*Programmer's Guide*

# *DLPC1438 Software Programmer's Guide*

## ABSTRACT

This document details the software interface requirements for a DLPC143x DLP® Pico™ 3D Print based system. The DLPC143x DLP Pico 3D Print Controllers support 3D printing applications with the use of a DLP300S or DLP301S DMD chip.

## Table of Contents



## List of Figures



## List of Tables

## Trademarks

Pico™ is a trademark of Texas Instruments.

DLP® is a registered trademark of Texas Instruments.

All trademarks are the property of their respective owners.

# 1 Introduction

The DLPC143x has three modes for use in 3D Print applications:

- Test Pattern Generator Mode for troubleshooting
- Splash Pattern Mode
- External Print Mode
- Standby Mode

External Print Mode displays pattern data that is stored on or streamed through the FPGA device or directly to the DLPC parallel data pins.

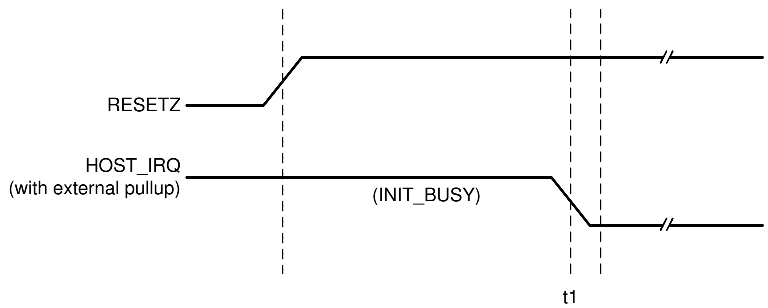
Splash Pattern Mode displays 2D patterns stored as images in flash.

# 2 System Overview

## 2.1 Initialization

The DLPC143x employs a boot ROM and associated boot software. This resident boot code consists of the minimum code necessary to load the software from flash to internal RAM for execution. For most DLPC143x product configurations, an external flash device can store the main application code, along with the other configuration and operational data required by the system for normal operation.

The HOST_IRQ signal provides the completion status of the DLPC143x system initialization when the system is powered on. Once PROJ_ON is high, HOST_IRQ is tri-stated with an external pull-up. After RESETZ is applied, the controller drives HOST_IRQ high while initializing and then drives it low once initialization has completed. The initialization period is determined by the boot configuration and can vary from one system to another.




t1: the first falling edge of HOST_IRQ indicates auto-initialization completion

**Figure 2-1. HOST_IRQ Timing Diagram**

**Note**

Make sure that I²C access to the DLPC143x does not start until HOST_IRQ goes low. Sending an I²C command while HOST_IRQ is high can prevent the system from booting.

## 2.2 I²C Interface Specification

The protocol used in communicating information to DLPC143x consist of a serial data bus conforming to the Philips I²C specification, up to 100 kHz. Commands are executed using I²C, where the DLPC143x behaves as a target.

The supported I²C transaction type for both writes and reads is shown in Table 2-1. The I²C interface supports variable-size transactions (i.e. variable number of bytes as parameters) depending on the command. The list of supported commands are discussed in the next section.

**Table 2-1. I²C Write and Read Transactions**

| Transaction | Address (One byte)[1] | Sub-Address (One byte)[2] | Remaining Data Bytes[3] |
|---|---|---|---|
| Write or Read Request | 36h (or 3Ah) | Command Opcode | Parameter values (0 → N bytes) |
| Read Response | 37h (or 3Bh) | | |

(1) The address corresponds to the chip address of the controller. Default address is 36h, contact TI if alternate address is required.
(2) The subaddress corresponds to a command.
(3) The data (if present) corresponds to any required command parameters.

The standard parameter byte format is shown below:

| msb | Parameter Byte | | | | | | lsb |
|---|---|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

# 3 System Write/Read Commands

## Overview

**Table 3-1. List of System Write/Read Software Commands**

| Command Type | Command Description | OpCode (hex) | Reference |
|---|---|---|---|
| ***General Operation Commands*** | | | |
| Write | Write Operating Mode Select | 05 | Section 3.1.1 |
| Read | Read Operating Mode Select | 06 | Section 3.1.2 |
| Write | Write Test Pattern Select | 0B | Section 3.1.3 |
| Read | Read Test Pattern Select | 0C | Section 3.1.4 |
| Write | Write Splash Screen Select | 0D | Section 3.1.5 |
| Read | Read Splash Screen Select | 0E | Section 3.1.6 |
| Read | Read Splash Screen Header | 0F | Section 3.1.7 |
| Write | Write Display Image Orientation | 14 | Section 3.1.8 |
| Read | Read Display Image Orientation | 15 | Section 3.1.9 |
| Write | Write Display Image Curtain | 16 | Section 3.1.10 |
| Read | Read Display Image Curtain | 17 | Section 3.1.11 |
| Write | Write Image Freeze | 1A | Section 3.1.12 |
| Read | Read Image Freeze | 1B | Section 3.1.13 |
| Write | Write Execute Batch File | 2D | Section 3.1.14 |
| Write | Write Splash Screen Execute | 35 | Section 3.1.15 |
| ***Illumination Control Commands*** | | | |
| Write | Write Illuminator LED Enable | 52 | Section 3.2.1 |
| Read | Read Illuminator LED Enable | 53 | Section 3.2.2 |
| Write | Write Illuminator LED Current PWM | 54 | Section 3.2.3 |
| Read | Read Illuminator LED Current PWM | 55 | Section 3.2.4 |
| Write | Write Illuminator LED Max Current PWM | 5C | Section 3.2.5 |
| Read | Read Illuminator LED Max Current PWM | 5D | Section 3.2.6 |
| ***3D Print Commands*** | | | |
| Write | Write Trigger Out Configuration | 92 | Section 3.3.3 |
| Read | Read Trigger Out Configuration | 93 | Section 3.3.4 |
| Read | Read Light Control Sequence Version | 9B | Section 3.3.5 |
| Write | Write External Print Configuration | A8 | Section 3.3.6 |
| Read | Read External Print Configuration | A9 | Section 3.3.7 |
| Write | Write External Print Control | C1 | Section 3.3.8 |
| Read | Read External Print Control | C2 | Section 3.3.9 |
| Write | Write Parallel Video | C3 | Section 3.3.10 |
| Read | Read Parallel Video | C4 | Section 3.3.11 |
| Write | Write Active Buffer | C5 | Section 3.3.12 |
| Read | Read Active Buffer | C6 | Section 3.3.13 |
| Write | Write FPGA Control | CA | Section 3.3.14 |
| Read | Read FPGA Control | CB | Section 3.3.15 |
| Read | Read FPGA SPI CRC16 | CE | Section 3.3.16 |
| ***Administrative Commands*** | | | |
| Read | Read Short Status | D0 | Section 3.4.1.1 |
| Read | Read System Status | D1 | Section 3.4.1.2 |
| Read | Read System Software Version | D2 | Section 3.4.1.3 |
| Read | Read Communication Status | D3 | Section 3.4.1.4 |

**Table 3-1. List of System Write/Read Software Commands (continued)**

| Command Type | Command Description | OpCode (hex) | Reference |
|---|---|---|---|
| Read | Read Controller Device ID | D4 | Section 3.4.1.5 |
| Read | Read DMD Device ID | D5 | Section 3.4.1.6 |
| Read | Read System Temperature | D6 | Section 3.4.1.7 |
| Read | Read Flash Build Version | D9 | Section 3.4.1.8 |
| Write | Write Flash Batch File Delay | DB | Section 3.4.1.9 |
| ***FPGA Commands*** | | | |
| Read | Read FPGA Version | 64 | Section 3.5.1 |
| Read | Read FPGA Status | 6F | Section 3.5.2 |
| Write | Write FPGA TPG | 67 | Section 3.5.3 |
| Read | Read FPGA TPG | 68 | Section 3.5.4 |
| ***Actuator Commands*** | | | |
| Write | Write Actuator Latency | 70 | Section 3.5.5.1 |
| Read | Read Actuator Latency | 71 | Section 3.5.5.2 |
| Write | Write Actuator Gain | 72 | Section 3.5.5.3 |
| Read | Read Actuator Gain | 73 | Section 3.5.5.4 |
| Write | Write Actuator Segment Length | 74 | Section 3.5.5.5 |
| Read | Read Actuator Segment Length | 75 | Section 3.5.5.6 |
| Write | Write Actuator Subframe Delay | 76 | Section 3.5.5.7 |
| Read | Read Actuator Subframe Delay | 77 | Section 3.5.5.8 |
| Write | Write Actuator Offset | 78 | Section 3.5.5.9 |
| Read | Read Actuator Offset | 79 | Section 3.5.5.10 |
| Write | Write Actuator Configuration Select | A2 | Section 3.5.5.11 |
| Read | Read Actuator Configuration Select | A3 | Section 3.5.5.12 |
| Write | Write Actuator Fixed Output Level | A4 | Section 3.5.5.13 |
| Read | Read Actuator Fixed Output Level | A5 | Section 3.5.5.14 |
| Write | Write Actuator Number of Segments | A6 | Section 3.5.5.15 |
| Read | Read Actuator Number of Segments | A7 | Section 3.5.5.16 |
| Write | Write Actuator Output Select | AA | Section 3.5.5.17 |
| Read | Read Actuator Output Select | AB | Section 3.5.5.18 |
| Read | Read Actuator Control | AF | Section 3.5.5.19 |
| Read | Read Actuator Temperature | C7 | Section 3.5.5.20 |
| Write | Write Actuator Orientation | C8 | Section 3.5.5.21 |
| Read | Read Actuator Orientation | C9 | Section 3.5.5.22 |
| ***Flash Update Commands*** | | | Section 3.6 |
| Read | Read Flash Update PreCheck | DDh | Section 3.6.1 |
| Write | Write Flash Data Type Select | DEh | Section 3.6.2 |
| Write | Write Flash Data Length | DFh | Section 3.6.3 |
| Write | Write Erase Flash Data | E0h | Section 3.6.4 |
| Write | Write Flash Start | E1h | Section 3.6.5 |
| Write | Write Flash Continue | E2h | Section 3.6.6 |
| Read | Read Flash Start | E3h | Section 3.6.7 |
| Read | Read Flash Continue | E4h | Section 3.6.8 |

The following sections describe each of the above listed commands in detail.

## 3.1 General Operation Commands

### 3.1.1 Write Operating Mode Select (05h)

This command selects the operating mode of the system.

#### *3.1.1.1 Write Parameters*

| Byte 1 | Operating Mode |
|---|---|
| 00h | *Reserved* |
| 01h | Display - Test Pattern Generator Mode |
| 02h | Display - Splash Screen Mode |
| 03h - 05h | *Reserved* |
| 06h | 3D Print - External Print Mode |
| 07h - FEh | *Reserved* |
| FFh | Standby Mode |

The Standby mode disables illumination power and sets the DMD in a 50-50 refresh duty cycle, where the mirrors are on 50% of the time and off during the remaining time. This 50-50 refresh state helps in prolonging the life of the DMD.

