SSD2805C

MIPI Getting Started

1 GENERAL DESCRIPTION

This document is a quick start guide to MIPI System Setup covering MIPI Software Development and MIPI master chip hardware configurations.

2 OVERVIEW

2.1 MIPI System Setup Flow

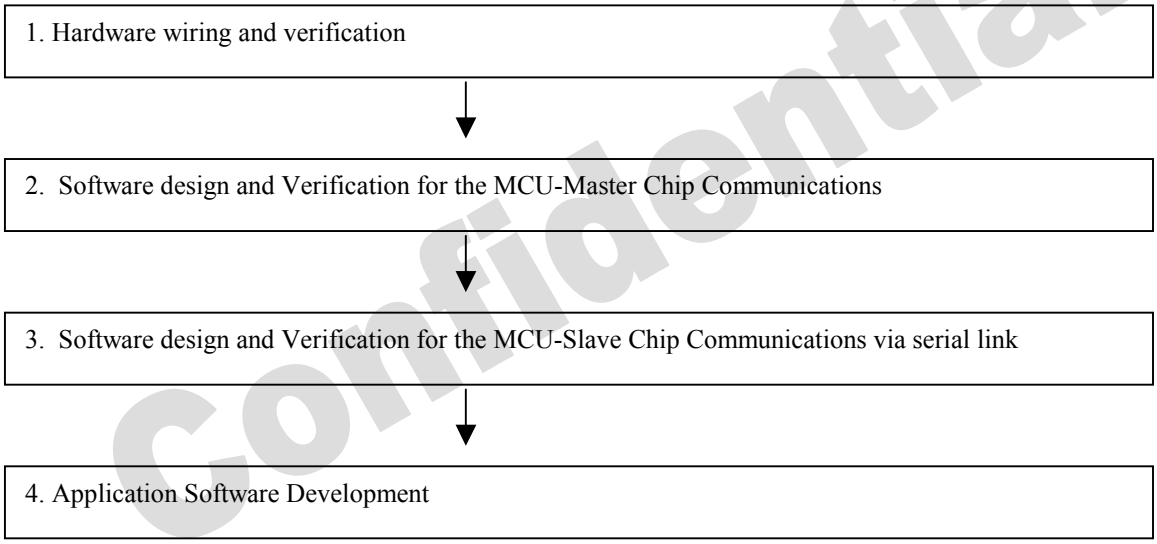
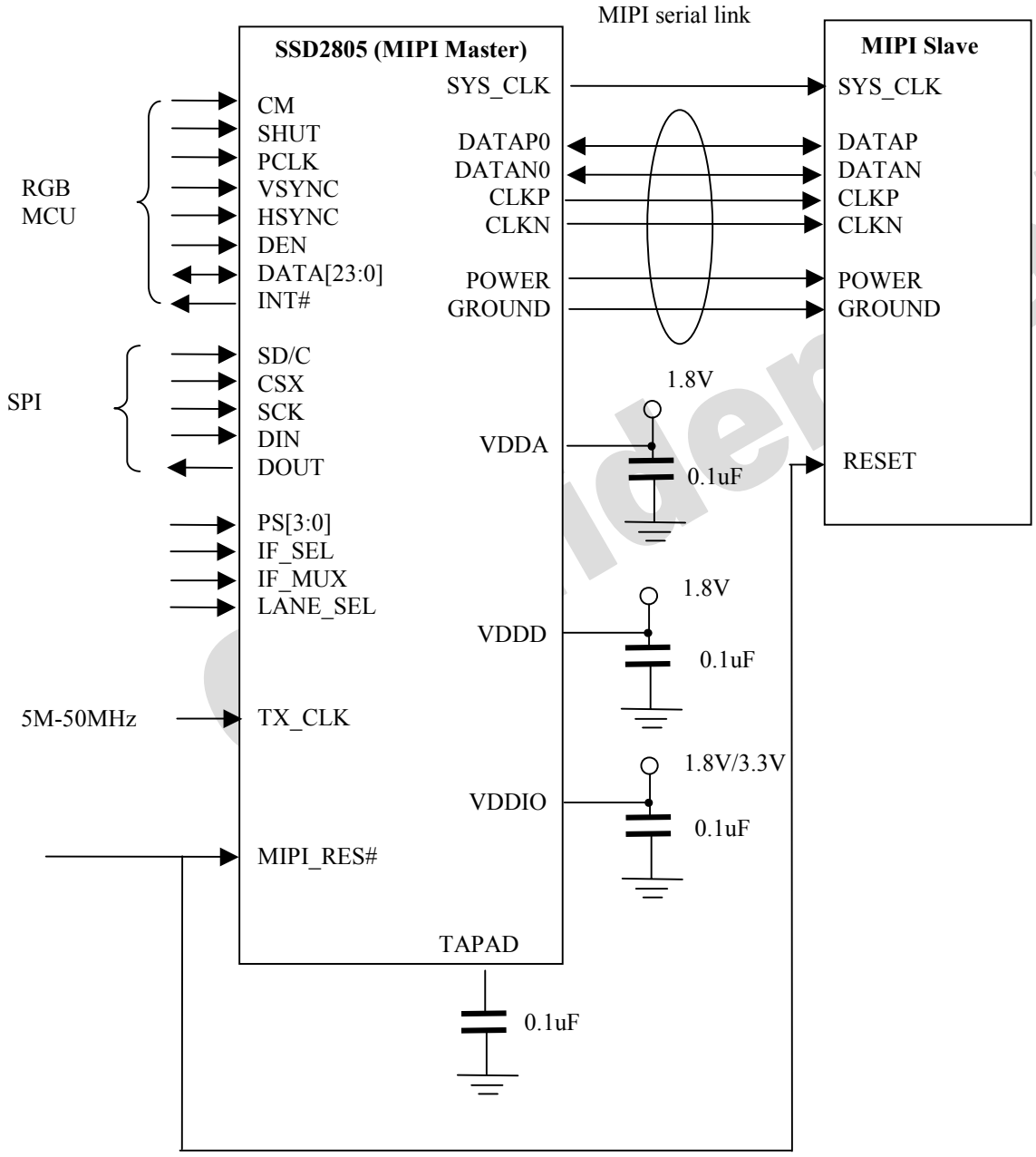


Figure 1 MIPI System Setup Flow

3 GETTING STARTED

3.1 Hardware wiring and verification



*The master and slave device should have a common ground.
*A 0.1uF decoupling capacitor should be applied to VDDA, VDDD and VDDIO respectively.

Figure 2 Block Diagram of application example *

* Please refer to the latest application example in product proposal/datasheet.

3.1.1 Check List for hardware wiring and verification

Hardware Wiring

1. Power
   Connect all power pins

2. Check the Input clock signal to TX_CLK (e.g. 20 MHz)

3. Chip Select Signal
   Connect the CSX for accessing the master chip registers (active low signal)

4. MIPI_RES#
   Design a power on reset mechanism for MIPI_RES#

5. TEST[2:0] tie to GND
6. TAPAD output 1.2V after hardware reset

7. MCU Interface between Master Chip and MCU (MCU/RGB/SPI)

7.1. Select the MCU-Master Chip interface type

Pins	Description	Reference Settings: RGB Interface+ SPI 8 bit 3 wire
PS[3:0]	Interface selection signal PS[1:0] is for SPI interface - 00: 3 wire 24 bit SPI interface - 01: 3 wire 8 bit SPI interface - 10: 4 wire 8 bit SPI interface - 11: No SPI interface PS[3:2] is for the MCU interface When IF_SEL is 1 - 00: 8 bit 8080 MCU interface - 01: 16 bit 80800 MCU interface - 10: 8 bit 6800 MCU interface - 11: 16 bit 6800 MCU interface (Program through register settings for 16, 18 and 24 bit RGB interfaces when IF_SEL selected 0)	PS[1:0]=01 (3 wire 8 bit SPI interface) PS[3:2]=don't care
IF_SEL	Interface selection signal - 0: A combination of RGB and SPI interface is selected - 1: A combination of MCU interface is selected	IF_SEL=0 (A combination of RGB and SPI interface is selected)
IF_MUX	Interface multiplex selection signal - 0: Multiplex scheme 1 - 1: Multiplex scheme 2	IF_MUX=0 (Multiplex scheme 1)
LANE_SEL	Number of data lane selection signal - 0: 1 data lane - 1: 2 data lanes	LANE_SEL=0 (1 data lane)

8. Test reading and writing the Master Chip Register (SPI 8 bit 3 wire example)
   Each SSD2805 register address refers to a 16-bit register value.
   8.1. Reading register [0xB0] is the first step to verify the MCU-SSD2805 communications.
   (Note: Please reset the SSD2805 chip at MIPI_RES pin before after power up)
   8.1.1. Make sure the SPI signal meets the timing requirement specified in SSD2805C datasheet.

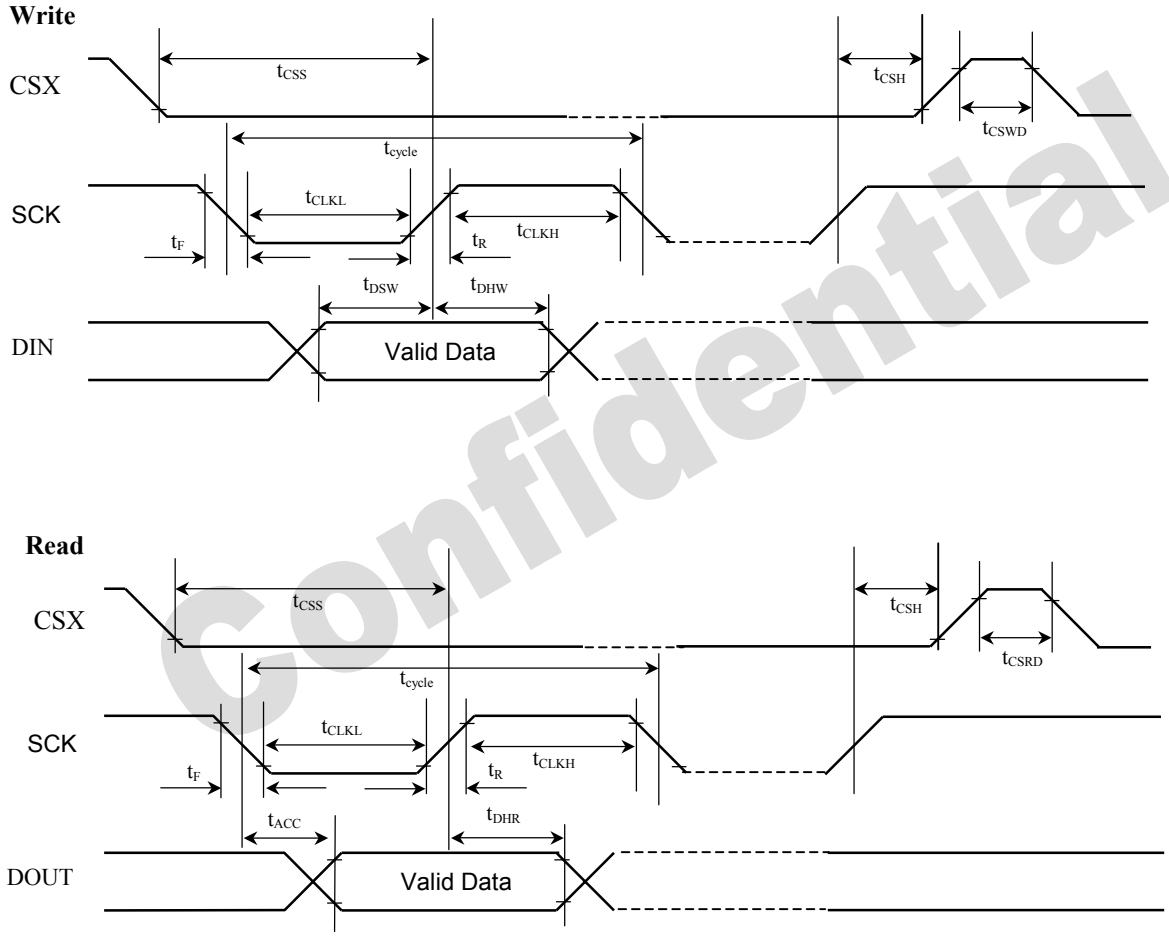


Figure 3: 8 bit 3 wire SPI Interface Timing Diagram

8.1.2. Read operation via SPI 8 bit 3 wire

SPI 8bit 3-wire interface consists of SCK, DIN, DOUT and SCSX. It only supports 8 bit data. Each cycle contains 8 bit data. The **first cycle** should be a write cycle to specify the **register address** for access. The subsequent cycles are read or write cycles for read or write operations.

The SCSX should be driven from 1 to 0 to start an operation and from 0 to 1 to end an operation. During 1 operation, the application processor can write or read multiple bytes.

Instead of SD/C_, an SDC bit is used to indicate whether the operation is for data or command. Each byte is associated with an SDC bit. When SDC is 1, the operation is for display data. When SDC is 0, the operation is for command. The SDC bit is sent priori to each byte. In other words, the SDC bit is the first bit of every 9 bits during 1 operation.

During read operation, since there is no RWX signal to indicate whether the operation is read or write, the read operation for SPI interface needs to be handled differently from the MCU interface. After SCSX is driven low, the first cycle is always a command write cycle, which specifies the register to access. The second cycle is still a command write cycle. If the command in this cycle matches the command in register **LRR**, the SPI interface will enter read mode. The subsequent cycles will be read cycles. If the command does not match, the SPI interface will remain in write mode.

