



Table of Contents

1 Purpose and Scope	2
2 UART Host Interface	2
2.1 UART Requirements.....	2
2.2 UART Commands.....	3
2.3 UART Read Commands.....	6
2.4 UART Command Builder Tool.....	7
3 MicroSD Card Playback	8
3.1 MicroSD Card Overview.....	8
3.2 MicroSD Card Formatting and Configuration.....	8
3.3 MicroSD Card Playback Operation.....	9
4 Graphics Library	10
4.1 Graphics Library Overview.....	10
4.2 Initial Animations.....	10
4.3 MicroSD Card Images.....	10
4.4 Splash Image.....	10
4.5 Smart Home Demo.....	10
5 DSI Mode	11
5.1 DSI Mode Overview.....	11
6 Revision History	12

Trademarks

All trademarks are the property of their respective owners.

1 Purpose and Scope

This guide document covers the MSPM0G3507 MCU and its interaction with the DLPC3421 chipset that make up the DLPDLR160CPEVM. The MCU serves as a host to the DLPC controller. The guide presents the functions and features that this MCU supports.

2 UART Host Interface

2.1 UART Requirements

The UART digital communication is a controller-peripheral communication link in which the MSPM0 is a peripheral device only. The controller sets when the data transmission begins and ends. The peripheral does not transmit data back to the controller until the controller commands it. A logic 1 value on the UART interface is defined as a recessive value (weak pullup on the RXD pin). A logic 0 value on the UART interface is defined as a dominant value (strong pulldown on the RXD pin).

The UART asynchronous-mode interface is designed for data-rates from 2400bps to 115200bps operation. Other parameters related to the operation of the UART interface include:

- Baud rate from 2.4kbps to 115.2kbps. Default is 9.6kbps
- 8 data bits
- 1 start bit
- 1 stop bit
- No parity bit
- No flow control
- Interfield wait time (only required for 1 stop bit configuration)

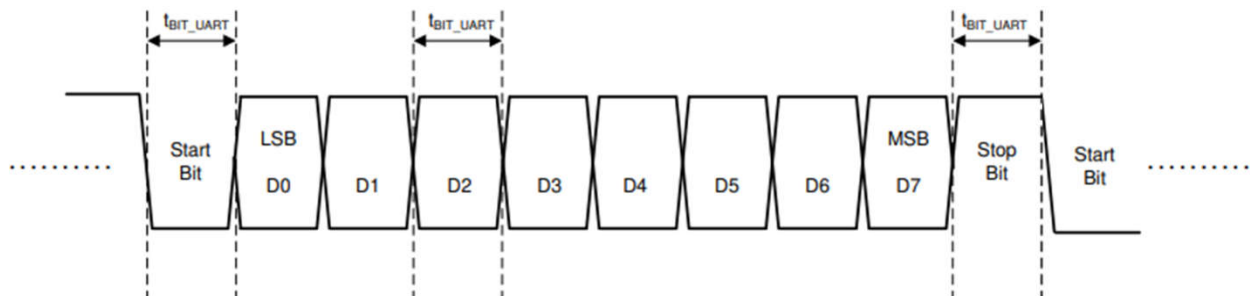


Figure 2-1. UART Asynchronous Interface Bit Timing

Both data and control are in little endian format. Data is transmitted through the UART interface in byte-sized packets. The first bit of the packet field is the start bit (dominant). The next 8 bits of the field are data bits to be processed by the UART receiver. The final bit in the field is the stop bit (recessive). The combined byte of information, and the start and stop bits make up an UART field.

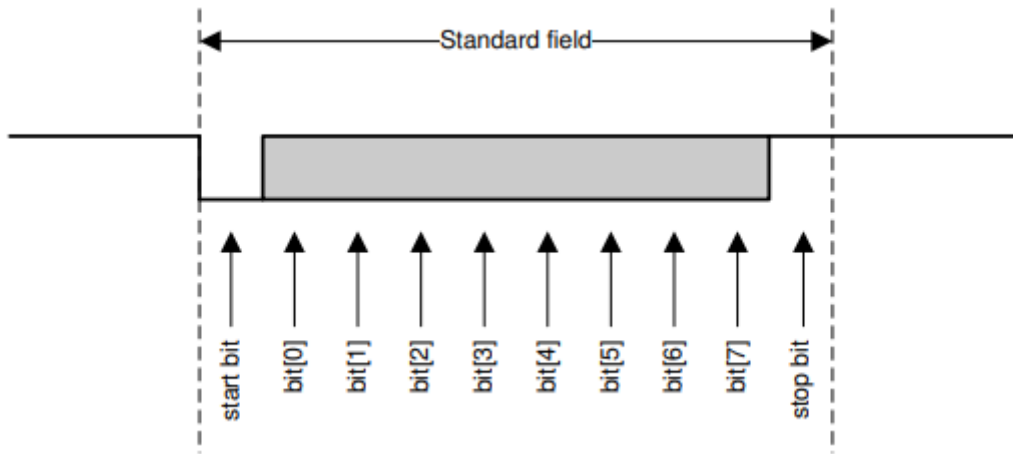


Figure 2-2. UART Interface Packet Field

The [TTL-232R-3V3](#) is a USB to Serial (TTL level) converter cable which allows for a simple way to connect TTL interface units to USB. This version of the FTDI USB to TTL serial adapter cables has the I/O pins configured to operate at 3.3V levels. This is an example USB to Serial device to use during testing.



Figure 2-3. USB to Serial Device

2.2 UART Commands

A group of fields makes up a transmission frame. A transmission frame is composed of the fields required to complete one transmission operation on the UART interface. [Figure 2-4](#) shows the structure of a data transmission operation in a transmission frame.

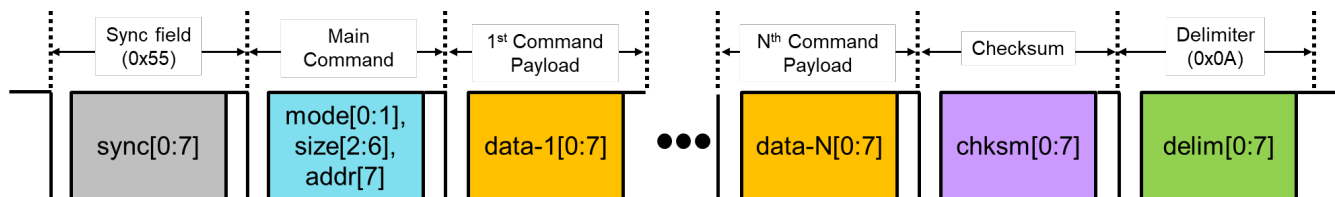


Figure 2-4. UART Interface Transmission Frame

UART events must be interrupt dependent, not polled, such that when the controller sends a UART payload, the MSPM0 enters a UART Interrupt Service Routine (ISR) to process the incoming payload. UART controller total incoming payload:

Byte Number	Byte Field Name	Value	Description
0	Sync	0x55	The sync field is the first field in every frame that is transmitted by the controller. The sync field is used by the MSPM0 device to confirm the correct baud rate of the frame that is sent by the controller.
1	Main Command	Bit0-1: Command Mode Bit2-6: Command Size Bit7: MSPM0 Address	<p>Command Mode:</p> <p>The command field is the second field in every frame sent by the controller. The command field contains instructions about what to do with and where to send the data that is transmitted to a particular MSPM0 device. The command field can also instruct the MSPM0 device to send data back to the controller during a read operation. The number of data fields to be transmitted is also determined by the command in the command field.</p> <p>0x00 = DLPC3421 Command 0x01 = SSD1963 Command 0x02 = MSPM0 Command 0x03 = Reserved / TI Internal Command</p> <p>Command Size:</p> <p>5 bits to indicate the number of write bytes (up to 32 bytes) to expect for the active command following this (Main Command) byte, but excludes the checksum from the count. The MSPM0 expects the given number of bytes to consider the UART controller transfer complete, or timeout to exit the UART receive function if the number of expected bytes is not received within 100ms of the last byte received.</p> <p>MSPM0 Address:</p> <p>In the MSPM0 device, the last bit of the command field is reserved for UART address information. The address information in the command field is compared to the UART_ADDR parameter defined in the MSPM0 source code. Upon receiving the command field, the MSPM0 device checks if the self-address matches the received address. If it matches, the device executes on the received command. If the address does not match, the device disregards the received frame. The default preprogrammed address for the MSPM0 device is 0.</p>
2 ... n	Command Mode Payload	n = up to 32 bytes	(Refer to the individual command mode payload tables below for details.)
2 ... n+1	Checksum	1 Byte	CheckSum8 Modulo 256: Sum of Bytes % 256 The Sync field is not included in the checksum calculation.
1	Delimiter	0X0A	The delimiter byte value of 0x0A is identical to the newline character is ASCII, and represents the end of the complete UART payload.

Command Mode:

0x00 = DLPC3421 Command sub-payload:

Byte Number	Byte Field Name	Value	Description
0	Address (Chip Address)	1 byte	Write = 0x36 or 0x3A Read Request = 0x36 or 0x3A Read Response = 0x37 or 0x3B
1	Read Data Length	1 byte	Max supported respond length is 255 (theoretical), but limit to 29 (actual based on payload structure). This value is "do not care" for a write commands.
2	Sub-Address (Command)	1 byte OpCode	Supports all command types listed in the DLPC3421 Programmer's Guide, which include: <ul style="list-style-type: none"> • General Operation • Illumination Control • Image Processing Control • General Setup • DSI Commands • Administrative • Flash Update
3 ... n	Remaining Data Bytes (Write Parameters)	n = up to 8 or 29 bytes	All commands require 8 or less bytes as write parameters. The maximum number of parameter bytes is limited to 8 based on the <i>Write Image Crop</i> (10h) command. Excludes full support for up to 1024 bytes of the <i>Write Flash Start</i> (E1h) and <i>Write Flash Continue</i> (E2h) commands; these commands are limited to 29 bytes due to the UART controller payload's Main Command size of 32 bytes.

See the [DLPC3421 Software Programmer's Guide](#) for additional details.

0x01 = SSD1963 Command sub-payload:

Byte Number	Byte Field Name	Value	Description
0	Command	1 byte hex code	Supports all commands listed in the SSD1963 LCD Controller data sheet.
1 ... n	Write Parameters	n = up to 9 bytes	All commands require 9 or less bytes as write parameters. The maximum number of parameter bytes is limited to 9 based on the <i>set_dbc_th</i> (D4h) and <i>get_dbc_th</i> (D5h) commands.

See the [SSD1963 LCD Controller data sheet](#) for additional details.

0x02 = MSPM0 Command sub-payload:

Byte Number	Byte Field Name	Value	Description
0-3	MSPM0 32-bit Address	4 bytes	MSPM0 uses a 32-bit (4 byte) address length. Byte 0 is MSB, and Byte 3 is LSB.
4-7	Register Read Data	4 bytes	Each register address stores a 32-bit data length value. Byte 4 is MSB, and Byte 7 is LSB.

See the [MSPM0G3507](#) data sheet for additional details.

0x03 = TI Internal Command sub-payload:

Byte Number	Byte Field Name	Value	Description
0 ... n	Reserved	n = up to 32 bytes	TI Internal functionality

2.3 UART Read Commands

If the MSPM0 is to return data to the external host controller (PC GUI), the UART peripheral total outgoing payload must match the incoming payload format. Data that is exclusive to the incoming payload is echoed back to controller. The PC GUI is programmed when to expect read/return data and sends null (0xFF) data to the peripheral while data is returned to the controller.

UART peripheral total outgoing payload:

Byte Number	Byte Field Name	Value	Description
0	Sync	0x55	The sync field is the first field in every frame that is transmitted by the controller. The sync field is used by the MSPM0 device to confirm the correct baud rate of the frame that is sent by the controller.
1	Main Command (Controller Echo)	1 byte	Refer to incoming payload description.
2 ... n	Command Mode Payload	n = up to 32 bytes	Refer to the individual command mode payload tables below for details.
2 ... n+1	Checksum	1 Byte	Checksum8 Modulo 256: Sum of Bytes % 256 The Sync field is not included in the checksum calculation.
1	Delimiter	0X0A	The delimiter byte value of 0x0A is identical to the newline character is ASCII, and represents the end of the complete UART payload.

Command Mode:

0x00 = DLPC3421 Command sub-payload:

Byte Number	Byte Field Name	Value	Description
0	Address (Chip Address) (Controller Echo)	1 byte	Write = 0x36 or 0x3A Read Request = 0x36 or 0x3A Read Response = 0x37 or 0x3B
1	Read Data Length (Controller Echo)	1 byte	Maximum supported respond length is 255 (theoretical), but limit to 29 (actual based on payload structure). This value is "do not care" for a write commands.
2	Sub-Address (Command) (Controller Echo)	1 byte	Supports all command types listed in the DLPC3421 Software Programmer's Guide , which include: <ul style="list-style-type: none"> • General Operation • Illumination Control • Image Processing Control • General Setup • DSI Commands • Administrative • Flash Update
3 ... n	Read Parameters	n = up to 29 bytes	The maximum number of return parameters to be expected is 29 bytes based on the command Read Sequence Header Attributes (26h). Optionally, consider full support for up to 256 read bytes of the Read Flash Start (E3h) and Read Flash Continue (E4h) commands; these commands are limited to 29 bytes due to the UART controller payload's Main Command size of 32 bytes.

See [DLPC3421 Software Programmer's Guide](#) for additional details.

0x02 = MSPM0 Command sub-payload:

Byte Number	Byte Field Name	Value	Description
0	Command (Controller Echo)	1 byte Hex Code	Refer to incoming payload description.
1 ... n	Read Parameters	n = up to 9 bytes	All commands require 9 or less bytes as read parameters. The maximum number of parameter bytes is limited to 9 based on the set_dbc_th (D4h) and get_dbc_th (D5h) commands.

See [SSD1963 LCD Controller data sheet](#) for additional details.

0x02 = MSPM0 Command sub-payload:

Byte Number	Byte Field Name	Value	Description
0-3	MSPM0 32-bit Address (Controller Echo)	4 bytes	MSPM0 uses a 32-bit (4 byte) address length. Byte 0 is MSB, and Byte 3 is LSB.
4-7	Register Read Data	4 bytes	Each register address stores a 32-bit data length value. Byte 4 is MSB, and Byte 7 is LSB.

See [MSPM0G3507 data sheet](#) for additional details.

0x03 = TI Internal Command sub-payload:

Byte Number	Byte Field Name	Value	Description
0 ... n	Reserved	n = up to 32 bytes	TI Internal functionality

2.4 UART Command Builder Tool

Use the *DLPDLCR160CPEVM_Command_Builder.xlsxm* to build the low-level UART command payload for the host controller.

The blue cells are user inputs, while the yellow cells are fixed or auto-generated.

	A	B	C	D	E	F	G	H	I	J	K
1			Main Command					command mode payload			
2			bit 0-1 : Command mode		bit 2-6 : Command size	bit 7 : MSPM0 Address					
3			Select value	-							
4	Sync		0	DLPC3421 Command							
5			1	SSD1963 Command							
6			2	MSPM0 Command							
7			3	Reserved / TI Internal Command							
8			0		4	0		37 01 0C 37			
9	55		10							8B	0A
10											
11											
12	Entire UART command to MSPM0										
13	55 10 37 01 0C 37 8B 0A										
14											
15											
16											
17											
18											
19			User input								
20											
21			Auto calculation								
22			(do not change)								
23											
24											

3 MicroSD Card Playback

3.1 MicroSD Card Overview

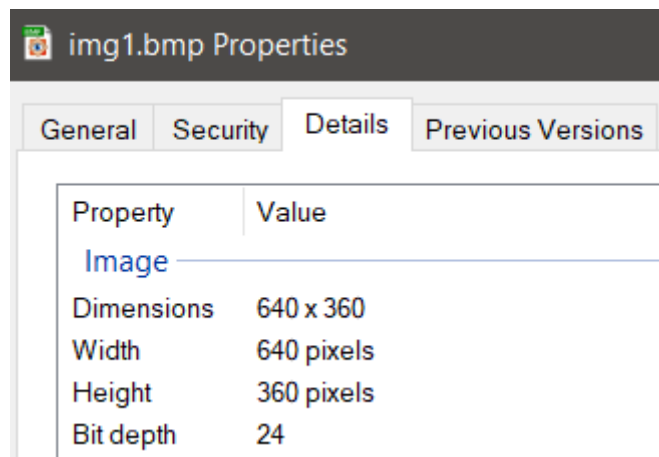
The EVM is designed to playback image files from a microSD card. The playback speed from the microSD card is approximately 1 frame per second.

3.2 MicroSD Card Formatting and Configuration

MicroSD cards shall be formatted to FAT32 to support SDHC capacities up to 32GB. Larger memory density SD cards must be formatted from exFAT to FAT32.

Each image must be a 24-bit bitmap (.BMP) file with an nHD resolution of 640×360. PNG, JPG, and other image file types are not supported by this EVM, and must be converted to BMP files before being loaded onto the microSD card. Each resulting .BMP file must be at least 691 200 bytes in size and not use any form of compression.

Example of the image properties for a compatible BMP image file:



A text file named “conf.txt” is required in the MicroSD card. The conf.txt file specifies a set of parameters for each image, such as the image file name, draw coordinates, background color, and time to display. Do not change the name of the conf.txt file because the MSPM0 specially looks for the conf.txt file at startup. If this file is not detected, or if the contents of the conf.txt file are erroneous, microSD card playback fails.

Details of the Configuration (conf.txt) file parameters:

Keyword	Parameter	Comments
boot_up_mode	1	1: Splash image 2: SSD (RGB_parallel) 3: DSI
number_of_sdcard_images	>=1	The number of maximum supported images is a function of the 512 character array divided by the length of the entire image_names row of characters. Must be at least '1' or greater.
image_names	image1_name.bmp, image2_name.bmp, ...	Image names must include the “.bmp” extension.
sequence_repeat	1	Must be a value of '1'
number_of_sequence	3 × number_of_sdcard_images	Indicator for the number of the sequence. Each image requires 3 sequences, thus number_of_sequence is determined by #_of_images multiplied by 3.
rgb_fill	R,G,B	Each R-G-B value can accept an 8-bit value to specific the background fill color of display before drawing image from SD card.





Keyword	Parameter	Comments
image	image_index, x_location, y_location	image_index is the element to load from the image_nams list. x/y_location refers to the coordinates to start drawing the bitmap image (typically 0,0)
delay	1000	Delay to show the image in milliseconds from the time the last pixel of the SD card image is displayed.
rgb_fill	R,G,B	
image	image_index, x_location, y_location	
delay	1000	

This is an example of the conf.txt file contents for a four image example:

```
boot_up_mode 2
number_of_sdcard_images 4
image_names img1.bmp,img2.bmp,img3.bmp,img4.bmp
sequence_repeat 1
number_of_sequence 12
rgb_fill 0,0,0
image 1,0,0
delay 1024
rgb_fill 0,0,0
image 2,0,0
delay 1024
rgb_fill 0,0,0
image 3,0,0
delay 1024
rgb_fill 0,0,0
image 4,0,0
delay 1024
```

Given the conf.txt file variables are case-sensitive, all inputs to the conf.txt file must in lower case.

The folder structure requires that the conf.txt file and the accessed imgX.bmp files are stored in the root folder of the microSD card. Here is an example of the folder structure for a four-image example:

Name	Date	Type	Size
<input checked="" type="checkbox"/> conf.txt	11/27/2023 11:50 PM	Text Document	1 KB
 img1.bmp	10/25/2023 2:22 AM	FastStone BMP File	676 KB
 img2.bmp	10/25/2023 2:22 AM	FastStone BMP File	676 KB
 img3.bmp	10/25/2023 2:22 AM	FastStone BMP File	676 KB
 img4.bmp	10/25/2023 2:23 AM	FastStone BMP File	676 KB

3.3 MicroSD Card Playback Operation

Follow these steps to enable playback from the SD card:

1. Remove power from EVM.
2. Insert the SD Card into the EVM.
3. Plug-in the EVM to a USB-C power supply.
4. Toggle the PROJ_ON switch to the on-state.

5. On the display menu, navigate to and select the microSD card option using the physical push button. Note: Click the push-button to toggle through the various menu options. Push and hold the push-button for at least 2 seconds to select an operating mode.
6. To escape the SD card playback mode before the last image is displayed, click the push-button.

4 Graphics Library

4.1 Graphics Library Overview

The MSPM0G3507 supports the use of a graphics library provided by Rinky-Dink Electronics. This allows the EVM to project images created by the MCU.

4.2 Initial Animations

When the DLPC3421 successfully powers up and begins displaying content, a scripted demo of content is displayed. The demo includes a still image, an animation, and a menu. Two of these elements use a graphics library that this section describes.

Function **StateplayLogo()** calls function **PlayLogo()**, which creates an animation and then reveals an image. Next, the main menu GUI is created with function **SetupMainMenu()**. This function defines what text shown and contains functions that define visually how the menu is created. This includes shapes and colors.

4.3 MicroSD Card Images

As [Section 3](#) describes, this EVM supports displaying SD card images. Select the microSD card option by long pressing the push button on the main menu GUI. By long pressing the push button, case statement **St_MMENUSDCARD** is entered and starts setting up the EVM to display images from the microSD card. The images are read and displayed one at a time until all of the specified images are shown. When this has been completed, a message provides further instructions.

The microSD card must be formatted exactly as specified in [Section 3](#) for this feature to operate correctly.

4.4 Splash Image

The MSPM0 demo supports the ability to display the splash images that are stored inside of the DLPC3421 controller on the EVM.

Starting from the main menu GUI, the user can select the flash image option. Once a long press is registered the MSPM0 will enter statement **St_MMENUFLASH**. This case statement will create a button select animation, change input source select, and display the splash image . The function will iteratively select the splash images and display them to the user. When this process is finished there will be a message with instructions on the display. Follow those to get back to the main menu.

4.5 Smart Home Demo

The Smart Home GUI demo is created solely by the MSPM0 and the graphics library. No images are being used from MicroSD card or FW images.

Selecting Smart Home demo on the main menu calls case statement **St_MMENUSH** and sets **StateNow = St_DEMO**. This calls the function **ExecMain()** and calls the **StateDemo()** function. The state demo function calls multiple graphics-related functions that display in the demo. The functions are listed below:

- **SetupMenus()**: Creates icons at the right-hand side of the display
- **SetupAlbums()**: Creates the icon at the top right-hand side of the display
- **SetupWeather()**: Creates the weather display at the bottom left of the display
- **UpdateClock()**: Creates the clock visual at the upper left side of the display
- **WeatherNow()**: Creates the weather display at the bottom left of the display

On the right side of the display, there are four icons that the user can select with the EVM push button. The icons consists of an exclamation point, check mark, calendar, and house icon. Using the push button to select the exclamation point icon on the right calls the function **MessagesNow()**. This functions creates an image with text. Using the push button to select the check mark calls the function **ToDoNow()**. This function generates a dynamic check list of items. Using the push button to select the calendar calls the function **CalendarNow()**. This function

generates a generic calendar image. Using the push button to select the house icon exits the smart home demo and returns to the main menu.

The animation and colors for the icons when switching between them or selecting them are controlled with the following functions.

Moving between and selecting icons:

- SelectHome(int move_button)
- SelectAttention(int move_button)
- SelectToDo(int move_button)
- SelectCalendar(int move_button)

5 DSI Mode

5.1 DSI Mode Overview

Display Serial Interface (DSI) is a video interface that is supported by the DLPC3421 controller on the DLPDL160CPEVM. This is an alternative to the parallel interface that the controller is using with the SSD1963 and MSPM0G3507. This method allows the EVM to keep its smaller size, because fewer pins are used, and have the ability to display external video content on the EVM. The EVM supports up to 4 DSI data lanes. The EVM is configured for 4 lanes out of the box.

For more information, see the [DLPC3421 Display Controller](#) data sheet. Review the following sections in the data sheet: DSI Input Data and Clock table, GPIO Peripheral Interface table, and DSI Host Timing Requirements.

The [DSI Setup and Debugging Guide](#) post gives is a more detailed DSI setup guide. This controller has some different commands from the guide mentioned that must be executed to allow for use of DSI. These commands can either be added to the Autoinit batch file or can be sent to the unit after the controller is initialized. The following is an example of the commands to send. The command order shown is not the only way that these commands can be sent. See the [DLPC3421 Software Programmer's Guide](#) for more detail on these commands.

DsiPortEnable (D7h)

ImageCrop (10h)

InputImageSize (2Eh)

DsiHsClockInput (BDh)

InputSourceSelect (05h)

ExternalVideoSourceFormat (07h)

$$\text{DSI HS Clock} = \frac{\text{Frame Rate} \times V_{\text{Total}} \times H_{\text{Total}} \times \text{bus width}}{\# \text{ of lanes} \times 2 \times 1 \times 10^6} \text{MHz} \quad (1)$$

- $V_{\text{Total}} = V_{\text{Active}} + V_{\text{FrontPorch}} + V_{\text{BackPorch}} + V_{\text{Sync}}$
- $H_{\text{Total}} = H_{\text{Active}} + H_{\text{FrontPorch}} + H_{\text{BackPorch}} + H_{\text{Sync}}$
- Bus width:
 - RGB888 = 24
 - RGB666 = 18
 - RGB565 = 16

1. Connect the EVM to power with the power switch in the off position (HOST_IRQ High).
2. Connect the DSI frontend to the EVM DSI connector.
3. Configure the DSI data lanes in LP11 mode. Do not send data from the frontend to the EVM.
4. Move the power switch to the ON position and wait for the controller to finish initialization (HOST_IRQ Low).
5. Send the commands mentioned above to enable DSI on the EVM and prepare it for the signals.
6. Send the video data from the DSI machine to the EVM.

Note

The information in this document is not a substitute for the specifications listed in the corresponding device data sheets. In the event of a discrepancy, the data sheet supersedes this document.

6 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

DATE	REVISION	NOTES
April 2024	*	Initial Release

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#) or other applicable terms available either on [ti.com](https://www.ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2024, Texas Instruments Incorporated