | horizontal resolution = 320 ; 256 colors = 1 pixel per byte; 1 pixel per fetch |
| AUX[3] = 0000 0110 | not in power save modes |
| AUX[4] = 1110 1111 | total 240 scan lines |
| AUX[5] = 0000 0000 | WF = 0 |
| AUX[6] = 0000 0000 | |
| AUX[7] = 0000 0000 | default starting address at 0000h (with AUX[6]) |
| AUX[8] = xxxx xxxx | don't care when not using split screen |
| AUX[9] = xxxx xxxx | don't care when not using split screen |
| AUX[A] = 1110 1111 | together with AUX[B] bit1–0, should be the same as or larger than AUX[5] bit1–0 and |
| AUX[B] = xxxx xx00 | AUX[4] when not using split screen |
| AUX[D] = 0000 0000 | no virtual screen |

x = don't care

A sample of values for the Look-Up Table to produce 256 colors is shown below;

RED: [00 02 04 06 09 0B 0D 0F]0F 0D 0B 09 06 04 02 00

GREEN: [00 02 04 06 09 0B 0D 0F]0F 0D 0B 09 06 04 02 00

BLUE: [00 05 0A 0F]0F 0A 05 00 01 06 09 0E 0D 09 04 02

**Note:** Refer to *S1D13503 Programming Notes and Examples, S18A-G-002-02*, for further information.

# 4 S1D13503 POWER CONSUMPTION

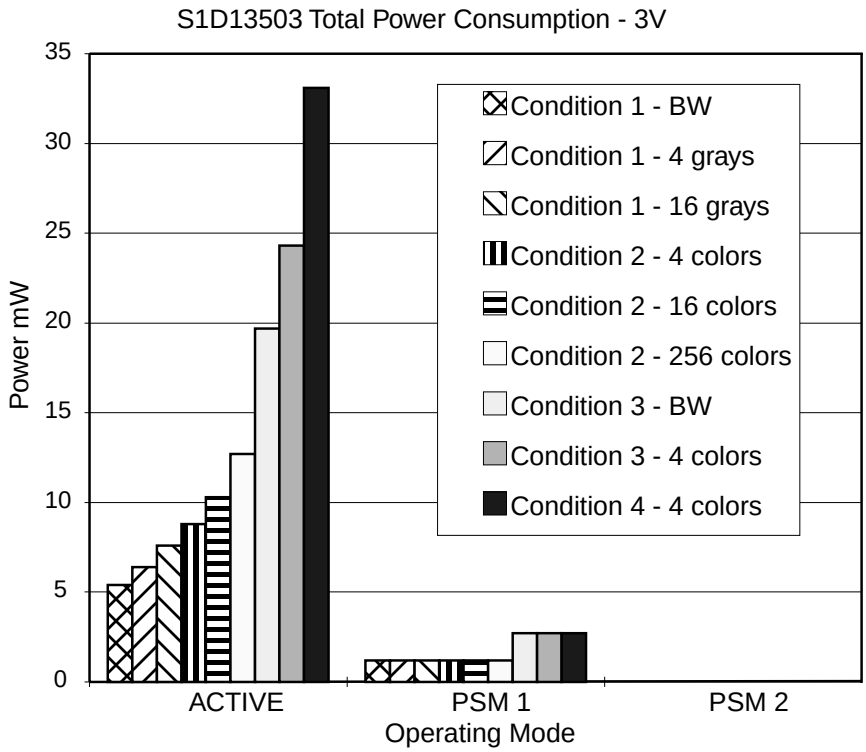
## 4.1 Conditions

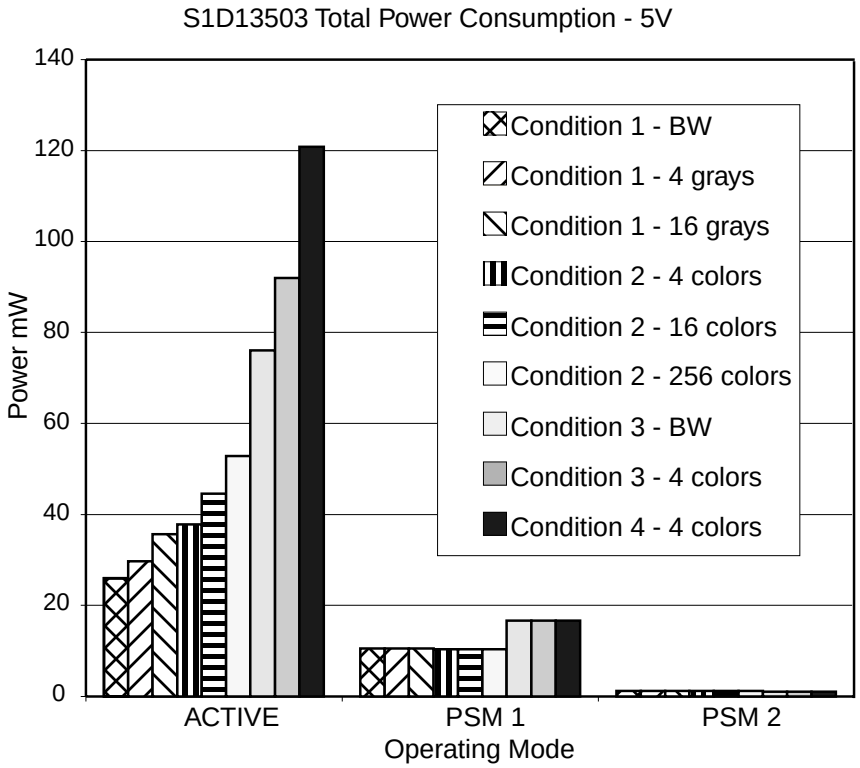
Table 4-1 Total Power Consumption - 3V

| | Test Condition | Gray Shades / Colors | Total Power Consumption | | |
|---|---|---|---|---|---|
| | | | Active | Power Save Mode | |
| | | | | 1 | 2 |
| 1 | Input Clock = 6MHz<br>LCD Panel Connected = 320 × 240 Monochrome<br>$V_{DD}$ = 3.0V | Black-and-White<br>4 Grays<br>16 Grays | 5.4 mW<br>6.4 mW<br>7.6 mW | 1.2 mW<br>1.2 mW<br>1.2 mW | less than 300 µW<br>less than 300 µW<br>less than 300 µW |
| 2 | Input Clock = 6MHz<br>LCD Panel Connected = 320 × 240 Color<br>$V_{DD}$ = 3. 0V | 4 Colors<br>16 Colors<br>256 Colors | 8.8 mW<br>10.3 mW<br>12.7 mW | 1.2 mW<br>1.2 mW<br>1.2 mW | less than 300 µW<br>less than 300 µW<br>less than 300 µW |
| 3 | Input Clock = 25MHz<br>LCD Panel Connected = 640 × 480 Monochrome<br>$V_{DD}$ = 3.0V | Black-and-White<br>4 Grays | 19.7 mW<br>24.3 mW | 2.7 mW<br>2.7 mW | less than 300 µW<br>less than 300 µW |
| 4 | Input Clock = 25MHz<br>LCD Panel Connected = 640 × 480 Color<br>$V_{DD}$ = 3.0V | 4 Colors | 33.1 mW | 2.7 mW | less than 300 µW |

Table 4-2 Total Power Consumption - 5V

| | Test Condition | Gray Shades / Colors | Total Power Consumption | | |
|---|---|---|---|---|---|
| | | | Active | Power Save Mode | |
| | | | | 1 | 2 |
| 1 | Input Clock = 6MHz<br>LCD Panel Connected = 320 × 240 Monochrome<br>$V_{DD}$ = 5.0V | Black-and-White<br>4 Grays<br>16 Grays | 26.0 mW<br>29.7 mW<br>35.7 mW | 10.5 mW<br>10.5 mW<br>10.5 mW | 1.2 mW<br>1.2 mW<br>1.2 mW |
| 2 | Input Clock = 6MHz<br>LCD Panel Connected = 320 × 240 Color<br>$V_{DD}$ = 5.0V | 4 Colors<br>16 Colors<br>256 Colors | 37.8 mW<br>44.5 mW<br>52.8 mW | 10.5 mW<br>10.5 mW<br>10.5 mW | 1.2 mW<br>1.2 mW<br>1.2 mW |
| 3 | Input Clock = 25MHz<br>LCD Panel Connected = 640 × 480 Monochrome<br>$V_{DD}$ = 5.0V | Black-and-White<br>4 Grays | 76.0 mW<br>92.0 mW | 16.7 mW<br>16.7 mW | 1.0 mW<br>1.0 mW |
| 4 | Input Clock = 25MHz<br>LCD Panel Connected = 640 × 480 Color<br>$V_{DD}$ = 5.0V | 4 Colors | 120.8 mW | 16.7 mW | 1.0 mW |


S1D13503 Total Power Consumption - 3V
Power mW
35
30
25
20
15
10
5
0
ACTIVE
PSM 1
PSM 2
Operating Mode
Condition 1 - BW
Condition 1 - 4 grays
Condition 1 - 16 grays
Condition 2 - 4 colors
Condition 2 - 16 colors
Condition 2 - 256 colors
Condition 3 - BW
Condition 3 - 4 colors
Condition 4 - 4 colors


S1D13503 Total Power Consumption - 5V
Power mW
140
120
100
80
60
40
20
0
ACTIVE
PSM 1
PSM 2
Operating Mode
Condition 1 - BW
Condition 1 - 4 grays
Condition 1 - 16 grays
Condition 2 - 4 colors
Condition 2 - 16 colors
Condition 2 - 256 colors
Condition 3 - BW
Condition 3 - 4 colors
Condition 4 - 4 colors

# 5 S1D13503 / SED1352 COMPARISON

The S1D13503 is pin compatible with, and includes all features of the SED1352. This allows an easy upgrade path for the system designer, both from the hardware and software aspect. The purpose of this document is to briefly describe the differences between these two controllers, for further details refer to the individual *Hardware Functional Specifications*.

## 5.1 Feature Comparison

Table 5-1 S1D13503 / SED1352 Feature Comparison

| Feature | S1D13503 | SED1352 |
|---|---|---|
| Color | • 4 / 16 / 256 colors | • Not available |
| Monochrome | • Black-and-White<br>• 4 / 16 Gray Shades | • Not available<br>• 4 / 16 Gray Shades |
| Display Data Formats | • 4 / 8-bit, Single / Dual Monochrome panel support<br>• 4 / 8 / 16-bit Single / Dual Color panel support (Note) | • 4 / 8-bit, Single / Dual Monochrome panel support<br>• Not available |
| Programmable Horizontal Non-Display Period | • Yes | • Fixed |
| Look-Up Tables | • 3 × 16 position, 4-bit wide Look-Up Tables | • 1 × 16 position, 4-bit wide Look-Up Table |
| Revision Code | • 2-bit fixed | • 1-bit fixed |

**Note:** 16-bit color panel support is provided by the S1D13503 using external logic.

All other features not mentioned above are supported by both controllers. See the *S1D13503 Hardware Functional Specification*, and the *SED1352 Hardware Functional Specification*, for further details.

## 5.2 S1D13503 Register Changes / Additions from the SED1352

See the *S1D13503 Hardware Functional Specification*, for details on these registers.

**AUX[01h]**

bit 2 LCD Data Width Bit 0
bit 3 Gray Shade / Color

**AUX[03h]**

bit 1 Color Mode
bit 2 BW / 256 Colors
bit 3 LCD Data Width Bit 1

**AUX[0Ch]**

bit 0:7 Horizontal Non-Display Period

**AUX[0Eh]**

bit 4 ID Bit / RGB Index Bit 0
bit 5 ID Bit / RGB Index Bit 1
bit 6 Green Bank Bit 0
bit 7 Green Bank Bit 1

**AUX[0Fh]**

bit 4 Blue Bank Bit 0
bit 5 Blue Bank Bit 1
bit 6 Red Bank Bit 0
bit 7 Red Bank Bit 1

# EPSON International Sales Operations

## AMERICA

### EPSON ELECTRONICS AMERICA, INC.

**- HEADQUARTERS -**
150 River Oaks Parkway
San Jose, CA 95134, U.S.A.
Phone: +1-408-922-0200 Fax: +1-408-922-0238

**- SALES OFFICES -**

**West**
1960 E. Grand Avenue
El Segundo, CA 90245, U.S.A.
Phone: +1-310-955-5300 Fax: +1-310-955-5400

**Central**
101 Virginia Street, Suite 290
Crystal Lake, IL 60014, U.S.A.
Phone: +1-815-455-7630 Fax: +1-815-455-7633

**Northeast**
301 Edgewater Place, Suite 120
Wakefield, MA 01880, U.S.A.
Phone: +1-781-246-3600 Fax: +1-781-246-5443

**Southeast**
3010 Royal Blvd. South, Suite 170
Alpharetta, GA 30005, U.S.A.
Phone: +1-877-EEA-0020 Fax: +1-770-777-2637

## EUROPE

### EPSON EUROPE ELECTRONICS GmbH

**- HEADQUARTERS -**
Riesstrasse 15
80992 Munich, GERMANY
Phone: +49-(0)89-14005-0 Fax: +49-(0)89-14005-110

**SALES OFFICE**
Altstadtstrasse 176
51379 Leverkusen, GERMANY
Phone: +49-(0)2171-5045-0 Fax: +49-(0)2171-5045-10

**UK BRANCH OFFICE**
Unit 2.4, Doncastle House, Doncastle Road
Bracknell, Berkshire RG12 8PE, ENGLAND
Phone: +44-(0)1344-381700 Fax: +44-(0)1344-381701

**FRENCH BRANCH OFFICE**
1 Avenue de l' Atlantique, LP 915 Les Conquerants
Z.A. de Courtaboeuf 2, F-91976 Les Ulis Cedex, FRANCE
Phone: +33-(0)1-64862350 Fax: +33-(0)1-64862355

**BARCELONA BRANCH OFFICE**
**Barcelona Design Center**
Edificio Prima Sant Cugat
Avda. Alcalde Barrils num. 64-68
E-08190 Sant Cugat del Vallès, SPAIN
Phone:+34-93-544-2490 Fax:+34-93-544-2491

## ASIA

### EPSON (CHINA) CO., LTD.

28F, Beijing Silver Tower 2# North RD DongSanHuan ChaoYang District, Beijing, CHINA
Phone: 64106655 Fax: 64107319

**SHANGHAI BRANCH**

4F, Bldg., 27, No. 69, Gui Jing Road
Caohejing, Shanghai, CHINA
Phone: 21-6485-5552 Fax: 21-6485-0775

### EPSON HONG KONG LTD.

20/F., Harbour Centre, 25 Harbour Road
Wanchai, Hong Kong
Phone: +852-2585-4600 Fax: +852-2827-4346
Telex: 65542 EPSCO HX

### EPSON TAIWAN TECHNOLOGY & TRADING LTD.

10F, No. 287, Nanking East Road, Sec. 3
Taipei
Phone: 02-2717-7360 Fax: 02-2712-9164
Telex: 24444 EPSONTB

**HSINCHU OFFICE**

13F-3, No. 295, Kuang-Fu Road, Sec. 2
HsinChu 300
Phone: 03-573-9900 Fax: 03-573-9169

### EPSON SINGAPORE PTE., LTD.

No. 1 Temasek Avenue, #36-00
Millenia Tower, SINGAPORE 039192
Phone: +65-337-7911 Fax: +65-334-2716

### SEIKO EPSON CORPORATION
### KOREA OFFICE

50F, KLI 63 Bldg., 60 Yoido-dong
Youngdeungpo-Ku, Seoul, 150-763, KOREA
Phone: 02-784-6027 Fax: 02-767-3677

### SEIKO EPSON CORPORATION
### ELECTRONIC DEVICES MARKETING DIVISION

**Electronic Device Marketing Department**
**IC Marketing & Engineering Group**
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5816 Fax: +81-(0)42-587-5624

**ED International Marketing Department**
**Europe & U.S.A.**
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5812 Fax: +81-(0)42-587-5564

**ED International Marketing Department**
**Asia**
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5814 Fax: +81-(0)42-587-5110


In pursuit of **“Saving” Technology**, Epson electronic devices.
Our lineup of semiconductors, liquid crystal displays and quartz devices
assists in creating the products of our customers’ dreams.
**Epson IS energy savings**.

**SEIKO EPSON CORPORATION**
ELECTRONIC DEVICES MARKETING DIVISION

■ **EPSON Electronic Devices Website**

http://www.epson.co.jp/device/



**First issue October,1997** Ⓜ
**Printed April, 2001 in Japan** ⒸB