The other operating modes have associated commands which are only applicable to that mode and must be run to properly configure the selected mode. The associated commands are listed below:

- Display - Test Pattern Generator:
  - *Write Test Pattern Select (0Bh)* - Section 3.1.3
- Display - Splash Screen:
  - *Write Splash Screen Select (0Dh)* - Section 3.1.5
  - *Write Splash Screen Execute (35h)* - Section 3.1.15
- Display - External Print Mode
  - Follow the procedure detailed in Section 3.3.1 for configurations without an FPGA or Section 3.3.2 for configurations with an FPGA to ensure proper use of this mode.

It is recommended that the source associated commands be sent prior to sending the *Write Operating Mode Select* command. These commands (except for *Write Splash Screen Execute*) describe the unique characteristics of their associated source, and once these settings have been defined, they are stored in a volatile manner. When source associated commands are sent when that source is not active, the controller software saves the new settings, but does not execute these commands. When that source becomes active (via the *Write Operating Mode Select* command), the controller applies these settings. Each time an operating mode selection is made , the system retrieves the settings defined previously and automatically applies them. As such, the user only needs to send these associated commands when the source first needs to be defined, or when the source characteristics for that port need to be changed. It is important to note that the appropriate associated commands must be updated when source characteristics change.

Refer to *Write Image Freeze* (Section 3.1.12.1) for information on hiding on-screen artifacts when selecting an input source.

### 3.1.2 Read Operating Mode Select (06h)

This command reads the operating mode of the system.

#### *3.1.2.1 Read Parameters*

This command has no parameters.

#### *3.1.2.2 Return Parameters*

| Byte 1 | Operating Mode |
|---|---|
| 00h | *Reserved* |
| 01h | Display - Test Pattern Generator Mode |
| 02h | Display - Splash Screen Mode |

| Byte 1 | Operating Mode |
|---|---|
| 03h - 05h | *Reserved* |
| 06h | 3D Print - External Print Mode |
| 07h - FEh | *Reserved* |
| FFh | Standby Mode |

1. This command works in all operating modes and has no effect on the current system configuration.
2. Any command issued after this command will need to wait on System_Ready.

### 3.1.3 Write Test Pattern Select (0Bh)

This command specifies an internal test pattern for display on the display module.

#### *3.1.3.1 Write Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Test Pattern Generator (TPG) pattern select |
| Byte 2 | *Reserved* |
| Byte 3 | Parameter 1 (see Table 3-3) |
| Byte 4 | Parameter 2 (see Table 3-3) |
| Byte 5 | Parameter 3 (see Table 3-3) |
| Byte 6 | Parameter 4 (see Table 3-3) |

**Byte 1 TPG pattern select**

b(7) Test pattern border:

- 00h: Disabled
- 01h: Enabled

b(6:4) Reserved

b(3:0) Pattern select:

- 00h: Solid field
- 01h: Fixed step horizontal ramp
- 02h: Fixed step vertical ramp
- 03h: Horizontal lines
- 04h: Diagonal lines
- 05h: Vertical lines
- 06h: Horizontal and vertical grid
- 07h: Checkerboard
- 08h: Color bars
- 09h-0Fh: Reserved

**Table 3-2. Foreground and Background Color Use**

| Pattern | Byte 2 | |
|---|---|---|
| | Foreground Color | Background Color |
| Solid field | Yes | No |
| Fixed step horizontal ramp | Yes | No |
| Fixed step vertical ramp | Yes | No |
| Horizontal lines | Yes | Yes |
| Vertical lines | Yes | Yes |
| Diagonal lines | Yes | Yes |
| Grid lines | Yes | Yes |
| Checkerboard | Yes | Yes |

**Table 3-3. Descriptions and Bit Assignments for Parameters 1-4**

| Pattern | Byte 6 (Parameter 4) | | Byte 5 (Parameter 3) | | Byte 4 (Parameter 2) | | Byte 3 (Parameter 1) | |
|---|---|---|---|---|---|---|---|---|
| | Description | Bits | Description | Bits | Description | Bits | Description | Bits |
| Solid field | N/A | | N/A | | N/A | | N/A | |
| Fixed step horizontal ramp | N/A | | N/A | | Brightest pixel value | 8 | Darkest pixel value | 8 |
| Fixed step vertical ramp | N/A | | N/A | | Brightest pixel value | 8 | Darkest pixel value | 8 |
| Horizontal lines | N/A | | N/A | | Background line width | 8 | Foreground line width | 8 |
| Vertical lines | N/A | | N/A | | Background line width | 8 | Foreground line width | 8 |
| Diagonal lines | N/A | | N/A | | Vertical spacing | 8 | Horizontal spacing | 8 |
| Grid lines | Vertical background line width | 8 | Vertical foreground line width | 8 | Horizontal background line width | 8 | Horizontal foreground line width | 8 |

**Table 3-3. Descriptions and Bit Assignments for Parameters 1-4 (continued)**

| Pattern | Byte 6 (Parameter 4) | | Byte 5 (Parameter 3) | | Byte 4 (Parameter 2) | | Byte 3 (Parameter 1) | |
|---|---|---|---|---|---|---|---|---|
| | Description | Bits | Description | Bits | Description | Bits | Description | Bits |
| Checkerboard | Number of vertical checkers (MSB) | 3 | Number of vertical checkers (LSB) | 8 | Number of horizontal checkers (MSB) | 3 | Number of horizontal checkers (LSB) | 8 |
| Color bars | N/A | | N/A | | N/A | | N/A | |

1. This command is used in conjunction with the *Write Operating Mode Select* command (Section 3.1.1). This command specifies which test pattern is to be displayed when the *Write Operating Mode Select* command selects Test Pattern Generator as the image source. The settings for this command are to be retained until changed using this command. The controller automatically applies these settings each time the Test Pattern Generator is selected.
2. Batch files (Section 3.1.14) can be created and stored in Flash and used to recall the settings for predefined test patterns.
3. The Test Pattern border selection creates a single pixel wide and tall white border around the specified test pattern.
4. *It is important that the user review the notes for the Write Operating Mode Select command (Section 3.1.1) to understand the concept of source associated commands. This concept determines when source associated commands are executed by the system. Note that this command is a source associated command.*
5. When a Foreground or Background Color is not used, the controller ignores the corresponding bit values. The number of parameter bytes to be used depends on the selected pattern. Table 3-4 shows the number of bytes to be used based on the specified pattern.

**Table 3-4. Number of Bytes Required based on Pattern Selection**

| **Specified Pattern** | Solid Field | Fixed Step Horz Ramp | Fixed Step Vert Ramp | Horz Lines | Vert Lines | Diag Lines | Grid Lines | Checker board | Color Bars |
|---|---|---|---|---|---|---|---|---|---|
| **Number of Bytes Required** | 2 | 4 | 4 | 4 | 4 | 4 | 6 | 6 | 1 |

6. The color for the fixed step vertical ramp pattern is specified using the foreground color. As noted in Table 3-3, the user specifies the start value and the stop value for the ramp. For this pattern, the system automatically determines the step size based on the start and stop values and the size of the display (DMD). The minimum start value = 0, the maximum stop value = 255, and the start value must always be smaller than the stop value. For example, if the start value = 0, the stop value = 255, and the DMD resolution is 768 tall, then the step size is 3 (768 pixels / 256 values = 3). Thus every value from 0 to 255 has a step size of 3 pixels (such that each step has 3 rows of pixels with the same gray scale value). The gray scale value always increments by 1 for each step between the start and stop values. An example of a fixed step vertical ramp pattern is shown in Figure 3-1.

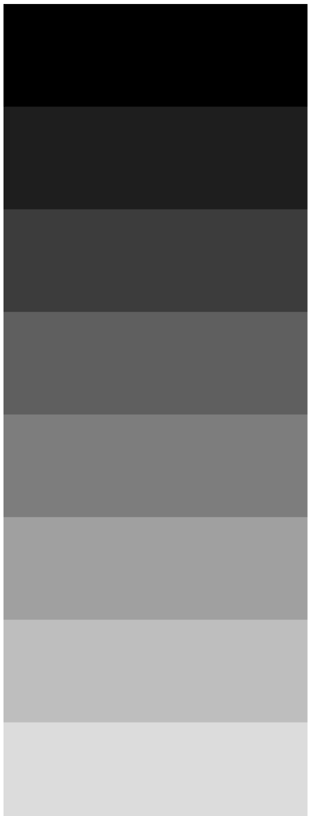


**Figure 3-1. Example of Fixed Step Vertical Ramp Test Pattern**

7. The colors for the horizontal lines pattern are specified using both the foreground and background colors. The foreground color is used for the horizontal lines, and the background color is used for the space between the lines. As noted in Table 3-3, the user specifies the foreground line width, as well as the background line width. The user must determine the line spacing for each resolution display. For example, if the foreground line width = 1, and the background line width = 9, there is a single pixel horizontal line on every tenth line. An example of a horizontal lines pattern is shown in Figure 3-2.

**Figure 3-2. Example of Horizontal Lines Test Pattern**

8. The colors for the vertical lines pattern are specified using both the foreground and background colors. The foreground color is used for the vertical lines, and the background color is used for the space between the lines. As noted in Table 3-3, the user specifies the foreground line width, as well as the background line width. The user must determine the line spacing for each resolution display. For example, if the foreground line width = 1, and the background line width = 9, there is a single pixel vertical line on every tenth line. An example of a vertical lines pattern is shown in Figure 3-3.

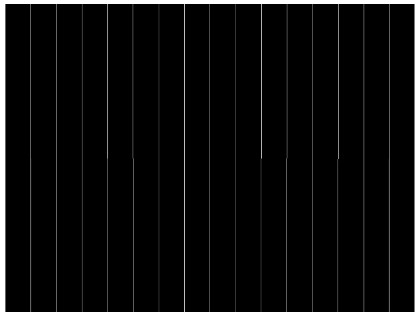


**Figure 3-3. Example of Vertical Lines Test Pattern**

9. The colors for the diagonal lines pattern are specified using both the foreground and background colors. The foreground color is used for the diagonal lines, and the background color is used for the space between the lines. As noted in Table 3-3, the user specifies the horizontal and vertical line spacing. The line width is always one pixel. The user determines the line spacing for each resolution display. Both horizontal and vertical line spacing must use the same value, and are limited to values of 3, 7, 15, 31, 63, 127, and 255. Invalid values result in a communication error (invalid command parameter). An example of a diagonal lines pattern is shown in Figure 3-4.

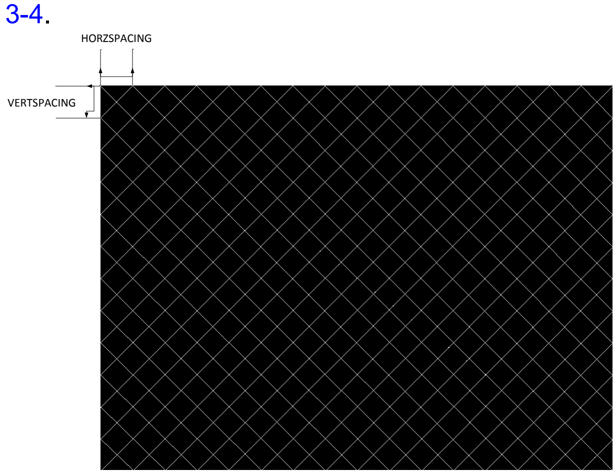



**Figure 3-4. Example of Diagonal Lines Test Pattern**

10. The colors for the grid lines pattern are specified using both the foreground and background colors. The foreground color is used for the grid lines, and the background color is used for the space between the lines. As noted in Table 3-3, the user specifies the horizontal foreground and background line width, as well as the vertical foreground and background line width. The user determines the line spacing for each resolution display. For example, if the horizontal foreground line width = 1, and background line width = 9, there is a single pixel horizontal line on every tenth line. If the vertical foreground line width = 1, and background line width = 9, there is a single pixel vertical line on every tenth line. An example of a grid lines pattern is shown in Figure 3-5.

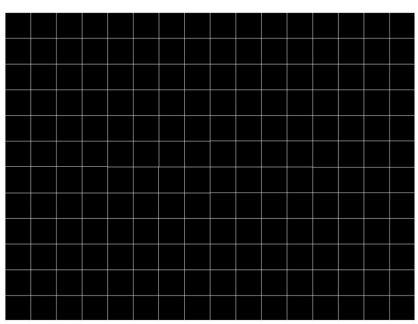


**Figure 3-5. Example of Grid Lines Test Pattern**

11. The colors for the checkerboard pattern are specified using both the foreground and background colors. The foreground color is used for one of the checkers, and the background color is used for the alternating checker. As noted in Table 3-3, the user specifies the number of horizontal checkers and the number of vertical checkers. For this pattern, the system automatically determines the checker size in each direction

based on the number of checkers and the size of the display (DMD). For example, if the number of horizontal checkers = 4, the number of vertical checkers = 4, and the DMD resolution is 1280x720, the size of the horizontal checkers is 320 pixels, and the size of the vertical checkers is 180 pixels (1280 pixels / 4 checkers = 320 pixels: 720 pixels / 4 checkers = 180 pixels). An example of a checkerboard pattern (16 checkers by 12 checkers) is shown in Figure 3-6.

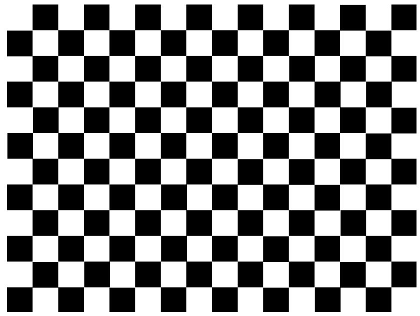


**Figure 3-6. Example of Checkerboard Test Pattern**

### 3.1.4 Read Test Pattern Select (0Ch)

This command reads the state of the test pattern selected for the display module.

#### *3.1.4.1 Read Parameters*

This command has no parameters.

#### *3.1.4.2 Return Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | TPG pattern select |
| Byte 2 | Foreground and background color (see Table 3-2) |
| Byte 3 | Parameter 1 (see Table 3-3) |
| Byte 4 | Parameter 2 (see Table 3-3) |
| Byte 5 | Parameter 3 (see Table 3-3) |
| Byte 6 | Parameter 4 (see Table 3-3) |

This command always returns six bytes as the host does not know how many bytes are valid until the pattern is selected. All unnecessary bytes (see Table 3-4) are set to '0'.

### 3.1.5 Write Splash Screen Select (0Dh)

#### *3.1.5.1 Write*

This command selects a stored splash screen to be displayed.

#### *3.1.5.2 Write Parameters (0Dh)*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Splash screen reference number (integer) |

1. This command is used in conjunction with the *Write Operating Mode Select* (Section 3.1.1) and the *Write Splash Screen Execute* (Section 3.1.15) commands. It specifies which splash screen is to be displayed when the *Input Source Select* command selects splash screen as the image source. The controller retains the settings for this command until changed using this command.
2. The steps required to display a splash screen are: select the desired splash screen (this command), change the input source to splash screen (using *Write Operating Mode Select*), and start the splash screen retrieval process (using *Write Splash Screen Execute*).
3. The Splash Screen is a unique source since it is read from Flash and sent down the processing path of the controller one time, to be stored in memory for display at the end of the processing path. As such, the user must set *all image processing settings (e.g. image crop, image orientation, display size, splash screen select, look select, splash screen as input source) before executing the Write Splash Screen Execute command.*
4. *It is important that the user review the notes for the Write Operating Mode Select command (Section 3.1.1) to understand the concept of source associated commands. This concept determines when source associated commands are executed by the system. Note that this command is a source associated command.*
5. The availability of splash screens is limited by the available space in flash memory.
6. All splash screens must be *landscape* oriented.
7. When this command is received while Splash Screen is the active source, other than storing the specified splash screen value, the only action of the controller software is to obtain the header information from the selected splash screen and store this in internal memory. Then, when the *Write Splash Screen Execute* command is received, the controller software uses this stored information to set up the processing path prior to pulling the splash data from flash.

**Note**

The DLPC143x only supports native resolution splash screen images. For the DLPC1438 this resolution is 1280x720. The system will not be properly configured if a resolution other than the DMD's native resolution is input.

### 3.1.6 Read Splash Screen Select (0Eh)

This command reads the last selected splash screen index.

#### *3.1.6.1 Read Parameters*

This command has no parameters.

#### *3.1.6.2 Return Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Splash screen index |

### 3.1.7 Read Splash Screen Header (0Fh)

This command reads the splash screen header information for the selected splash screen.

#### *3.1.7.1 Read Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Splash screen reference number (integer) |

The read parameter is used to specify the splash screen for which the header parameters are to be returned. If a splash screen value is provided for which there is no splash screen available, the controller issues an error (invalid command parameter value – communication status) and the command does not execute.

#### *3.1.7.2 Return Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Splash image width in pixels (LSByte) |
| Byte 2 | Splash image width in pixels (MSByte) |
| Byte 3 | Splash image height in pixels (LSByte) |
| Byte 4 | Splash image height in pixels (MSByte) |
| Byte 5 | Splash image size in bytes (LSByte) |
| Byte 6 | Splash image size in bytes |
| Byte 7 | Splash image size in bytes |
| Byte 8 | Splash image size in bytes (MSByte) |
| Byte 9 | Pixel format |
| Byte 10 | Compression type |
| Byte 11 | Color order |
| Byte 12 | Chroma order |
| Byte 13 | Byte order |

Parameter definitions referenced are in Table 3-5.

**Table 3-5. Splash Screen Header Definitions**

| Parameter | Values |
|---|---|
| Pixel format | '0h' = 24-bit RGB unpacked (not used)<br>'1h' = 24-bit RGB packed (not used)<br>'2h' = 16-bit RGB 5-6-5<br>'3h' = 16-bit YCbCr 4:2:2 |
| Compression type | '0h' = Uncompressed<br>'1h' = RGB RLE compressed<br>'2h' = User-defined (not used)<br>'3h' = YUV RLE compressed |
| Color order | '0h' = 00RRGGBB<br>'1h' = 00GGRRBB |
| Chroma order | '0h' = Cr is first pixel<br>'1h' = Cb is first pixel |
| Byte order | '0h' = Little endian<br>'1h' = Big endian |

### 3.1.8 Write Display Image Orientation (14h)

This command specifies the image orientation of the displayed image. *This setting does not apply to Internal Pattern mode.*

#### *3.1.8.1 Write Parameters*

| Parameter Byte | |
|---|---|
| b(7:3) | *Reserved* |
| b(2) | Short axis image flip:<br>• 0: Image not flipped.<br>• 1: Image flipped. |
| b(1) | Long axis image flip:<br>• 0: Image not flipped.<br>• 1: Image flipped. |
| b(0) | |


1.

**Figure 3-7. Short-Axis Flip**


**Figure 3-8. Long-Axis Flip**

### 3.1.9 Read Display Image Orientation (15h)

This command reads the specified displayed image orientation.

#### *3.1.9.1 Read Parameters*

This command has no parameters.

#### *3.1.9.2 Return Parameters*

| Parameter Byte | |
|---|---|
| b(7:3) | *Reserved* |
| b(2) | Short axis image flip:<br>• 0: Image not flipped.<br>• 1: Image flipped. |
| b(1) | Long axis image flip:<br>• 0: Image not flipped.<br>• 1: Image flipped. |
| b(0) | *Reserved* |

### 3.1.10 Write Display Image Curtain (16h)

This command fills the entire display with a user-specified color.

#### *3.1.10.1 Write Parameters*

| Parameter Byte | |
|---|---|
| b(7:4) | *Reserved* |
| b(3:1) | Select curtain color:<br>• 0h: Black<br>• 7h: White |
| b(0) | Curtain enable:<br>• '0': Curtain disabled<br>• '1': Curtain enabled |

- The Image Curtain fills the entire display with a user specified color.

### 3.1.11 Read Display Image Curtain (17h)

This command reads the state of the image curtain control function.

#### *3.1.11.1 Read Parameters*

This command has no parameters.

#### *3.1.11.2 Return Parameters*

| Parameter Byte | |
|---|---|
| b(7:4) | *Reserved* |
| b(3:1) | Select curtain color:<br>• 0h: Black<br>• 7h: White |
| b(0) | Curtain enable:<br>• '0': Curtain disabled<br>• '1': Curtain enabled |

### 3.1.12 Write Image Freeze (1Ah)

This command enables or disables the image freeze function.

#### *3.1.12.1 Write Parameters*

| Parameter Byte | |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | Image freeze:<br>• '0': Image freeze disabled<br>• '1': Image freeze enabled |

Default: 00h

1. Normal use of the Image Freeze capability typically has two main functions. The first function is to allow the end user to freeze the current image on the screen for their own uses. The second function is to allow the user to reduce or prevent system changes from showing up on the display as visual artifacts. In this second case, the image is frozen, system changes are made, and when complete, the image is unfrozen. In all cases, when the image is unfrozen, the display starts showing the most resent input image. Thus input data between the freeze point and the unfreeze point is lost. Suggestions to the host system for the types of image changes likely to necessitate the use of the image freeze command to hide artifacts are discussed in Section 3.1.12.2.
2. The controller software never (either *automatically* nor *under-the-hood*) freezes or unfreezes the image. This applies when software is making updates to the system on its own volition, as well as for any operation commanded via $I^2C$. The controller software does not freeze or unfreeze the image for any reason except when explicitly commanded by the *Write Image Freeze* command.
3. If the user chooses not to make use of Image Freeze, is recommended that they change the source itself before changing image parameters to minimize transition artifacts.

#### *3.1.12.2 Use of Image Freeze to Reduce On-Screen Artifacts*

Commands that take a long time to process, require a lot a data to be loaded from flash, or change the frame timing of the system may create on-screen artifacts. The *Write Image Freeze* command can try and minimize, if not eliminate, these artifacts. The process is:

1. Send a *Write Image Freeze* command to enable freeze.
2. Send commands with the potential to create image artifacts.
3. Send a *Write Image Freeze* command to disable freeze.

As the system processes commands to the controller serially, it requires no special timing or delay between these commands. Make sure that the number of commands placed between the freeze and unfreeze is small, as it is likely not desirable for the image to be frozen for a "long" period of time. A list of commands that may produce image artifacts is listed in Table 3-6. This is not an all-inclusive list, however, and the user is responsible for determining if and when use of the image freeze command meets their product needs.

**Table 3-6. Partial List of Commands that May Benefit from the Use of Image Freeze**

| Command | Command OpCode |
|---|---|
| Write Input Source Select | 05h |
| Write External Video Source Format Select(1) | 07h |
| Write Test Pattern Select(1) | 0Bh |
| Write Splash Screen Select(1) | 0Dh |
| Write Look Select | 22h |

(1) If changed while this source is the active source

Table 3-7 of how to use the image freeze command.

**Table 3-7. Test Pattern Generator Example Using Image Freeze**

| Command | Notes |
|---|---|
| Write Image Freeze = Freeze | |

**Table 3-7. Test Pattern Generator Example Using Image Freeze (continued)**

| Command | Notes |
| --- | --- |
| Write Image Crop, Write Display Size, Write Display Image Orientation, Write Test Pattern Select. | Potential data processing commands that may be required for proper display of test pattern image. These are used as appropriate. It is recommended that these be set before the Write Operating Mode Select command. |
| **Write Operating Mode Select = Test Pattern Generator** | |
| Write Image Freeze = Unfreeze | |

### 3.1.13 Read Image Freeze (1Bh)

This command reads the state of the image freeze function.

#### *3.1.13.1 Read Parameters*

This command has no parameters.

#### *3.1.13.2 Return Parameters*

| Parameter Byte | |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | Image freeze:<br>• '0': Image freeze disabled<br>• '1': Image freeze enabled |

### 3.1.14 Write Execute Flash Batch File (2Dh)

This command executes a batch file stored in flash.

#### *3.1.14.1 Write Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Batch File Number |

1. Most system *Write* commands specified in this document that can be sent by itself can be grouped together with other system commands or command parameters into a Flash batch file. See Table 3-8 for a list of commands which cannot be used in batch files.
2. One example for a Flash batch file might be the commands and command parameters required for initialization of the system after power-up.
3. The Flash batch file numbers to be specified in this byte are enumerated values (0,1,2,3...).
4. Flash batch file 0 is a special Auto-Init batch file that is run automatically by the DLPC148X software immediately after system initialization has been completed. The controller does not typically call the Flash batch file 0 using the *Write Execute Flash Batch File* command (although the system does allow it). This special Flash batch file typically specifies the default operating mode the system initializes to.
5. Embedding Flash batch file calls within a Flash batch file is not allowed (i.e. calling another batch file from within a batch file is not allowed). Multiple batch files can be executed consecutively by sending multiple execute batch file commands.
6. The system provides the ability to add an execution delay between commands within a Flash batch file. This is done using the *Write Flash Batch File Delay (DBh)* command (See Section 3.4.1.9).
7. The order of command execution for commands within a Flash batch file is the same as if the commands had been received over the $I^2C$ port.

**Table 3-8. List of Commands Excluded from Batch File Use**

| Command | Op-Code |
|---|---|
| Write Execute Flash Batch File | 2D |
| Write Flash Data Type Select | DE |
| Write Flash Data Length | DF |
| Write Erase Flash Data | E0 |
| Write Flash Start | E1 |
| Write Flash Continue | E2 |
| All *Read* commands | Various |

### 3.1.15 Write Splash Screen Execute (35h)

This command starts the process of retrieving a splash screen from flash for display.

#### *3.1.15.1 Write Parameters*

This command has no parameters.

Some important points to be noted about this command:

1. This command is used in conjunction with the Write Operating Mode Select (Section 3.1.1) and the Write Splash Screen Select (Section 3.1.6) commands. It is used to start the process of retrieving a splash screen from Flash for display.
2. The Splash Screen is a unique source as it is read from Flash and sent down the processing path of the controller one time, to be stored in memory for display at the end of the processing path. Set *all image processing settings* (image size, image crop, image orientation, display size, splash screen select, look select, splash screen as input source) *before executing this command*. Any data path processing changed after the splash screen has been executed requires this command to be re-executed before the controller displays the result. This way, the controller repeats the splash screen retrieval process each time it receives this command.
3. The process of retrieving the splash screen from SPI Flash can take a significant amount of time depending on the size of the compressed image stored in flash. During this period, the controller will not accept any new I2C commands. The user must ensure that the splash screen has been successfully displayed before sending any further commands.
4. When this command is processed in Display - Splash Screen mode, the system automatically initializes the system color processing based on the splash header information prior to sending the splash image down the data path. However, in Light Control - Splash Pattern mode, no color processing is performed and the stored bitmap is displayed as is, irrespective of whether the image is stored in RGB565 format or YCrCb (16-bit) format. Therefore, to get an accurate representation of the input image in splash pattern mode, the images must be stored in RGB565 format in flash.
5. *It is important that the user review the notes for the Write Operating Mode Select command (Section 3.1.1) to understand the concept of source associated commands. This concept determines when source associated commands are executed by the system. Note that this command is a source associated command; however, this command is special in that there is no maintained state or history. Thus, this command has no “settings” to be stored or reused by the system.*

## 3.2 Illumination Control Commands

### 3.2.1 Write Illuminator LED Enable (52h)

This command enables or disables individual LEDs. *This setting cannot be changed during and active print while in External Print modes.*

#### *3.2.1.1 Write Parameters*

| **Byte 1** | |
|---|---|
| b(7:3) | *Reserved* |
| b(2) | LED 1 enable:<br>• 0: LED 1 disabled<br>• 1: LED 1 enabled |
| b(1) | LED 2 enable:<br>• 0: LED 2 disabled<br>• 1: LED 2 enabled |
| b(0) | LED 3 enable:<br>• 0: LED 3 disabled<br>• 1: LED 3 enabled |

Default: Firmware specified

### 3.2.2 Read Illuminator LED Enable (53h)

This command reads the state of the LED enables.

#### 3.2.2.1 Read Parameters

This command has no parameters.

#### 3.2.2.2 Return Parameters

| Byte 1 | |
|---|---|
| b(7:3) | *Reserved* |
| b(2) | LED 1 enable:<br>• 0: LED 1 disabled<br>• 1: LED 1 enabled |
| b(1) | LED 2 enable:<br>• 0: LED 2 disabled<br>• 1: LED 2 enabled |
| b(0) | LED 3 enable:<br>• 0: LED 3 disabled<br>• 1: LED 3 enabled |

### 3.2.3 Write Illuminator LED Current PWM (54h)

This command sets the PWM values for the LEDs 1, 2, and 3.

#### *3.2.3.1 Write Parameters*

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | LED 1 PWM parameter | LSByte |
| Byte 2 | LED 1 PWM parameter | MSByte |
| Byte 3 | LED 2 PWM parameter | LSByte |
| Byte 4 | LED 2 PWM parameter | MSByte |
| Byte 5 | LED 3 PWM parameter | LSByte |
| Byte 6 | LED 3 PWM parameter | MSByte |

1. When an all-white image is being displayed, this command allows the system white point to be adjusted while also establishing the total LED power.
2. The parameters specified by this command have a resolution of 10 bits, and are defined by the appropriate PMIC specification.

### 3.2.4 Read Illuminator LED Current PWM (55h)

This command reads the PWM parameters for the illuminator LEDs of the display module.

#### 3.2.4.1 Read Parameters

This command has no parameters.

#### 3.2.4.2 Return Parameters

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | LED 1 PWM parameter | LSByte |
| Byte 2 | LED 1 PWM parameter | MSByte |
| Byte 3 | LED 2 PWM parameter | LSByte |
| Byte 4 | LED 2 PWM parameter | MSByte |
| Byte 5 | LED 3 PWM parameter | LSByte |
| Byte 6 | LED 3 PWM parameter | MSByte |

See *Section 3.2.3* for a detailed description of the return parameters.

Unused most significant bits are set to '0'.

### 3.2.5 Write Illuminator LED Max Current PWM (5Ch)

This command specifies the maximum LED PWM allowed for each LED.

#### *3.2.5.1 Write Parameters*

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Maximum LED 1 PWM | LSByte |
| Byte 2 | Maximum LED 1 PWM | MSByte |
| Byte 3 | Maximum LED 2 PWM | LSByte |
| Byte 4 | Maximum LED 2 PWM | MSByte |
| Byte 5 | Maximum LED 3 PWM | LSByte |
| Byte 6 | Maximum LED 3 PWM | MSByte |

Default: Firmware specified

1. The parameters specified by this command have a resolution of 10 bits, and are defined by the appropriate PMIC specification.
2. Set unused most significant bits to ‘0’.

### 3.2.6 Read Illuminator LED Max Current PWM (5Dh)

This command reads the specified maximum LED PWM allowed for each LED.

#### *3.2.6.1 Read Parameters*

This command has no parameters.

#### *3.2.6.2 Return Parameters*

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Maximum LED 1 PWM | LSByte |
| Byte 2 | Maximum LED 1 PWM | MSByte |
| Byte 3 | Maximum LED 2 PWM | LSByte |
| Byte 4 | Maximum LED 2 PWM | MSByte |
| Byte 5 | Maximum LED 3 PWM | LSByte |
| Byte 6 | Maximum LED 3 PWM | MSByte |

1. See the Write *RGB LED PWM Control* command for a detailed description of the return parameters.
2. Unused most significant bits are set to‘0’.

## 3.3 3D Print Commands

This section describes the commands used in 3D Print modes. The commands for 3D Print include:

1. Trigger Out Configuration
   a. Write Trigger Out Configuration (see Section 3.3.3)
   b. Read Trigger Out Configuration (see Section 3.3.4)
2. Sensing Sequence Information
   a. Read Sensing Sequence Version (see Section 3.3.5)
3. External Print Control
   a. Write External Print Configuration (see Section 3.3.6)
   b. Read External Print Configuration (see Section 3.3.7)
   c. Write External Print Control (see Section 3.3.8)
   d. Read External Print Control (see Section 3.3.9)
4. FPGA SPI Processing Control
   a. Write Parallel Video (see Section 3.3.10)
   b. Read Parallel Video (see Section 3.3.11)
   c. Write Active Buffer (see Section 3.3.12)
   d. Read Active Buffer (see Section 3.3.13)
   e. Write FPGA Control (see Section 3.3.14)
   f. Read FPGA Control (see Section 3.3.15)
   g. Read FPGA SPI CRC16 (see Section 3.3.16)

### 3.3.1 3D Print Procedure Without FPGA Front-End

To properly configure the DLPC143x controller for External Print mode without an FPGA controlling the front-end of data being fed, the following commands are to be sent in order:

First layer:

1. Set External Print Configuration (see Section 3.3.6)
2. Send video data to DLP143x parallel input lines
3. Set Operating Mode to External Print (see Section 3.1.1)

**Note**

any command issued after this command will need to wait on System_Ready

4. Set External Print Layer Control with the needed dark and exposed frames (see Section 3.3.8)

Subsequent layers:

1. Send video data to DLP143x parallel input lines

2. Set External Print Layer Control with the needed dark and exposed frames (see Section 3.3.8)
3. Continue to send layers until last layer

Note: Without an FPGA to act as a buffer, the DLPC1438 immediately displays any changing image data received on the parallel data lines and actuator features are not be supported. It is recommended to allow the complete frame count to be run before changing video data on the parallel inputs.

### 3.3.2 3D Print Procedure With FPGA Front-End

To properly configure the DLPC143x controller for External Print mode with an FPGA controlling the front-end of data being fed, the following commands are to be sent in order:

First layer:

1. Ensure that CRC16 calculation is enabled (see Section 3.3.14)
2. Ensure that buffer active number is 0 (see Section 3.3.12)
   a. Note this can be set to active buffer 1 if active buffer is set to 0 in step 5.
3. Set External Print Configuration (see Section 3.3.6)
4. Send image into buffer
   a. SPI transmission speed can be set up to 50 MB/s
   b. Configure video data according to SPI specifications (see Section 4.1)
   c. Follow formatting guidelines (see Section 4.2)
   d. SPI images will be sent into **non-active** buffer
5. Write FPGA active buffer to 1 (see Section 3.3.12)
   a. This causes the FPGA to read and transmit the data on buffer 0 while allowing data to be written to buffer 1
6. Set parallel buffer to "Read and send buffer" (see Section 3.3.10)
7. Set Operating Mode to External Print (see Section 3.1.1)

---

**Note**

any command issued after this command will need to wait on System_Ready

---

8. Set External Print Layer Control with the needed dark and exposed frames (see Section 3.3.8)
   a. The FPGA data is updated at the beginning of the sequence, so previous data stored in the FPGA is transmitted before new data. This may potentially cause an undesired image to be projected before the intended image. It is therefore recommended to allow 3 or more dark frames before exposed frames to allow the data to reach the ASIC.
9. Check FPGA SPI CRC16 Value (see Section 3.3.16) or check SPI CRC Error (see Section 3.5.2) if needed

Subsequent layers:

1. Send image into buffer
2. Write FPGA active buffer (see Section 3.3.12)
   a. Write FPGA active buffer to 0 for later even layers if buffer active number is set to 0 in step 2 above and written active buffer is set to 1 in step 5 above.
   b. Write FPGA active buffer to 1 for later odd layers if buffer active number is set to 0 in step 2 above and written active buffer is set to 1 in step 5 above.
3. Set External Print Layer Control with the needed dark and exposed frames (see Section 3.3.8)
4. Check FPGA SPI CRC16 Value (see Section 3.3.16) or check SPI CRC Error (see Section 3.5.2) if needed
5. Continue to send even and odd layers until last layer

### 3.3.3 Write Trigger Out Configuration (92h)

This command defines the Trigger Out characteristics.

#### *3.3.3.1 Write Parameters (92h)*

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Trigger Configuration | |
| Byte 2 | Delay (in µs) | LSByte |
| Byte 3 | Delay (in µs) | |
| Byte 4 | Delay (in µs) | |
| Byte 5 | Delay (in µs) | MSByte |

| Byte 1 | Trigger Configuration |
|---|---|
| b(7:3) | *Reserved* |
| b(2) | Trigger Inversion<br>'0': Not Inverted<br>'1': Inverted |
| b(1) | Trigger Enable<br>'0': Disable<br>'1': Enable |
| b(0) | Trigger Select<br>'0': Trigger Out 1<br>'1': Trigger Out 2 |

Default: Flash settings defined

1. Trigger Out settings are only applied when the user sends *Write Operating Mode Select* command with an External Print mode selected.
2. The controller retains and uses the Trigger Out setting until next *Write Trigger Out Configuration* command.
3. Delay range of Trigger Out 1: [0, Pattern Period]
4. Delay range of Trigger Out 2: [ -Pre-Illumination Dark Time, Pattern Period]
5. Trigger Out 2 supports negative values, meaning the trigger can be sent in advance. The delay is processed as signed 16-bit integer.

### 3.3.4 Read Trigger Out Configuration (93h)

This command reads the Trigger Out configuration of the specified trigger.

#### *3.3.4.1 Read Parameters*

| Parameter Byte | |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | Trigger Select<br>'0': Trigger Out 1<br>'1': Trigger Out 2 |

#### *3.3.4.2 Return Parameters*

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Trigger Configuration | |
| Byte 2 | Delay (in µs) | LSByte |
| Byte 3 | Delay (in µs) | |
| Byte 4 | Delay (in µs) | |
| Byte 5 | Delay (in µs) | MSByte |

| Byte 1 | Trigger Configuration |
|---|---|
| b(7:3) | *Reserved* |
| b(2) | Trigger Inversion<br>'0': Not Inverted<br>'1': Inverted |
| b(1) | Trigger Enable<br>'0': Disable<br>'1': Enable |
| b(0) | Trigger Select<br>'0': Trigger Out 1<br>'1': Trigger Out 2 |

Trigger Out 2 supports negative values, meaning the trigger can be sent in advance. The delay is processed as signed 16-bit integer.

### 3.3.5 Read Sensing Sequence Version (9Bh)

This command reads the 3D Print Sequence Binary version.

#### *3.3.5.1 Read Parameters*

This command has no parameters.

#### *3.3.5.2 Return Parameters*

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Sensing Sequence Binary version | Reserved |
| Byte 2 | Sensing Sequence Binary version | Patch |
| Byte 3 | Sensing Sequence Binary version | Minor |
| Byte 4 | Sensing Sequence Binary version | Major |

### 3.3.6 Write External Print Configuration (A8h)

This command sets the degamma index and designates the illuminator to be used for External Print Mode.

NOTE: This command cannot be properly executed when in External Print mode to avoid changing the print mode during a print sequence. Setting the External Print Configuration must therefore be completed in either

Test Pattern Generator mode or Standby mode before entering into External Print mode as described in Section 3.3.1 and Section 3.3.2.

#### 3.3.6.1 Write Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Degamma Transfer Function Index select |
| Byte 2 | Illuminator LED enable |

| Byte 1 | Degamma Transfer Function Index select |
|---|---|
| 00h | Linear |
| 01h | Uniformity Correction Optimized |

1. The Linear degamma transfer functions indices correspond to linear CMT LUTs from a 0 value to 255.
2. The Uniformity Correction Optimized transfer function indices correspond to CMT LUTs that are optimized for 3D print applications. These tables have a begin at a 0 value. The tables next value outputs a light intensity of roughly half of the maximum (128) and then increments linearly to the maximum intensity.

| Byte 2 | Description |
|---|---|
| b(7:2) | *Reserved* |
| b(2) | Illuminator LED 3 enable:<br>• '0': LED disable<br>• '1': LED enable |
| b(1) | Illuminator LED 2 enable:<br>• '0': LED disable<br>• '1: LED enable |
| b(2) | Illuminator LED 3 enable:<br>• '0': LED disable<br>• '1': LED enable |

1. NOTE: The 3D Print software has been designed to support only one LED. Multiple LEDs are not to be enabled at the same time.

### 3.3.7 Read External Print Configuration (A9h)

This command reads the degamma index and illuminator to be used for External Print Mode.

#### 3.3.7.1 Read Parameters

This command has no parameters.

#### 3.3.7.2 Return Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Degamma Transfer Function Index select |
| Byte 2 | Illuminator LED enable |

| Byte 1 | Degamma Transfer Function Index select |
|---|---|
| 00h | Linear |
| 01h | Uniformity Correction Optimized |

1. The Linear degamma transfer functions indices correspond to linear CMT LUTs from a 0 value to 255.
2. The Uniformity Correction Optimized transfer function indices correspond to CMT LUTs that are optimized for 3D print applications. These tables have a begin at a 0 value. The tables next value outputs a light intensity of roughly half of the maximum (128) and then increments linearly to the maximum intensity.

| Byte 2 | Description |
|---|---|
| b(7:2) | *Reserved* |

| Byte 2 | Description |
|---|---|
| b(2) | Illuminator LED 3 enable:<br>• '0': LED disabled<br>• '1': LED enabled |
| b(1) | Illuminator LED 2 enable:<br>• '0': LED disabled<br>• '1: LED enabled |
| b(2) | Illuminator LED 3 enable:<br>• '0': LED disabled<br>• '1': LED enabled |

1. NOTE: The 3D Print software has been designed to support only one LED. Multiple LEDs are not to be enabled at the same time.

### 3.3.8 Write External Print Control (C1h)

This command specifies the control for the 3D Print external print layer. This includes enabling the print control, setting the number of dark frames, and setting the number of exposed frames. Sending this command begins the count of exposed frames after the External Print Configuration has been established.

#### *3.3.8.1 Write Parameters*

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | External Print Control | |
| Byte 2 | Number of Dark Frames | LSByte |
| Byte 3 | Number of Dark Frames | MSByte |
| Byte 4 | Number of Exposed Frames | LSByte |
| Byte 5 | Number of Exposed Frames | MSByte |

| Byte 1 | Description |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | External Print Control:<br>• '0': Start<br>• '1': Stop |

1. The FPGA data is updated at the beginning of the sequence, so previous data stored in the FPGA is transmitted before new data. This may potentially cause an undesired image to be projected before the intended image. It is therefore recommended to allow 3 or more dark frames before exposed frames to allow the data to reach the ASIC.

**Note**

If the Number of Exposed frames is set to 0xFFFF (the mad 16-bit value), the Number of Exposed frames will run infinitely. This state can be exited by entering into Standby Mode.

### 3.3.9 Read External Print Control (C2h)

This command reads the status of the 3D External Print. This includes enabling the print control, the number of dark and exposed frames set to run, and the number of dark and exposed frames that have run.

#### *3.3.9.1 Read Parameters*

This command has no parameters.

#### *3.3.9.2 Return Parameters*

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | External Print Control | |
| Byte 2 | Number of Dark Frames | LSByte |

| Parameter Bytes | Description | |
|---|---|---|
| Byte 3 | Number of Dark Frames | MSByte |
| Byte 4 | Number of Exposed Frames | LSByte |
| Byte 5 | Number of Exposed Frames | MSByte |

| Byte 1 | Description |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | External Print Control:<br>• '0': Start<br>• '1': Stop |

### 3.3.10 Write Parallel Video (C3h)

This command enables the FPGA parallel video interface. This command and Write FPGA TPG (see Section 3.5.3) cannot enable at the same time. In order to disable this command, switch the DLPC1438 into Standby mode through the Operating Mode Select (see Section 3.1.1).

#### *3.3.10.1 Write Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | FPGA Parallel Video Buffer Enable |

| Byte 1 | Description |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | Parallel Buffer Active<br>• '0': Don't read buffer<br>• '1': Read and send buffer |

### 3.3.11 Read Parallel Video (C4h)

This command reads the parallel video interface enable.

#### *3.3.11.1 Read Parameters*

This command has no parameters.

#### *3.3.11.2 Return Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | FPGA Parallel Video Buffer Enable |

| Byte 1 | Description |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | Parallel Buffer Active<br>• '0': Don't read buffer<br>• '1': Read and send buffer |

### 3.3.12 Write Active Buffer (C5h)

This command selects the FPGA active buffer number for SPI writes.

#### *3.3.12.1 Write Parameters*

| Parameter Byte | Description |
|---|---|
| Byte 1 | FPGA Active Buffer |

1. The active buffer receives the incoming data from the SPI front-end.
2. The inactive buffer contains the data that is actively transmitted to the ASIC.

### 3.3.13 Read Active Buffer (C6h)

This command returns the FPGA active buffer number for SPI writes.

#### *3.3.13.1 Read Parameters*

This command has no parameters.

#### *3.3.13.2 Return Parameters*

| Parameter Byte | Description |
|---|---|
| Byte 1 | FPGA Active Buffer |

1. The active buffer receives the incoming data from the SPI front-end.
2. The inactive buffer contains the data that is actively transmitted to the ASIC.

### 3.3.14 Write FPGA Control (CAh)

This command writes the FPGA control parameters for the pixel data stream from the SPI interface to the ASIC.

#### *3.3.14.1 Write Parameters*

| Parameter Byte | Description |
|---|---|
| b(7:4) | *Reserved* |
| b(3) | FPGA CRC16 error inject:<br>• '0': No error injection<br>• '1': Error injection |
| b(2) | FPGA CRC16 calculation:<br>• '0': Disable CRC16 calculation<br>• '1': Enable CRC16 calculation |
| b(1) | FPGA reset:<br>• '0': No reset<br>• '1': Reset FPGA |
| b(0) | FPGA reset unlock:<br>• '0': Don't allow reset<br>• '1': Allow reset |

### 3.3.15 Read FPGA Control (CBh)

This command returns the FPGA control parameters for the pixel data stream from the SPI interface to the ASIC.

#### *3.3.15.1 Read Parameters*

This command has no parameters.

#### *3.3.15.2 Return Parameters*

| Parameter Byte | Description |
|---|---|
| b(7:4) | *Reserved* |
| b(3) | FPGA CRC16 error inject:<br>• '0': No error injection<br>• '1': Error injection |
| b(2) | FPGA CRC16 calculation:<br>• '0': Disable CRC16 calculation<br>• '1': Enable CRC16 calculation |
| b(1) | FPGA reset:<br>• '0': No reset<br>• '1': Reset FPGA |

| Parameter Byte | Description |
|---|---|
| b(0) | FPGA reset unlock:<br>• '0': Don't allow reset<br>• '1': Allow reset |

### 3.3.16 Read FPGA SPI CRC16 (CEh)

This command returns the FPGA SPI CRC16 results for the pixel data stream. The FPGA CRC16 calculation must be enabled within Section 3.3.14 before sending SPI data to receive a valid return on this command.

#### 3.3.16.1 Read Parameters

This command has no parameters.

#### 3.3.16.2 Return Parameters

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | FPGA SPI CRC16 value | LSByte |
| Byte 2 | FPGA SPI CRC16 value | MSByte |

## 3.4 General Setup Commands

### 3.4.1 Administrative Commands

#### *3.4.1.1 Read Short Status (D0h)*

This command provides a short system status.

##### 3.4.1.1.1 Read Parameters

This command has no parameters.

##### 3.4.1.1.2 Return Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Short System Status |

**Byte 1**

b(7) Boot/Main Application
- '0': Boot
- '1': Main

b(6) 3D Print Sequence Error
- '0': No Error
- '1': Error

b(5) Flash Error
- '0': No Error
- '1': Error

b(4) Flash Erase Complete
- '0': Complete
- '1': Not Complete

b(3) System Error
- '0': No Error
- '1': Error

b(2) *Reserved*
- '0': No Error
- '1': Error

b(1) Communication Error
- '0': No Error
- '1': Error

b(0) System Initialization
- '0': Not Complete
- '1': Complete

1. The controller sets the *Flash Erase Complete* status bit at the start of the Flash erase process, and clears the bit when it completes the erase process. The flash status can be obtained during or after the erase process. To obtain this status during the erase process, only this command can be sent after the start of the flash erase. If the user sends any other command during the erase process, the controller holds the command without processing until it completes the flash erase process (thus blocking any following status requests until it processes the previously sent command).
2. The *Flash Error* bit indicates an error during any Flash operation. For Flash writes, the controller updates this bit at the end of each write transaction. However, when the controller detects an error, this bit remains in the error state until cleared. This error state allows the user the option of checking the status between each write transaction, or at the end of the update. When a write transaction starts, the flash status (and this error bit) is not accessible until the write transaction has completed.
3. The *Communication Error* bit indicates an error on the $I^2C$ command interface. The *Read Communication Status* command gives specific details about communication errors.
4. The *System Error* bit indicates any errors other than *Flash Error* and *Communication Error*. The *Read Communication Status* command gives specific details about communication errors.
5. The controller clears the *Flash Error*, *Communication Error*, and *System Error* bits after executing the *Read Short Status* command.
6. When the *3D Print Sequence Error* bit is set, read the *System Status (D1h)* for specific errors in the selected External Print mode.
7. Check the *Read Short Status* command *periodically*, *not continuously*. Continuous access can severely degrade system performance.

### 3.4.1.2 Read System Status (D1h)

This command reads system status information.

#### 3.4.1.2.1 Read Parameters

This command has no parameters.

#### 3.4.1.2.2 Return Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | DMD interface status |
| Byte 2 | LED status |
| Byte 3 | Internal interrupt status |
| Byte 4 | Miscellaneous status |

All system status error bits are cleared by the *Read System Status* command.

| Byte 1 | Light Control and DMD interface status |
|---|---|
| b(7:3) | *Reserved* |
| b(2) | DMD Training Error<br>'0': No Error<br>'1': Error |
| b(1) | DMD Interface Error<br>'0': No Error<br>'1': Error |
| b(0) | DMD Device Error<br>'0': No Error<br>'1': Error |

1. The system sets the DMD Device Error for the following conditions:
   - The system cannot read the DMD Device ID from the DMD
   - The firmware specified DMD Device ID does not match the actual DMD Device ID
2. The system sets the DMD Interface Error when there are power management setup conflicts on this interface.
3. The system sets the DMD Training Error when the training algorithm is unable to find a data eye that meets the specified requirements.

| Byte 2 | LED Status | | |
|---|---|---|---|
| b(7:6) | *Reserved* | | |
| b(5) | Blue LED Error | b(2) | Blue LED State |
| b(4) | Green LED Error | b(1) | Green LED State |
| b(3) | Red LED Error<br>'0': No Error<br>'1': Error | b(0) | Red LED State<br>'0': Off<br>'1': On |

| Byte 3 | Internal Interrupt Status |
|---|---|
| b(7:3) | Light Control Error (Only applicable for Light Control Operating Modes)<br>0h: No Error<br>1h: illumination time not supported<br>2h: Pre-illumination dark time not supported<br>3h: Post-illumination dark time not supported<br>4h: Trigger Out 1 delay not supported<br>5h: Trigger Out 2 delay not supported |
| b(2) | DC Power Supply<br>'0': Supply voltage is in normal range<br>'1': Supply voltage is low |
| b(1) | Sequence Error<br>'0': No Error<br>'1': Error |
| b(0) | Sequence Abort Error<br>'0': No Error<br>'1': Error |

The DC power supply voltage status reported in b(2) is based on DLPA300x PMIC chip monitoring of the DC power supply voltage by using the chip BAT_LOW_WARN feature. The status of b(2) matches the value of the BAT_LOW_WARN bit in the DLPA300x PMIC chip status interrupt register.

| Byte 4 | Miscellaneous Status | | |
|---|---|---|---|
| b(7) | *Reserved*<br>'0': No Error<br>'1': Error | b(3) | Primary or Secondary Operation<br>'0': Primary<br>'1': Secondary |
| b(6) | Actuator Subframe Error<br>'0': No Error<br>'1': Error | b(2) | Single or Dual Controller Configuration<br>'0': Single<br>'1': Dual |
| b(5) | '0': No Error<br>'1': Error | b(1) | '0': No Error<br>'1': Error |
| b(4) | Product Configuration Error<br>'0': No Error<br>'1': Error | b(0) | '0': No Error<br>'1': Error |

1. The system sets the Primary or Secondary bit as appropriate in both single and dual controller configurations.
2. The system sets the Product Configuration Error bit if it determines that some piece of the product configuration is not correct. Some examples are:
   - Invalid Controller and DMD Combination
   - Invalid Controller and DLPAx000 Combination
   - Invalid Flash build for current Controller, DMD, or DLPAx000 configuration
3. The system sets the Actuator Subframe Error bit if the system detects AWG_ERROR condition.

### 3.4.1.3 Read System Software Version (D2h)

This command reads the main application software version information.

#### 3.4.1.3.1 Read Parameters

This command has no parameters.

#### 3.4.1.3.2 Return Parameters

| Parameter Bytes | Description | | |
|---|---|---|---|
| Byte 1 | Controller main application software version | Patch | LSByte |
| Byte 2 | Controller main application software version | Patch | MSByte |
| Byte 3 | Controller main application software version | Minor | |
| Byte 4 | Controller main application software version | Major | |

### 3.4.1.4 Read Communication Status (D3h)

#### 3.4.1.4.1 Read Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Command bus selection |

| Byte 1 | Command bus selection | | |
|---|---|---|---|
| b(7:2) | *Reserved* | b(1:0) | '10': $I^2C$ |

#### 3.4.1.4.2 Return Parameters

| Parameter Bytes | Description |
|---|---|
| Bytes 1 - 4 | *Reserved* |
| Byte 5 | $I^2C$ Communication status |
| Byte 6 | Aborted or Invalid Command op-code |

The system clears all communication status error bits when this command is read.

| Byte 5 | $I^2C$ Communication status | | |
|---|---|---|---|
| b(7) | *Reserved*<br>'0': No Error<br>'1': Error | b(3) | Flash Batch File Error<br>'0': No Error<br>'1': Error |
| b(6) | $I^2C$ Bus Timeout Error<br>'0': No Error<br>'1': Error | b(2) | Command Processing Error<br>'0': No Error<br>'1': Error |
| b(5) | Invalid Number of Command Parameters<br>'0': No Error<br>'1': Error | b(1) | Invalid Command Parameter Value<br>'0': No Error<br>'1': Error |
| b(4) | Read Command Error<br>'0': No Error<br>'1': Error | b(0) | Invalid Command Error<br>'0': No Error<br>'1': Error |

1. The system sets the Invalid Command Error bit when it does not recognize the command op-code. Byte 6 contains the invalid op-code received by the controller.
2. The system sets the Invalid Command Parameter Error bit when the it detects that the value of a command parameter is not valid (e.g. out of allowed range).
3. The system sets the Command Processing Error bit when a fault is detected while processing a command. In this case, the system aborts the command and moves to the next command. Byte 6 contains the op-code of the aborted command.
4. The system sets the Flash Batch File Error bit when an error occurs during the processing of a flash batch file. When the systems sets this bit, it typically sets another bit to indicate what kind of error was detected (such as: Invalid Command Error).
5. The system sets the Read Command Error bit when the host terminates the read operation before all of the requested data has been provided, or if the host continues to request read data after all of the requested data has been provided.
6. The system sets the Invalid Number of Command Parameters Error bit when too many or too few command parameters are received. In this case, the system aborts the command and moves to the next command. Byte 6 contains the op-code of the aborted command.
7. The system sets the Bus Timeout Error bit when it releases control of the bus because the bus timeout value was exceeded.

### 3.4.1.5 Read Controller Device ID (D4h)

This command reads the Controller Device ID.

#### 3.4.1.5.1 Read Parameters

This command has no parameters.

#### 3.4.1.5.2 Return Parameters

| Byte 1 | |
|---|---|
| b(7:4) | *Reserved* |
| b(3:0) | Controller Device ID |

Unused controller device ID values are reserved.

### 3.4.1.6 Read DMD Device ID (D5h)

This command reads the DMD device ID.

#### 3.4.1.6.1 Read Parameters

| Byte 1 | DMD Register Selection |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | '0': DMD Device ID |
| | '1': *Reserved* |

#### 3.4.1.6.2 Return Parameters

| Parameter Bytes | Description |
|---|---|
| Bytes 1 – 4 | See Table 3-9 |

**Table 3-9. DMD Device ID Reference Table**

| DMD Device ID | | | | Device Description |
|---|---|---|---|---|
| **Byte 1 (Identifier)** | **Byte 2 (Byte Count)** | **Byte 3 (ID-MSByte)** | **Byte 4 (ID-LSByte)** | **Resolution and Type** |
| 60h | 0Dh | 00h | 68h / 72h / 87h | 0.3 720p (1280x720, Sub-LVDS) |

### 3.4.1.7 Read System Temperature (D6h)

This command is used to read the system temperature using an external thermistor (if available).

#### 3.4.1.7.1 Read Parameters

The command has no read parameters.

#### 3.4.1.7.2 Read Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | LSByte |
| Byte 2 | MSByte |

Figure 3-9 shows the bit order and definition for the signed magnitude system temperature data, which is returned in degrees C. The unspecified msbits (bits 15:12) is set to '0'.

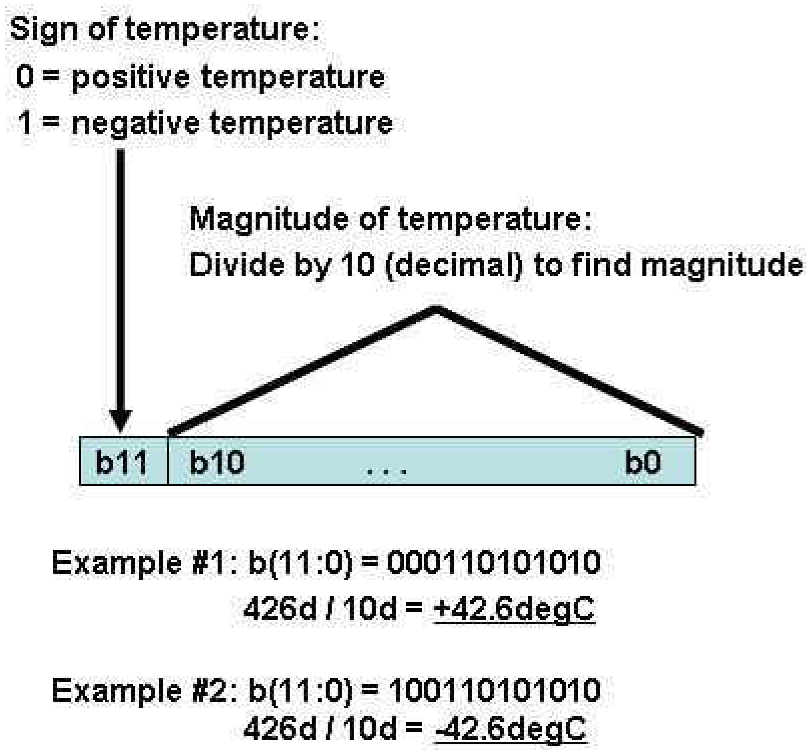



**Figure 3-9. Bit Order and Definition for System Temperature**

### 3.4.1.8 Read Flash Build Version (D9h)

This command reads the controller flash version.

#### 3.4.1.8.1 Read Parameters

The command has no read parameters.

#### 3.4.1.8.2 Return Parameters

| Parameter Bytes | Description | | |
|---|---|---|---|
| Byte 1 | Flash build version | Patch | LSByte |
| Byte 2 | Flash build version | Patch | MSByte |
| Byte 3 | Flash build version | Minor | |
| Byte 4 | Flash build version | Major | |

#### 3.4.1.9 Write Flash Batch File Delay (DBh)

This command is used to specify an execution time delay within a Flash batch file.

##### 3.4.1.9.1 Write Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Flash Batch File Delay (LSB) |
| Byte 2 | Flash Batch File Delay (MSB) |

1. The Flash batch file delay is to be specified in units of 1ms (e.g. 500ms = 01F4h ).
2. This command is used to specify an execution delay time within a Flash batch file. It can only be used within a Flash batch file, and is not a valid command on the I2C interface.
3. Typical use of this command is in the Auto-Init Flash batch file (batch file 0), but is valid for use in any batch file (See Section 3.1.14).

## 3.5 FPGA Commands

This section covers the commands issued through the DLPC1438 to the FPGA. The FPGA on the system is tasked with formatting external test patterns, actuator control, and SPI pixel data streaming.

### 3.5.1 Read FPGA Version (64h)

The command Reads the version number of the currently flashed FPGA firmware.

#### 3.5.1.1 Read Parameters

This command has no read parameters.

#### 3.5.1.2 Return Parameters

| Parameter Bytes | Description | | |
|---|---|---|---|
| Byte 1 | FPGA Build Version | Patch | LSByte |
| Byte 2 | FPGA Build Version | Patch | MSByte |
| Byte 3 | FPGA Build Version | Minor | |
| Byte 4 | FPGA Build Version | Major | |

### 3.5.2 Read FPGA Status (6Fh)

This command reads the FPGA main status.

#### 3.5.2.1 Read Parameters

This command has no parameters.

#### 3.5.2.2 Return Parameters

| Parameter Bytes | Description |
|---|---|
| b(15:11) | *Reserved* |
| b(10) | Read FIFO full<br>• 0: FIFO not full<br>• 1: FIFO full |
| b(9:6) | *Reserved* |
| b(5) | Write FIFO full<br>• 0: FIFO not full<br>• 1: FIFO full |
| b(4) | Actuator waveform error<br>• 0: No error<br>• 1: Error |
| b(3) | SPI CRC error<br>• 0: No error<br>• 1: Error |

| Parameter Bytes | Description |
| --- | --- |
| b(2) | Memory initialization error<br>• 0: No error<br>• 1: Error |
| b(1) | Mem PLL locked<br>• 0: No error<br>• 1: Error |
| b(0) | Video PLL locked<br>• 0: No error<br>• 1: Error |

### 3.5.3 Write FPGA TPG (67h)

This command configures the FPGA test pattern parameters. This pattern can be used by putting the DLPC1438 into external print mode through the Operating Mode Select (see Section 3.1.1), setting the External Print Configuration (see Section 3.3.6), and writing the External Print Layer Control (see Section 3.3.8). The external FPGA TPG mode is primarily intended for debugging. This command and Write Parallel Video (see Section 3.3.10) cannot enable at the same time. In order to disable this command, switch the DLPC1438 into Standby mode through the Operating Mode Select (see Section 3.1.1).

#### *3.5.3.1 Write Parameters*

| Parameter Bytes | Description |
| --- | --- |
| Byte 1 | Test Pattern Generator (TPG) and border enable |
| Byte 2 | Test Pattern select |

| Byte 1 | Description |
| --- | --- |
| b(7:2) | *Reserved* |
| b(1) | Border enable:<br>• 0: Border disabled<br>• 1: Border enabled |
| b(0) | TPG enable:<br>• 0: Disabled<br>• 1: Enabled |

| Byte 2 | Description |
| --- | --- |
| b(7:4) | *Reserved* |
| b(3:0) | Pattern select<br>• 0x4: Cycle Actuator Calibration Pattern 0-3<br>• 0x8: Grid<br>• 0x9: Horizontal Ramp<br>• 0xA: Vertical Ramp<br>• 0xB: Checkerboard<br>• 0xC: Frame and Cross<br>• 0xD: Horizontal Lines<br>• 0xE: Vertical Lines<br>• 0xF: Grid, 45 Degrees |

### 3.5.4 Read FPGA TPG (68h)

This command reads the FPGA Test Pattern Generator (TPG) configuration parameters.

#### *3.5.4.1 Read Parameters*

This command contains no parameters.

#### 3.5.4.2 Return Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Test Pattern Generator (TPG) and border enable |
| Byte 2 | Test Pattern select |

| Byte 1 | Description |
|---|---|
| b(7:2) | *Reserved* |
| b(1) | Border enable:<br>• 0: Border disabled<br>• 1: Border enabled |
| b(0) | TPG enable:<br>• 0: Disabled<br>• 1: Enabled |

| Byte 2 | Description |
|---|---|
| b(7:4) | *Reserved* |
| b(3:0) | Pattern select<br>• 0x4: Cycle Actuator Calibration Pattern 0-3<br>• 0x8: Grid<br>• 0x9: Horizontal Ramp<br>• 0xA: Vertical Ramp<br>• 0xB: Checkerboard<br>• 0xC: Frame and Cross<br>• 0xD: Horizontal Lines<br>• 0xE: Vertical Lines<br>• 0xF: Grid, 45 Degrees |

### 3.5.5 Actuator Commands

This section covers the commands issued to the FPGA through the DLPC1438 to set up and control the actuator.

#### 3.5.5.1 Write Actuator Latency (70h)

This command writes the actuator latency.

##### 3.5.5.1.1 Write Parameters

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Delay for actuator sync GPIO in clock ticks, with a step of 133.333 ns | LSByte |
| Byte 2 | Delay for actuator sync GPIO in clock ticks, with a step of 133.333 ns | |
| Byte 3 | Delay for actuator sync GPIO in clock ticks, with a step of 133.333 ns | |
| Byte 4 | Delay for actuator sync GPIO in clock ticks, with a step of 133.333 ns | MSByte |
| Byte 5 | Actuator latency auto-scaling | |

| Byte 3 | Actuator Latency Auto-Scaling |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | Scaling enable<br>• 0: No scaling<br>• 1: Scaling |

#### 3.5.5.2 Read Actuator Latency (71h)

This command reads the actuator latency.

#### 3.5.5.2.1 Read Parameters

This command has no parameters.

#### 3.5.5.2.2 Return Parameters

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Delay for actuator sync GPIO in clock ticks, with a step of 133.333 ns | LSByte |
| Byte 2 | Delay for actuator sync GPIO in clock ticks, with a step of 133.333 ns | |
| Byte 3 | Delay for actuator sync GPIO in clock ticks, with a step of 133.333 ns | |
| Byte 4 | Delay for actuator sync GPIO in clock ticks, with a step of 133.333 ns | MSByte |
| Byte 5 | Actuator latency auto-scaling | |

| Byte 3 | Actuator Latency Auto-Scaling |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | Scaling enable<br>• 0: No scaling<br>• 1: Scaling |

### *3.5.5.3 Write Actuator Gain (72h)*

This command writes the actuator gain value.

#### 3.5.5.3.1 Write Parameters

| Parameter Byte | Description |
|---|---|
| Byte 1 | Actuator gain value |

1. This parameter has a range of 0 to 1.9921875.
2. 0x80 represents a gain of 1.

### *3.5.5.4 Read Actuator Gain (73h)*

This command reads the actuator gain value.

#### 3.5.5.4.1 Read Parameters

This command has no parameters.

#### 3.5.5.4.2 Return Parameters

| Parameter Byte | Description |
|---|---|
| Byte 1 | Actuator gain value |

1. This parameter has a range of 0 to 1.9921875.
2. 0x80 represents a gain of 1.

### *3.5.5.5 Write Actuator Segment Length (74h)*

This command writes the actuator segment length.

#### 3.5.5.5.1 Write Parameters

The parameters define the duration of each step (level) for the actuator function.

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Actuator segment length value | LSByte |
| Byte 2 | Actuator segment length value | MSByte |

1. This parameter can range from 17 to 4095.

### *3.5.5.6 Read Actuator Segment Length (75h)*

This command reads the actuator segment length value.

#### 3.5.5.6.1 Read Parameters

This command has no parameters.

#### 3.5.5.6.2 Return Parameters

The parameters define the duration of each step (level) for the actuator function.

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Actuator segment length value | LSByte |
| Byte 2 | Actuator segment length value | MSByte |

1. This parameter can range from 17 to 4095.

### *3.5.5.7 Write Actuator Subframe Delay (76h)*

This command writes the actuator subframe delay.

#### 3.5.5.7.1 Write Parameters

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Delay for actuator subframe | LSByte |
| Byte 2 | Delay for actuator subframe | |
| Byte 3 | Delay for actuator subframe | |
| Byte 4 | Delay for actuator subframe | MSByte |

1. The delay for actuator subframe is in clock ticks.
2. There is a step every 0.01984 micro-seconds.

### *3.5.5.8 Read Actuator Subframe Delay (77h)*

This command reads the actuator subframe delay.

#### 3.5.5.8.1 Read Parameters

This command has no parameters.

#### 3.5.5.8.2 Return Parameters

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Delay for actuator subframe | LSByte |
| Byte 2 | Delay for actuator subframe | |
| Byte 3 | Delay for actuator subframe | |
| Byte 4 | Delay for actuator subframe | MSByte |

1. The delay for actuator subframe is in clock ticks.
2. There is a step every 0.01984 micro-seconds.

### *3.5.5.9 Write Actuator Offset (78h)*

This command writes the actuator offset value.

#### 3.5.5.9.1 Write Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Actuator offset |

1. This parameter is presented in 8-bit signed format, and has a range of -128 to 127. 2's compliment format is used for negative value.

### *3.5.5.10 Read Actuator Offset (79h)*

This command reads the actuator offset value.

#### 3.5.5.10.1 Read Parameters

This command has no parameters.

#### 3.5.5.10.2 Return Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Actuator offset |

1. This parameter is presented in 8-bit signed format, and has a range of -128 to 127. 2's compliment format is used for negative value.

### *3.5.5.11 Write Actuator Configuration Select (A2h)*

This command selects AWC0 on axis 0 or AWC1 on axis 1 for subsequent configuration commands.

#### 3.5.5.11.1 Write Parameters

| Parameter Byte | Description |
|---|---|
| Byte 1 | Actuator waveform generator select |

| Byte 1 | Description |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | Actuator waveform axis selection:<br>• 0: Axis 0<br>• 1: Axis 1 |

### *3.5.5.12 Read Actuator Configuration Select (A3h)*

This command returns the selected AWC for subsequent configuration.

#### 3.5.5.12.1 Read Parameters

This command has no parameters.

#### 3.5.5.12.2 Return Parameters

| Parameter Byte | Description |
|---|---|
| Byte 1 | Actuator waveform generator select |

| Byte 1 | Description |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | Actuator waveform axis selection:<br>• 0: Axis 0<br>• 1: Axis 1 |

### *3.5.5.13 Write Actuator Fixed Output Level (A4h)*

This command writes actuator fixed output value. This value defines the level that is output on the PWM or DAC data bus when the fixed output enable is selected (see Section 3.5.5.17).

**Note**

Fixed actuator output for PWM configurations will not appear until the controller is put into External Print mode when the value stored will be applied. Before entering the mode, the value will be 0 on the output.

#### 3.5.5.13.1 Write Parameters

| Parameter Byte | Description |
|---|---|
| Byte 1 | Actuator fixed output value |

1. This parameter is presented in 8-bit signed format, and has a range of -128 to 127. 2's compliment format is used for negative value.

#### 3.5.5.14 Read Actuator Fixed Output Level (A5h)

This command reads the actuator fixed output value.

##### 3.5.5.14.1 Read Parameters

This command has no parameters.

##### 3.5.5.14.2 Return Parameters

| Parameter Byte | Description |
|---|---|
| Byte 1 | Actuator fixed output value |

1. This parameter is presented in 8-bit signed format, and has a range of -128 to 127. 2's compliment format is used for negative value.

#### 3.5.5.15 Write Actuator Number of Segments (A6h)

This command defines the actuator number of segments.

##### 3.5.5.15.1 Write Parameters

| Parameter Byte | Description |
|---|---|
| Byte 1 | Actuator Number of Segments value 1 |

1. This number is range of 2 to 255 in steps of 1.
2. Actuator Number of Segments settings are only applied when the user sends *Write Operating Mode Select* to *Print External Layer* (see Section 3.1.1). The Actuator Number of Segments value is sent when *Write Operating Mode Select - External Print Mode* is called. If this command is not sent, the software stores the settings until it is called.

#### 3.5.5.16 Read Actuator Number of Segments (A7h)

This command returns the Number of Segments selection.

##### 3.5.5.16.1 Read Parameters

This command has no parameters.

##### 3.5.5.16.2 Return Parameters

| Parameter Byte | Description |
|---|---|
| Byte 1 | Actuator Number of Segments value 1 |

1. This number is range of 2 to 255 in steps of 1.

#### 3.5.5.17 Write Actuator Output Select (AAh)

This command writes the actuator output select type. To set the fixed output level see Section 3.5.5.13.

##### 3.5.5.17.1 Write Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Fixed Output Enable |

| Byte 1 | Description |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | Fixed output enable:<br>• 0: Fixed output disable<br>• 1: Fixed output enable |

**Note**

MCU and EEPROM actuators may have the fixed output pre-configured. If this is the case, the settings stored in the actuator will be loaded once the DLPC is put into External Print Mode (Section 3.1.1). It is recommended to issue this command after entering into External Print Mode.

### *3.5.5.18 Read Actuator Output Select (ABh)*

This command reads the actuator output select type.

#### 3.5.5.18.1 Read Parameters

This command has no parameters.

#### 3.5.5.18.2 Return Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Fixed Output Enable |

| Byte 1 | Description |
|---|---|
| b(7:1) | *Reserved* |
| b(0) | Fixed output enable:<br>• 0: Fixed output disable<br>• 1: Fixed output enable |

### *3.5.5.19 Read Actuator Control (AFh)*

This command reads the actuator output enable fields.

#### 3.5.5.19.1 Read Parameters

This command has no parameters.

#### 3.5.5.19.2 Return Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Actuator type |
| Byte 2 | Actuator output select |

| Byte 1 | Description |
|---|---|
| b(7:2) | *Reserved* |
| b(1:0) | Actuator type:<br>• 0x0: None<br>• 0x1: Standalone<br>• 0x2: TI Common Actuator Interface (MCU)<br>• 0x3: TI Common Actuator Interface (EEPROM) |

| Byte 2 | Description |
|---|---|
| b(7:2) | *Reserved* |
| b(1) | Actuator waveform function<br>• 0: Disabled<br>• 1: Enabled |

### *3.5.5.20 Read Actuator Temperature (C7h)*

This command reads the actuator temperature for TI common actuators only.

#### 3.5.5.20.1 Read Parameters

This command has no parameters.

#### 3.5.5.20.2 Return Parameters

| Parameter Bytes | Description | |
|---|---|---|
| Byte 1 | Actuator temperature in degrees C | LSByte |
| Byte 2 | Actuator temperature in degrees C | MSByte |
| Byte 3 | Actuator illumination type | |
| Byte 4 | Number of configuration updates | LSByte |
| