FOCUS LCDs
LCDs MADE SIMPLE®
Ph. 480-503-4295 | NOPP@FocusLCDs.com

TFT | OLED | GRAPHIC | CHARACTER | UWVD | SEGMENT | CUSTOM

Application Note FAN4214

Initializing MIPI DSI Displays in Adapted Command Mode

The MIPI DSI display interface has two modes of operation. The mode that will be discussed in this application note is the command mode.

Initializing MIPI DSI Displays in Command Mode

The MIPI DSI display interface has two modes of operation. The mode that will be discussed in this application note is the command mode. The command mode is used when the display has available memory to store image data. The display contains an internal IC that controls memory and display functions. The command mode of the MIPI DSI interface is similar to other display interfaces where control is done through command registers. These registers are defined by the controller to access and mange internal memory.

Figure 1: Focus LCDs E35KB-FW1000-N MIPI DSI Display

The display used in this application is E35KB-FW1000-N. This display is a 3.5" TFT with a one lane MIPI DSI interface. The display has the embedded display driver ILI9488 with 345kB of internal GRAM for display data. The processor used in this application is STM32L4R9AI from ST Microelectronics. This processor has a 2 lane MIPI interface that can support up to 1GB/s clock speed on the MIPI DSI link.

Specifications of this display are reviewed in the table below. For more information on this display, please review the datasheet E35KB-FW1000-N.

Characteristic	Specification	Unit
Display Type	TFT Active Matrix	-
Size	54.46 x 82.94	mm
Interface	1-lane MIPI DSI	-
Controller	ILI9488	-
Mode	Transmissive	-
Color Depth	16.7M	Colors
Resolution	320 x 480	Pixels

Hardware Connection

The connection from the display to the processor is done through two differential data lines and two differential clock lines. The display has a ribbon cable with 20 pins that consist of DSI signals and power lines. The processor used in the application is STM32L4R9AI-Discovery evaluation board. This board supports one or two lane MIPI DSI display communication.

The display is interfaced over the one lane MIPI interface at a clock speed of 500MB/s. The processor is programmed through the micro-USB ST-Link interface. There hardware connections between the board and the display are minimal and the only peripheral in this demonstration will be the display. Other applicable peripherals are available on the board such as a micro-SD port and an RGB camera.

Figure 2: STM32L4R9I-Discovery Microcontroller

The connection to the MIPI DSI link on the evaluation board is through a 60-pin high speed Q Strip Mezzanine connector. The display's FFC ribbon connection must be rerouted from the FPC input to a Q Strip output. The connector used for the conversion from ribbon to Q Strip is part number QTH-030-01-L-D-A from Samtec. The pin output description from the evaluation board can be found in the user manual of the processor.

Figure 3: STM32L4 MIPI DSI Link QStrip Connector

The connection from display FPC connector to Q Strip is routed in the following diagram. The 60-pins from the evaluation board are matched with the display FFC ribbon cable. The display is connected through a 20 pin FFC connector with a 0.5mm pitch.

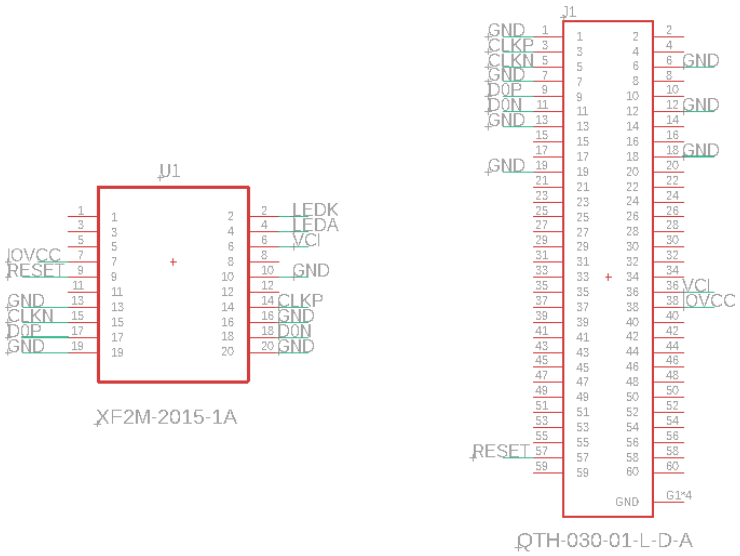

Figure 4: FFC Conversion to Q Strip

The backlight is powered externally from the output available on the evaluation board at 16.6V and 60mA. The unused features of the MIPI DSI connector on the microcontroller can be left open. The unused pins offer features of capacitive touch integration and SPI functions.

The voltage pins, VCI and IOVCC, can be lined together at 3.3V and the high level for the GPIO pins. Alternatively, VCI can be connected to the VCI 3.3V pin output from the controller and IOVCC can be connected to the 1.8V output. In this scenario the IOVCC pin would be the high level for the GPIO pins.

Software Connection

The software used to initialize the display is through the STM Cube IDE. This was chosen because there are pre-defined functions for initializing the processor and memory resources of the board. There is documentation online about interfacing displays through the MIPI DSI port on the ST Microelectronics website.

The example in this application is built off a reference example available from the Cube MX package software. The example is specific to the DSI and the processor STM32L4R9AI. The base example is from the reference “DSI_CmdMode_SingleBuffer” for the STM32L4R9I-Discovery package. This example can be downloaded from CubeMX or Github.

The example integrates different features of the board to operate the display. Building on a pre-written example can be beneficial because you do not have to define the architectures of the MCU. You have access to initialization commands for the lower-level peripherals, memory, and processor. You can utilize the hardware abstraction layer (HAL) functions of the board to integrate these features into the application. The HAL drivers used in this application are found in the “*STM32L4xx_HAL_Driver*” files.

The display initialization process is typically a simple process of sending 8-bit commands and data to the display. This example includes extra features irrelevant to the display initialization but are deeply

integrated into the graphics operations. This adds an extra layer of complexity to a process that would typically be straightforward. The graphics accelerator, memory management and display controller features are interconnected and need to be included to use the DSI communication functions. Once the display is setup, these features can be implemented to optimize and scale the application.

The command mode of the MIPI DSI protocol is used when the display has access to internal RAM available for the framebuffer. There are two versions of the command mode. One is the standard command mode and the other is the adapted command mode. The adapted command mode sends initialization data to format the image that is stored in the frame buffer and is used when the display contains internal RAM for the image data. This application will use adapted command mode.

The adapted command mode is used in the low speed, low power mode of the MIPI protocol. While this is a lower speed than the available high-speed mode, the display is still operating at very high speeds. There is no visible delay in operation when using the low power command mode. The video mode is typically used in applications that do not have available memory.

The MIPI DSI command mode sends data in short and long packets. The HAL API commands for the MIPI DSI interface can send the data in these two modes. Below is a brief description of th short and long packet data functions available in the “stm32l4xx_hal_dsi” driver.

Short Packet Write

```
HAL_DSI_ShortWrite(&DsiHandle, VirtualChannelID, ShortPktDataType, Command, Data);
```

Long Packet Write

```
uint8_t data[4] = {byte1, byte2, byte3, byte4};
HAL_DSI_LongWrite(&DsiHandle, VirtualChannelID, LongPktDataType, #ofData, Command, data[4]);
```

The DSI Handle includes the initialization properties that are compatible with the display interface. These parameters include the number of data lanes, clock speed and mode specific timings. The properties of the DSI Host must be consistent with the properties assigned to the display. The values can be found in the datasheet of the display controller.

The virtual channel ID is used for signaling more than one display. In this case, the value can be set to zero because we are only signaling one display. The data type is based on the mode and the number of data bytes to be sent. In the long packet operation, the data can be stored in array up to 65,534 bytes. The commands are specified in the display controller datasheet.

The MIPI DSI HAL driver takes because of the packet format of the interface. The specifications included in the packet write operations are formatted into the correct MIPI DSI sequence. This includes the data ID, word count, error correction and state changes.

DSI Host Initialization

The DSI MIPI interface must be configured for the interface settings and clock speeds. The interface used in this application is one lane with a clock speed of 500MBps. The display communication interface is initialized by the following commands using the STM32 DSI HAL drivers.

```
/**********************************/
/* DSI Host Initilization Settings */

     DSI_PLLInitTypeDef      dsiPllInit;
     DSI_PHY_TimerTypeDef    PhyTimings;

     __HAL_DSI_RESET_HANDLE_STATE(&DsiHandle);
     DsiHandle.Instance = DSI;
     DsiHandle.Init.AutomaticClockLaneControl =     DSI_AUTO_CLK_LANE_CTRL_DISABLE;
     DsiHandle.Init.TXEscapeCkdiv             =     4;
     DsiHandle.Init.NumberOfLanes             =     DSI_ONE_DATA_LANE;

     dsiPllInit.PLLNDIV =       125;
     dsiPllInit.PLLIDF  =       DSI_PLL_IN_DIV4;
     dsiPllInit.PLLODF  =       DSI_PLL_OUT_DIV1;
     if(HAL_DSI_Init(&DsiHandle, &dsiPllInit) != HAL_OK)
     {
        return(LCD_ERROR);
     }
     PhyTimings.ClockLaneHS2LPTime  = 33;
     PhyTimings.ClockLaneLP2HSTime  = 30;
     PhyTimings.DataLaneHS2LPTime   = 11;
     PhyTimings.DataLaneLP2HSTime   = 21;
     PhyTimings.DataLaneMaxReadTime = 0;
     PhyTimings.StopWaitTime        = 7;
    if(HAL_DSI_ConfigPhyTimer(&DsiHandle, &PhyTimings) != HAL_OK)
      {
        return(LCD_ERROR);
      }
```

The transmitting escape clock is set in the low power command mode. The value of the transmitting clock must be less than 20MHz. Since we want to use the one data lane MIPI interface with a transmission speed of 500Mbps (62.5Mhz) and an escape clock less than 20Mhz, the clock division is set to 4 (TXEscape = 16Mhz).

The physical timings of the host are set to be consistent with the display specified timings. These values specify the time between state changes indicated by the display controller.

DSI Low Power Command Initialization

The command and data packets must be initilized prior to use to indicate the type of data we are sending. Packet size, power mode and RGB signal definitions must be declared so that the DSI HAL driver knows how to format the data correctly. The definition requirements will be different depending on the MIPI DSI communication mode. The definitions used in this application are initilized for the low-power command mode.

```
/*****************************/
/* DSI Command Initilization */

DSI_LPCmdTypeDef LPCmf;
DSI_CmdCfgTypeDef CmdCfg;

      LPCmd.LPGenShortWriteNoP  = DSI_LP_GSW0P_ENABLE;
      LPCmd.LPGenShortWriteOneP = DSI_LP_GSW1P_ENABLE;
      LPCmd.LPGenShortWriteTwoP = DSI_LP_GSW2P_ENABLE;
      LPCmd.LPGenShortReadNoP   = DSI_LP_GSR0P_ENABLE;
      LPCmd.LPGenShortReadOneP  = DSI_LP_GSR1P_ENABLE;
      LPCmd.LPGenShortReadTwoP  = DSI_LP_GSR2P_ENABLE;
      LPCmd.LPGenLongWrite      = DSI_LP_GLW_ENABLE;
      LPCmd.LPDcsShortWriteNoP  = DSI_LP_DSW0P_ENABLE;
      LPCmd.LPDcsShortWriteOneP = DSI_LP_DSW1P_ENABLE;
      LPCmd.LPDcsShortReadNoP   = DSI_LP_DSR0P_ENABLE;
      LPCmd.LPDcsLongWrite      = DSI_LP_DLW_ENABLE;
      LPCmd.LPMaxReadPacket     = DSI_LP_MRDP_DISABLE;
      LPCmd.AcknowledgeRequest  = DSI_ACKNOWLEDGE_DISABLE;
      if(HAL_DSI_ConfigCommand(&DsiHandle, &LPCmd) != HAL_OK)
      {
        return(LCD_ERROR);
      }

      CmdCfg.VirtualChannelID      = 0;
      CmdCfg.ColorCoding           = DSI_RGB888;
      CmdCfg.CommandSize           = 320;
      CmdCfg.TearingEffectSource   = DSI_TE_DSILINK;
      CmdCfg.TearingEffectPolarity = DSI_TE_FALLING_EDGE;
      CmdCfg.HSPolarity            = DSI_HSYNC_ACTIVE_LOW;
      CmdCfg.VSPolarity            = DSI_VSYNC_ACTIVE_LOW;
      CmdCfg.DEPolarity            = DSI_DATA_ENABLE_ACTIVE_HIGH;
      CmdCfg.VSyncPol              = DSI_VSYNC_FALLING;
      CmdCfg.AutomaticRefresh      = DSI_AR_ENABLE;
      CmdCfg.TEAcknowledgeRequest  = DSI_TE_ACKNOWLEDGE_ENABLE;
      if(HAL_DSI_ConfigAdaptedCommandMode(&DsiHandle, &CmdCfg) != HAL_OK)
      {
        return(LCD_ERROR);
      }
```

Display Initialization

It is required to initialize the display by sending commands and data to the display driver. The commands can be found in the datasheet for the IC, ILI9488. The commands are sent over the low power command DCS (Data Command Set) interface and specify settings such as page size, gamma settings and power control. The DCS functions are used in both short and long packets. The values needed to initialize this display are the following.

```
/**********************************/
/* ILI9488 Initilizaiton Commands */

    _HAL_DSI_ENABLE(&DsiHandle);
    uint8_t gammaP[15] = {0x00,0x04,0x0E,0x08,0x17,
                          0x0A,0x40,0x79,0x4D,0x07,
                          0x0E,0x0A,0x1A,0x1D,0x0F};

    uint8_t gammaN[15] = {0x00,0x1B,0x1F,0x02,0x10,
                          0x05,0x32,0x34,0x43,0x02,
                          0x0A,0x09,0x33,0x37,0x0F};

    HAL_DSI_LongWrite(&DsiHandle, 0, DSI_DCS_LONG_PKT_WRITE, 15,0xE0,gammaP);
    HAL_DSI_LongWrite(&DsiHandle, 0, DSI_DCS_LONG_PKT_WRITE,15, 0xE1, gammaN);

    uint8_t PWR[15] = {0x18, 0x17};
    HAL_DSI_LongWrite(&DsiHandle, 0, DSI_DCS_LONG_PKT_WRITE, 2,0xC0,PWR);
    HAL_DSI_ShortWrite(&DsiHandle, 0, DSI_DCS_SHORT_PKT_WRITE_P1, 0xC1, 0x41);

    uint8_t VCOM[3] = {0x00,0x1A,0x80};
    HAL_DSI_LongWrite(&DsiHandle, 0, DSI_DCS_LONG_PKT_WRITE, 3,0xC5,VCOM);
    HAL_DSI_ShortWrite(&DsiHandle, 0, DSI_DCS_SHORT_PKT_WRITE_P1, 0x36, 0x48);
    HAL_DSI_ShortWrite(&DsiHandle, 0, DSI_DCS_SHORT_PKT_WRITE_P1, 0x3A, 0x77);
    HAL_DSI_ShortWrite(&DsiHandle, 0, DSI_DCS_SHORT_PKT_WRITE_P1, 0xB0, 0x00);
    HAL_DSI_ShortWrite(&DsiHandle, 0, DSI_DCS_SHORT_PKT_WRITE_P1, 0xb1, 0xA0);
    HAL_DSI_ShortWrite(&DsiHandle, 0, DSI_DCS_SHORT_PKT_WRITE_P1, 0xb4, 0x02);

    uint8_t DM[2] = {0X02,0X02};
    HAL_DSI_LongWrite(&DsiHandle,0,DSI_DCS_LONG_PKT_WRITE,2,0xB6,DM);
    HAL_DSI_ShortWrite(&DsiHandle, 0, DSI_DCS_SHORT_PKT_WRITE_P1, 0xE9, 0x00);
    uint8_t DSI[4] = {0xa9, 0x51, 0x2c, 0x82};
    HAL_DSI_LongWrite(&DsiHandle, 0, DSI_DCS_LONG_PKT_WRITE,4, 0xF7, DSI);

    uint8_t COL[4] = {0x00, 0x00, 0x01, 0x40};              //320
    uint8_t PAGE[4] = {0x00, 0x00, 0x01, 0xe0};      //480

    HAL_DSI_LongWrite(&DsiHandle,0,DSI_DCS_LONG_PKT_WRITE,4,0x2A,COL);
    HAL_DSI_LongWrite(&DsiHandle,0,DSI_DCS_LONG_PKT_WRITE,4,0x2B,PAGE);
    HAL_DSI_ShortWrite(&DsiHandle, 0, DSI_DCS_SHORT_PKT_WRITE_P0,0x11,0x00);
    HAL_Delay(120);
    HAL_DSI_ShortWrite(&DsiHandle, 0, DSI_DCS_SHORT_PKT_WRITE_P0,0x29,0x00);
    HAL_DSI_ShortWrite(&DsiHandle, 0, DSI_DCS_SHORT_PKT_WRITE_P0, 0x2C, 0x00);

    __HAL_DSI_WRAPPER_ENABLE(&DsiHandle);
```

This example stores image data in the display memory which needs to be written from a memory location on the controller. The graphics memory management library (GFXMMU) formats the page of data in a

virtual memory location linked to a dedicated memory address. Therefore it is necessary to include the additional features outside of the DSI functions.

For a minimal display initialization application, the data can be stored in a number of memory locations without the use of the graphics memory management features. The controllers flash and RAM memory locations require no initialization. This method is not typically used because the frame buffer for the display would take a large portion of the resources available on the MCU.

After the display is sent the initialization commands through the DSI command interface, it can be used in the adapted command or video mode to send image data. The data will be stored in the internal frame buffer of the display. Additional features such as memory management, graphics acceleration and image layering can be incorporated after the display is initialized with the specified commands.

The display in this application has a resolution of 320x480 pixels which is on the lower end for MIPI DSI display resolutions. MIPI DSI operates at high speeds and can support high resolutions and graphics. The MIPI DSI displays with higher resolution do not typcially have a display controller with internal RAM available. If the display does not have internal RAM, then the command mode must be substituted with the video mode operation.

DISCLAIMER

Buyers and others who are developing systems that incorporate Focus LCDs products (collectively, “Designers”) understand and agree that Designers remain responsible for using their independent analysis, evaluation and judgment in designing their applications and that Designers have full and exclusive responsibility to assure the safety of Designers' applications and compliance of their applications (and of all Focus LCDs products used in or for Designers’ applications) with all applicable regulations, laws and other applicable requirements.

Designer represents that, with respect to their applications, Designer has all the necessary expertise to create and implement safeguards that:

(1) anticipate dangerous consequences of failures

(2) monitor failures and their consequences, and

(3) lessen the likelihood of failures that might cause harm and take appropriate actions.

Designer agrees that prior to using or distributing any applications that include Focus LCDs products, Designer will thoroughly test such applications and the functionality of such Focus LCDs products as used in such applications.