After entering the read mode, the return data is provided on DOUT, on the falling edge of SCK. The application processor should use the rising edge of SCK to sample the data. Please note that there is no SDC bit to read out from SSD2805. Hence, each read cycle consists of 8 bits instead of 9 bits. This is the difference between read and write cycles. Please see the diagram below for illustration. Optionally, the SCSX can be driven to 1 in between cycles.

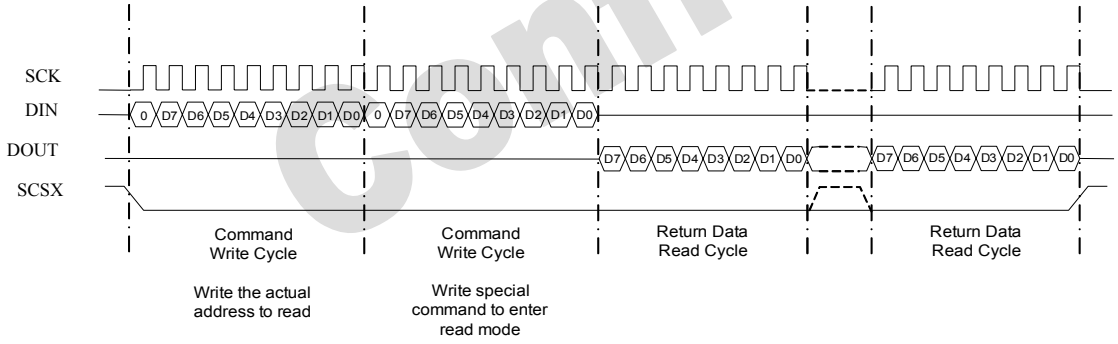
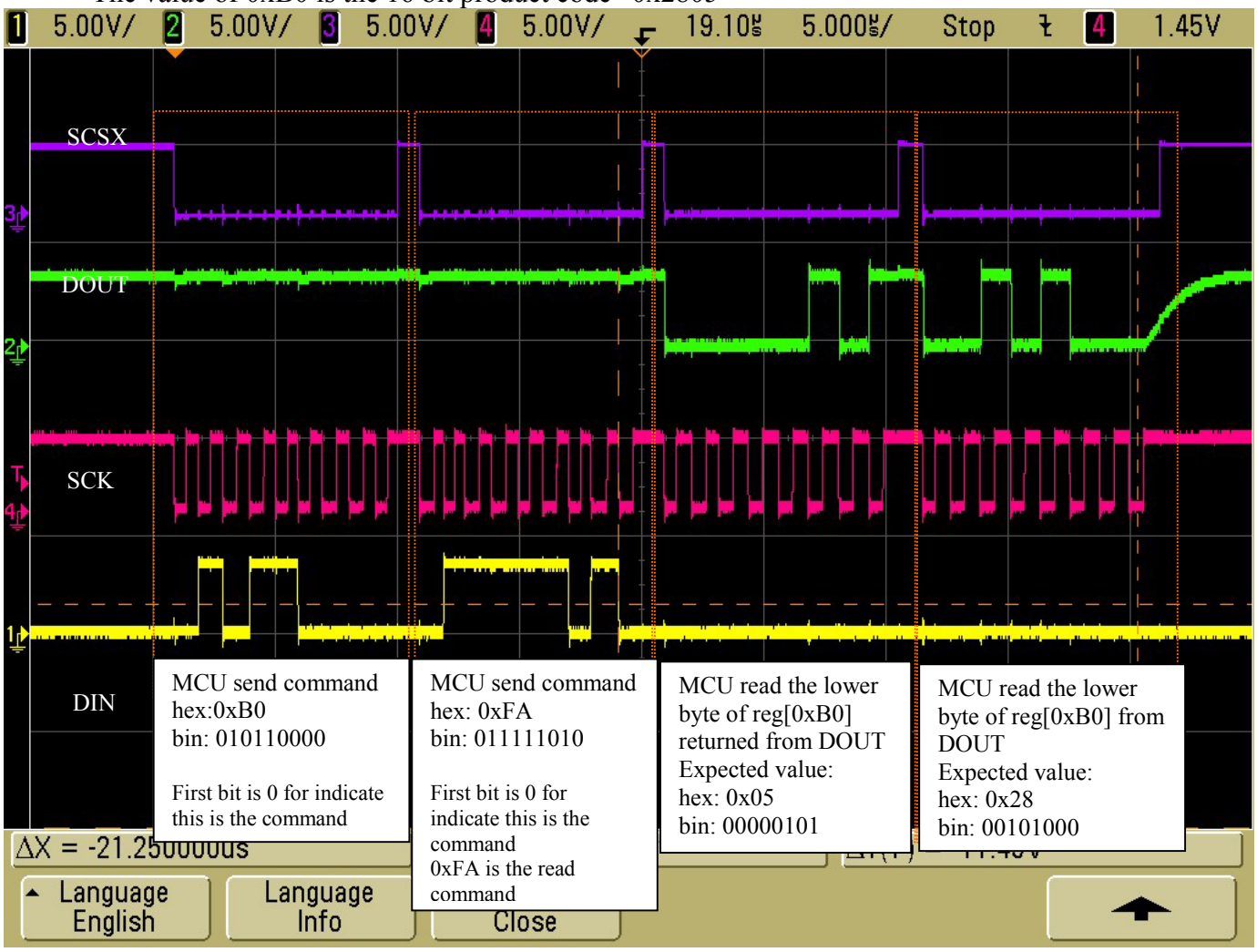


Figure 4: Illustration of Read Operation for 8 bit 3 Wire Interface

8.1.3. Read product code from register[0xB0] via SPI 8 bit 3 wire

The value of 0xB0 is the 16 bit product code "0x2805"



8.2. Test reading all SSD2805 master chip register value after reading register [0xB0]

8.2.1. To verify the correctness of the read back value, please refer to the default register value after reset.

8.3. Test writing SSD2805 master chip register

8.3.1. Write operation via SPI 8 bit 3-wire interface

Follow the timing requirement in 8.1.1.

During write operation, DIN will be sampled by SSD2805 at the rising edge of SCK. The first rising edge of SCK after the falling edge of SCSX samples the SDC bit. The second rising edge samples bit 7 of the 8 bit data. The third rising edge of SCK samples the bit 6 of the 8 bit data, and so on. Please see the diagram below for illustration. Optionally, the SCSX can be driven to 1 in between cycles.

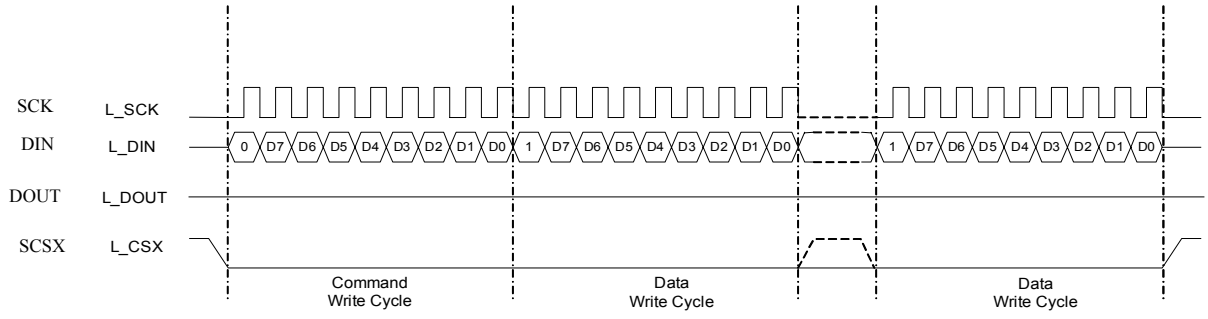
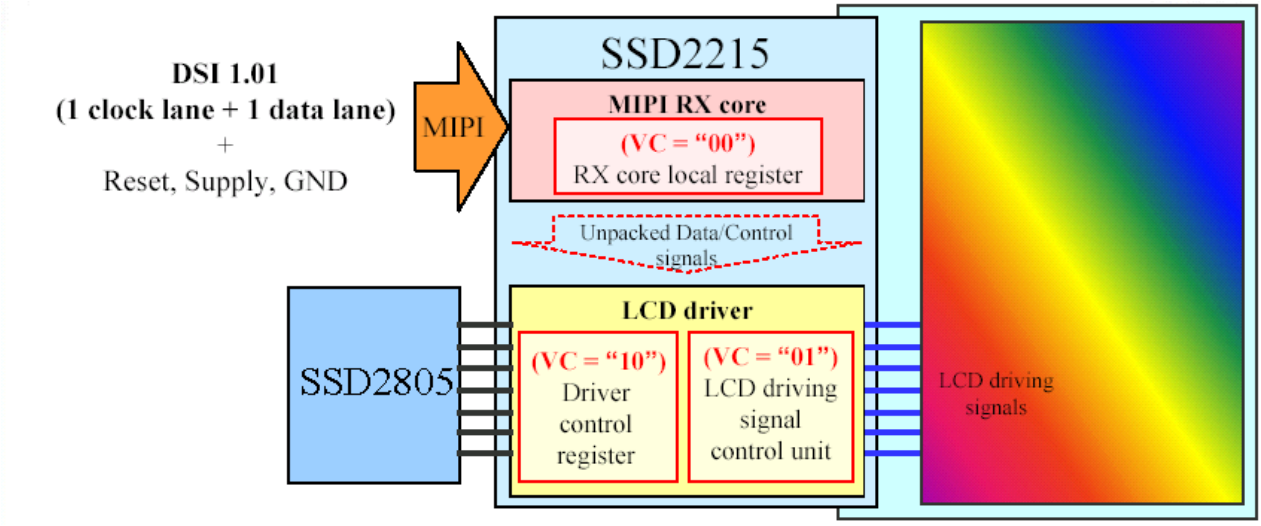


Figure 5: Illustration of Write Operation for 8 bit 3 Wire Interface

9. Reference Design using SSD2805 (Master Bridge) with SSD2215 (MIPI TFT Driver)

9.1. System Block Diagram

SSD2215 is a MIPI Slave Bridge embedded RAM-less TFT driver supporting QVGA(240x320) 24bpp. The following are the block diagram illustrating SSD2805 with SSD2215.



SSD2215 consist of 3 VC (virtual channel) for different logic blocks control.

-VC="00" used for MIPI RX core register. To setup the MIPI channel related setting, user must send generic command to VC="00".

-VC="10" used for LCD driver control register. User has to send DCS/Generic command to VC="10" for display control. (e.g. Display On/Off, Inversion, LCD driving voltage setting…)

-VC="01" used for timing control/DAC of the LCD driving signal. All video packets must send to VC="01" in order to allow SSD2215 keep refresh/update the display.

(Note: SSD2215 support video mode display, hence, TX side must send video stream to SSD2215 continuously to maintain display on).

Figure 6: Block Diagram of Reference System (SSD2805+SSD2215)

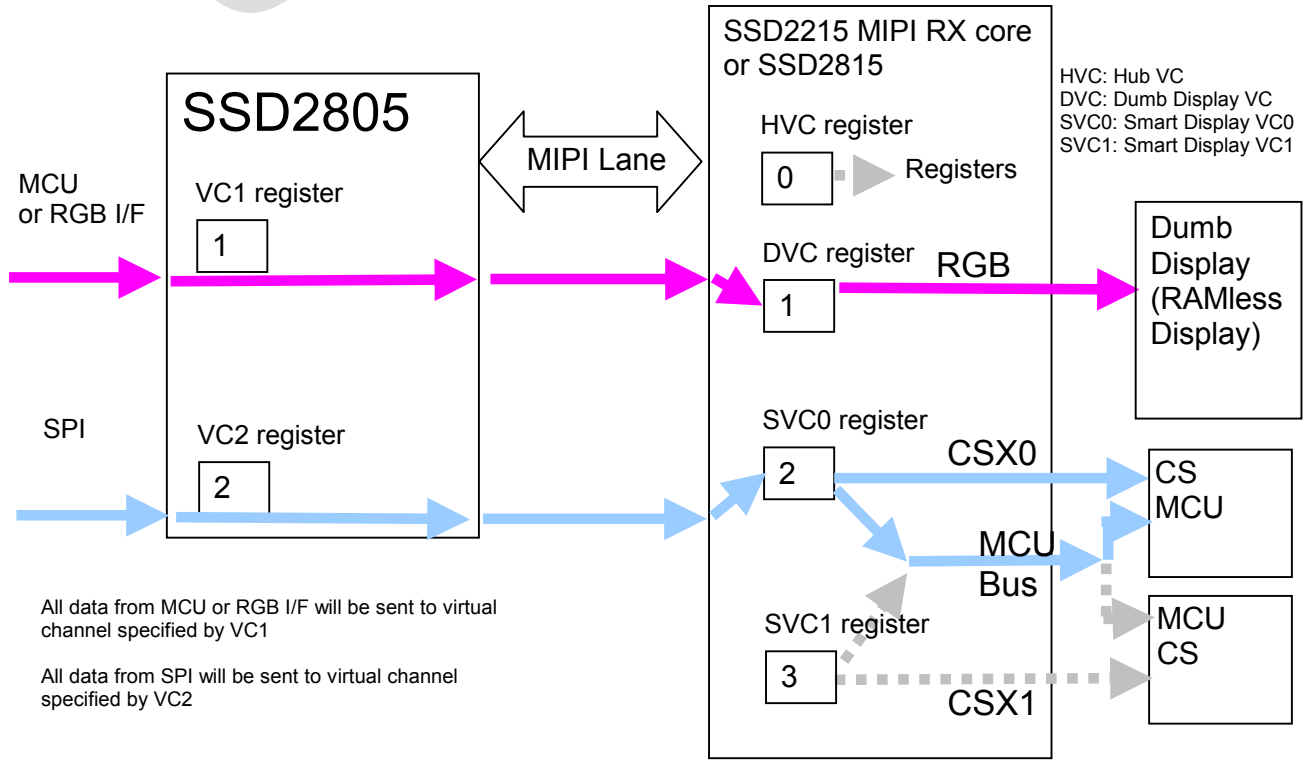


Figure 7: Virtual Channel System

9.2. Software Flow to display data (Host Processor communicates with SPI 8 bit 3 wire)

Step 1. System Reset >100us
Pull low Reset pin for both SSD2805 and SSD2215

Note 2.1:
Register [addr]= <value> is to write local register
Example: Register[BA]= 0x311F
is to write SSD2805 local register(0xBA) with value 0x311F

Set Register is implemented by (assume SPI 8 bit 3 wire is used)
SPI Write (COMMAND) 0xBA //Address
SPI Write (DATA) 0x1F //Low bye Value
SPI Write (DATA) 0x31 //High Byte Value

Step 2. SSD2805 PLL Initialization [Note 2.1 and 2.2]

//Set PLL (Register[0xBA]=0x311F)
SPI Write (COMMAND) 0xBA
SPI Write (DATA) 0x1F
SPI Write (DATA) 0x31

//Enable PLL (Register[0xB9]=0x0001)
SPI Write (COMMAND) 0xB9
SPI Write (DATA) 0x01
SPI Write (DATA) 0x00

Note 2.2:
If PLL has not yet locked, the whole system is operating using the PLL input clock.
The SPI serial clock should be 1/8 of this system clock.

For example, 10MHz is input at TX_CLK during power up. The whole system is running at 10MHz. (REG[B7] Bit 5 (**CSS)** is 0 by default). SPI serial clock should be less than 1.25MHz before PLL is locked.

Step 3. Check PLL Lock Status
(Method 1 or Method 2 or Method 3)

Method 1:
Use Hardware Interrupt to check the Lock Status

//By default,
//**PLSE** (PLL Lock Status Enable Bit) is 1 (enable)
//**PLS** (PLL Lock Status) map to **INT_** (interrupt) pin

Wait for **INT_** pin going low

Method 2:
Check PLS (PLL Lock Status) bit by Reading Register[0xC6] Bit 7

//Read Register[0xC6]
SPI Write (COMMAND) 0xC6
SPI Write (COMMAND) 0xFA
SPI Read =>Value Bit[7:0]
SPI Read => Value Bit[15:8]

//Repeat to check Register[0xC6] Bit[7] **PLS** until **PLS**=1

Method 3:

Wait for 1 ms

Step 4. Configure MIPI RX Core Register (or SSD2815 Register) [Note 4]

Step 4.1 Setup before sending Generic Write Packet to RX Core Register

```
//Configure to Generic Write Mode, using LP Mode
//(Low Power)
//Register[0xB7]=0x0210
SPI Write (COMMAND)  0xB7
SPI Write (DATA)     0x10
SPI Write (DATA)     0x02

// Set VC2=0
//Note VC2 is for data input from SPI interface,
//i.e. Data from SPI will be route to Virtual Channel "0"
//By default, Virtual Channel 0 is for RX core registers
//Register[0xB8]=0x0041
SPI Write (COMMAND)  0xB8
SPI Write (DATA)     0x41
SPI Write (DATA)     0x00

//Set LP Clock Divider
//Register[0xBB]=0x0003
SPI Write (COMMAND)  0xBB
SPI Write (DATA)     0x03
SPI Write (DATA)     0x00

//Set Low byte of Transmit Data Count (TDC[15:0])
//Register[0xBC]=0x0004
SPI Write (COMMAND)  0xBC
SPI Write (DATA)     0x04
SPI Write (DATA)     0x00

//Set High byte of Transmit Data Count (TDC[31:16])
//Register[0xBD]=0x0000
SPI Write (COMMAND)  0xBD
SPI Write (DATA)     0x00
SPI Write (DATA)     0x00

//Set Packet Size thershold=4,write_reg(0xBE,0x0004)
//Register[0xBE]=0x0004
SPI Write (COMMAND)  0xBE
SPI Write (DATA)     0x04
SPI Write (DATA)     0x00
```

Step 4.2 Writing MIPI RX Core Register using Generic Write Packet

```
//Enable PCLK output for RGB interface
//RX_CORE_Register[0x0006]=0x0001
SPI Write (COMMAND)  0xBF
SPI Write (DATA)     0x06
SPI Write (DATA)     0x00
SPI Write (DATA)     0x01
SPI Write (DATA)     0x00
```

Note 4.1

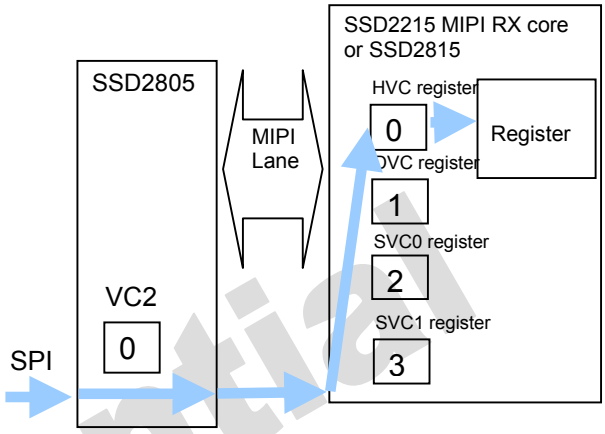


Figure 7. Configure MIPI RX Core Register

Note 4.2

In MIPI RX Core, the register address is 16 bit and value is also 16 bit.

For example, writing RX register [0x0004]=0x0207,
0x04 0x00 0x07 0x02
will be written to Generic Packet Drop register (REG[0xBF]).

So the Transmit Data Count (TDC) = 4
Packet Size threshold is the size for partitioning the packet data. To prevent partitioning, set Packet Size Thershold=4

Step 4. Configure MIPI RX Core Register (or SSD2815 Register) (continue)

Step 4.2 Writing MIPI RX Core Register using Generic Write Packet (continue)

```
//Set VSYNC active period and HSYNC active period
//RX_CORE_Register[0x0001]=0x0A29
SPI Write (COMMAND)  0xBF
SPI Write (DATA)     0x01
SPI Write (DATA)     0x00
SPI Write (DATA)     0x29
SPI Write (DATA)     0x0A

//Configure Endian=0, CO=0
//RX_CORE_Register[0x0005]=0x1E00
SPI Write (COMMAND)  0xBF
SPI Write (DATA)     0x05
SPI Write (DATA)     0x00
SPI Write (DATA)     0x00
SPI Write (DATA)     0x1E

//Configuring the output interface [Note 4.3]
//RGB+MCU(CS0)+SPI(CS1),RGB uses 24bpp,
//MCU interface=8bit, DBI Type B(8080)
//(Low Power)
//RX_CORE_Register[0x0004]=0x0207
SPI Write (COMMAND)  0xBF
SPI Write (DATA)     0x04
SPI Write (DATA)     0x00
SPI Write (DATA)     0x07
SPI Write (DATA)     0x02

//Set Command Count = 1 for Smart Display Data
//[Note 4.4]
//RX_CORE_Register[0x0009]=0x0001
SPI Write (COMMAND)  0xBF
SPI Write (DATA)     0x09
SPI Write (DATA)     0x00
SPI Write (DATA)     0x01
SPI Write (DATA)     0x00
```

Note 4.3

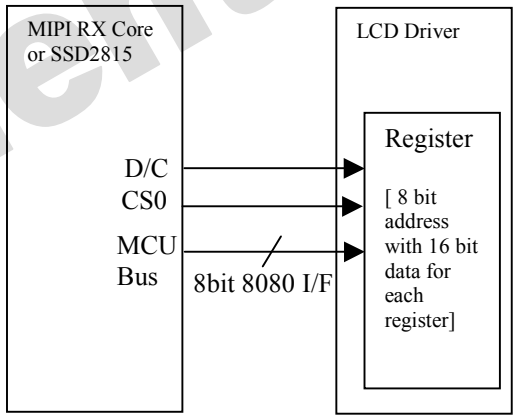


Figure 7. LCD Driver Interface and RX Core

Note 4.4

For LCD driver inside SSD2215, the register address is 8 bit and value is16 bit.

For example,
writing panel driver register [0xC8]=0x0022,
0xC8 0x00 0x22
will be written to Generic Packet Drop register (REG[0xBF]).

By setting Command Count=1,
RX Core will output the first byte with D/C pin =CMD and the following byte with D/C pin =DATA

So the Transmit Data Count (TDC) = 3
Packet Size threshold is the size for partitioning the packet data. Set Packet Size Threshold=3

Step 5. Configure LCD Driver Register

Step 5.1 Setup before sending Generic Write Packet to LCD Driver via MCU Interface (CS0) [Note 5.1]

```
// Set VC2=2
//Note VC2 is for data input from SPI interface,
//i.e. Data from SPI will be route to Virtual Channel "2"
//By default, Virtual Channel 2 is for CS0 of MCU I/F
//Register[0xB8]=0x0049
SPI Write (COMMAND)  0xB8
SPI Write (DATA)     0x49
SPI Write (DATA)     0x00

//Set Low byte of Transmit Data Count (TDC[15:0])
//Register[0xBC]=0x0003
SPI Write (COMMAND)  0xBC
SPI Write (DATA)     0x03
SPI Write (DATA)     0x00

//Set High byte of Transmit Data Count (TDC[31:16])
//Register[0xBD]=0x0000
SPI Write (COMMAND)  0xBD
SPI Write (DATA)     0x00
SPI Write (DATA)     0x00

//Set Packet Size thershold=3,write_reg(0xBE,0x0003)
//Register[0xBE]=0x0003
SPI Write (COMMAND)  0xBE
SPI Write (DATA)     0x03
SPI Write (DATA)     0x00
```

Step 5.2 Writing LCD Driver Register using Generic Write Packet via MCU Interface (CS0)

```
// Set LCD driver register [0xC8] = 0x00 0x22
SPI Write (COMMAND)  0xBF
SPI Write (DATA)     0xC8
SPI Write (DATA)     0x00
SPI Write (DATA)     0x22

// Set LCD driver register [0xB3] = 0x64 0x62
SPI Write (COMMAND)  0xBF
SPI Write (DATA)     0xB3
SPI Write (DATA)     0x64
SPI Write (DATA)     0x62

…
```

(LCD driver register settings varies between different LCD modules. Please refer to the LCD module datasheet.)

Note 5.1

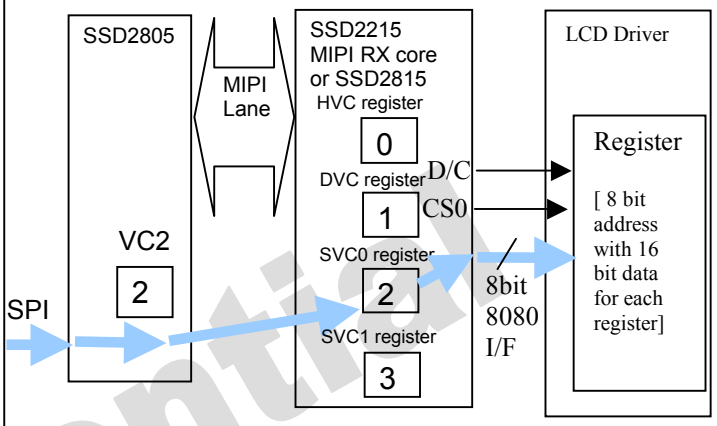


Figure 7. Configure LCD Driver Register

Step 6. Configure SSD2805 RGB interface (using NON-BURST mode Video mode)

Step 6.1 Enable the RGB data output from Host Processor

PCLK will be used as clock source in non-burst mode

Step 6.2 Disable PLL and Change Clock Source [Note 6.1]

```
//set PLL Lock Status Enable(Bit 7)
//in Interrupt Control Register
//Register[0xC5]=0x0080
SPI Write (COMMAND)  0xC5
SPI Write (DATA)     0x80
SPI Write (DATA)     0x00

//Disable PLL
//Register[0xB9]=0x0000
SPI Write (COMMAND)  0xB9
SPI Write (DATA)     0x00
SPI Write (DATA)     0x00

//Change Clock Source as PCLK (for non-burst mode)
//Register[0xB7]=0x0230
SPI Write (COMMAND)  0xB7
SPI Write (DATA)     0x30
SPI Write (DATA)     0x02

//Set PLL multiplier
//Register[0xBA]=0x102F
SPI Write (COMMAND)  0xBA
SPI Write (DATA)     0x2F
SPI Write (DATA)     0x10
```

Step 6.3 Configure RGB timing

```
//Vertical and Horizontal Sync Period, Register[0xB1]=0x0203
SPI Write (COMMAND)  0xB1
SPI Write (DATA)     0x03
SPI Write (DATA)     0x02

//Vertical and Horizontal back porch ,Register[0xB2]=0x021F
SPI Write (COMMAND)  0xB2
SPI Write (DATA)     0x1F
SPI Write (DATA)     0x02

//Vertical and horizontal front porch ,Register[0xB3]=0x0E0A
SPI Write (COMMAND)  0xB3
SPI Write (DATA)     0x0A
SPI Write (DATA)     0x0E

//Horizontal Active Period ,Register[0xB4]=0x00F0
SPI Write (COMMAND)  0xB4
SPI Write (DATA)     0xF0
SPI Write (DATA)     0x00

//Vertical Active Period ,Register[0xB5]=0x0140
SPI Write (COMMAND)  0xB5
SPI Write (DATA)     0x40
SPI Write (DATA)     0x01
```

Note 6.1:
If PLL has not yet locked, the whole system is operating using the PLL input clock.
The SPI serial clock should be 1/8 of this system clock.

For example, 10MHz is input at TX_CLK during power up. The whole system is running at 10MHz. (REG[B7] Bit 5 (**CSS)** is 0 by default). SPI serial clock should be less than 1.25MHz before PLL is locked.

Note 6.2:

With Register[0xB7] Bit 5 Clock Source Select(CSS)=1 (PCLK)

For burst mode,
PLL output clock >=Bit Per Pixel(bpp) x Pixel Clock(PCLK)

For Non-burst mode,
PLL output clock = Bit Per Pixel (bpp) x Pixel Clock (PCLK)

e.g. Set Register[0xBA]=0x102F
PDIV = 0x1 ➔Post Divider=2
DIV = 0x0 ➔Divider = 1
MUL = 0x2F =>47(dec) ➔Multiplier=48
Input clock=PCLK input from Host Controller (e.g. 6MHz)

PLL output Clock
= (PCLK * Multiplier/Divider)/ Post Divider
= (PCLK * 48/1)/2
= PCLK * 24bpp ← required for Non Burst mode
= 144MHz

Constraints for PLL settings:

Input clock / Divider ≥ 5MHz
500 MHz ≥ (Input clock / Divider) *Multiplier ≥ 225MHz

Step 6. Configure SSD2805 RGB interface (using NON-BURST mode Video mode) (continue)

Step 6.4 Configure MIPI Video Mode

```
//Set NON-BURST Mode with Sync Event 24bpp
//Register[0xB6]=0x0067
SPI Write (COMMAND)  0xB6
SPI Write (DATA)     0x67
SPI Write (DATA)     0x00

//Re-enable PLL, Register[0xB9]=0x0001
SPI Write (COMMAND)  0xB9
SPI Write (DATA)     0x01
SPI Write (DATA)     0x00
```

[Check PLL Lock Status as in Step 3]

```
//HS TX Timer1 (MIPI Signal), Register[0xCF]=0x0000
SPI Write (COMMAND)  0xCF
SPI Write (DATA)     0x00
SPI Write (DATA)     0x00

//HS TX Timer2 (MIPI Signal), Register[0xD0]=0x8000
SPI Write (COMMAND)  0xD0
SPI Write (DATA)     0x00
SPI Write (DATA)     0x80

//Set endian and color order(panel driver dependent)
//Register[0xD6]=0x0005
SPI Write (COMMAND)  0xD6
SPI Write (DATA)     0x05
SPI Write (DATA)     0x00

//Set VC1=1, Register[0xB8]=0x0049
SPI Write (COMMAND)  0xB8
SPI Write (DATA)     0x49
SPI Write (DATA)     0x00

//Enable Video Mode (Start Video Data Transmission)
SPI Write (COMMAND)  0xB7
SPI Write (DATA)     0x39
SPI Write (DATA)     0x02
```

Note 6.3

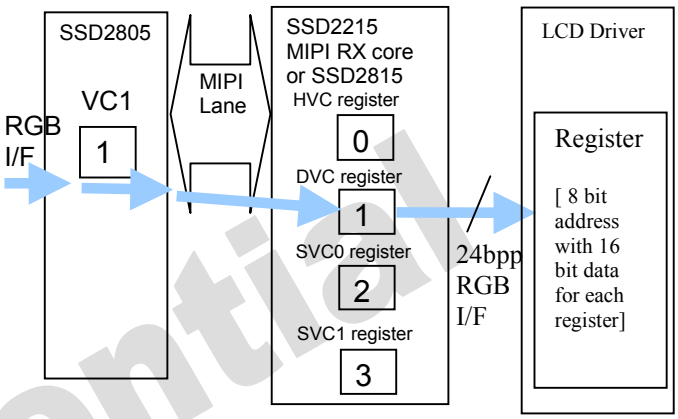


Figure 7. Configure LCD Driver Register

9.3. Software Flow to sleep (Host Processor communicates with SPI 8 bit 3 wire)

Step 1. Send set_display_off DCS command [Note 1.1]
[Turn off display on slave module]

```
// Use DCS packet, Register[0xB7]=0x026B
SPI Write (COMMAND)  0xB7
SPI Write (DATA)     0x6B
SPI Write (DATA)     0x02

//Set Low byte of Transmit Data Count (TDC[15:0])
//Register[0xBC]=0x0000
SPI Write (COMMAND)  0xBC
SPI Write (DATA)     0x00
SPI Write (DATA)     0x00

//Set High byte of Transmit Data Count (TDC[31:16])
//Register[0xBD]=0x0000
SPI Write (COMMAND)  0xBD
SPI Write (DATA)     0x00
SPI Write (DATA)     0x00

//Set Packet Size thershold=2,write_reg(0xBE,0x0002)
//Register[0xBE]=0x0002
SPI Write (COMMAND)  0xBE
SPI Write (DATA)     0x02
SPI Write (DATA)     0x00

//Write DCS command 0x28 (set_display_off) [Note 1.2]
SPI Write (COMMAND)  0x28
```

Wait for 20ms (Depends on MIPI Slave requirement)

Step 2. Send enter_sleep_mode DCS command
[Put Slave into Sleep mode] [Note 1.3]

```
//Write DCS command 0x10 (enter_sleep_mode)
SPI Write (COMMAND)  0x10
```

Wait for 50ms (Depends on MIPI Slave requirement)

Step 3. Disable Video Mode

```
//Disable Video Mode, Register[0xB7]=0x2220
SPI Write (COMMAND)  0xB7
SPI Write (DATA)     0x20
SPI Write (DATA)     0x22
```

Step 4. Disable PLL

```
//Disable Video Mode, Register[0xB9]=0x0000
SPI Write (COMMAND)  0xB9
SPI Write (DATA)     0x00
SPI Write (DATA)     0x00
```

Step 5. Put MIPI System into ULPS mode [enter LP00 state]

```
//Sleep Mode Enable(SLP)=1, Register[0xB7]=0x0224
SPI Write (COMMAND)  0xB7
SPI Write (DATA)     0x24
SPI Write (DATA)     0x02
```

Step 6. Disable the clock source (PCLK or TX_CLK) selected by Clock Source Selected (CSS) of Register [0xB7] and all input signal.

Note 1.1:

The Transmit Data Count for DCS command is the total length of parameter (Command is not included)

For DCS Command 0x28 does not follow by any parameter, the Transmit Data Count =0

Note 1.2:

Address range 0x00~0xAF are for MIPI DCS command
Address range 0xB0~0xD9 is for SSD2805 local register
Address range 0xDA~0xFF is for MIPI DCS command set and expansion

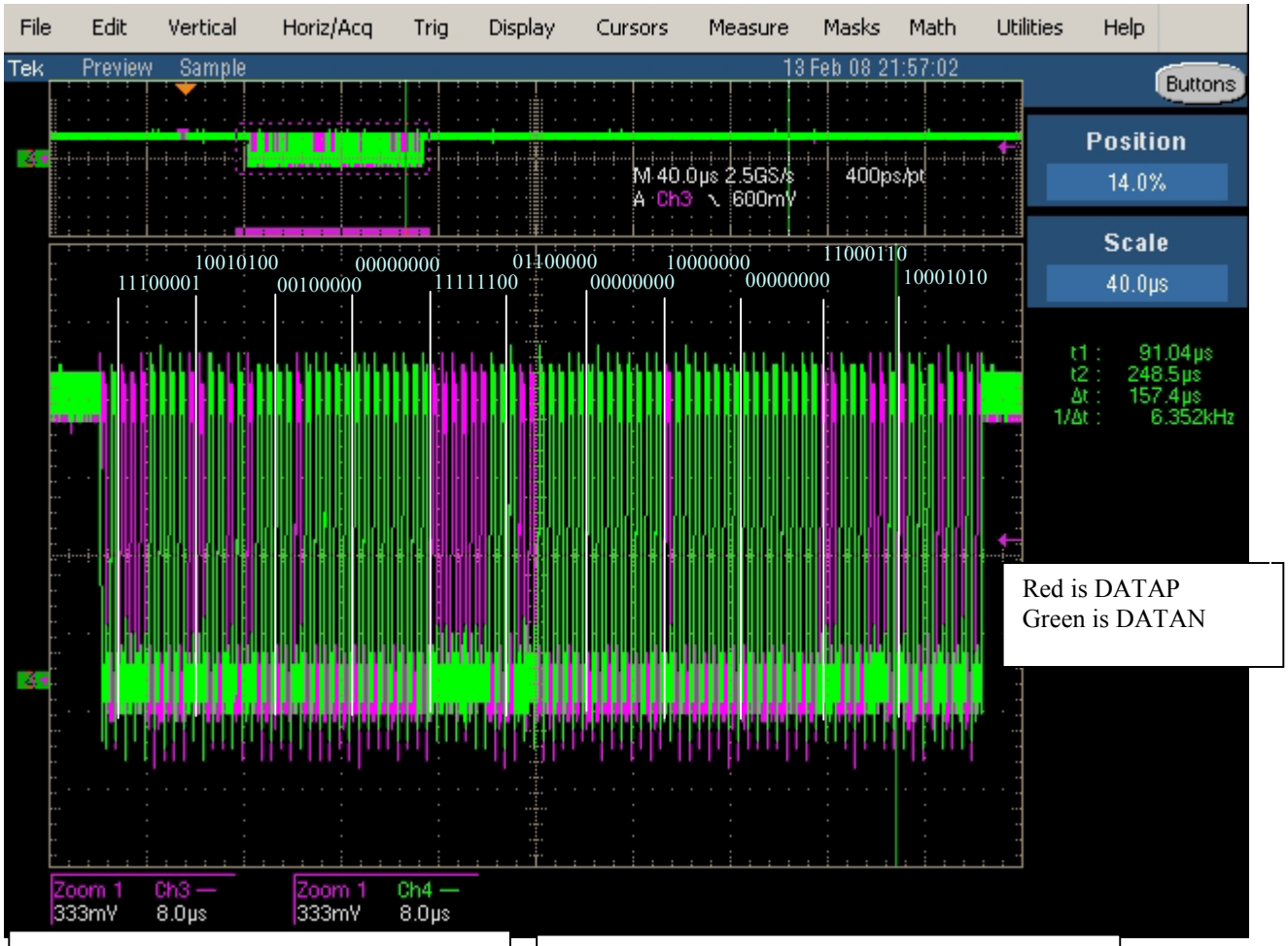
So “SPI Write (Command)” can be used for specifying an SSD2805 local register address or issuing a DCS command.

If command is between 0xB0 and 0xD9, it is determined as a SSD2805 local register address, other command will be send as DCS command

Note 1.3:

If MIPI RX do not support DCS command 0x10 (enter_sleep_mode), a rising edge of SHUT pin can generate a DSI Shut Down Peripheral packet. (The falling edge of SHUT can generate a DSI Turn ON Peripheral packet)

10. Sample MIPI Lane Signal for Generic Write Packet at LP mode
This is to set RX Core Register [0x0006]=0x0001 using Virtual Channel 0



Red is DATAP
Green is DATAN

SSD2805 Sequence
reset
setreg ba 311F //set PLL
setreg b9 0001 //enable PLL
setreg b7 0210 //Set LP mode
setreg bb 0003 //LP clock divider
setreg c5 0002 //set interrupt
setreg b8 0041 //route
setreg bc 0004 //packet size (lo byte) = 4
setreg bd 0000 //packet size (hi byte) = 0
setreg be 0004 //packet thershold = 4

//set RX register[0006]=0001
SPI_WRTE bf COMMAND
SPI_WRITE 06 DATA
SPI_WRITE 00 DATA
SPI_WRITE 01 DATA
SPI_WRITE 00 DATA

Bit Stream Description
NOTE: LSB come first
11100001[0x87] LPDT Command(Escape Mode, Lower Power Data Transmission)

//////////Start of DSI Command
////////////////////////// Generic Long Write Cmd (29h)
10010100[0x29] //0x29 is generic long write, to VC=0
00100000[0x04]// useful data [Lo byte] is 4 byte
00000000[0x00] //useful data [Hi byte] is 4 byte
11111100[0x3F] //ECC

01100000[0x06] //useful data
00000000[0x00] //useful data
10000000[0x01] //useful data
00000000[0x00] //useful data

11000110[0x63] //16 bit checksum
10001010[0x51]

Figure 8: Sample MIPI Lane Signal for Generic Write Packet in LP mode

http://www.solomon-systech.com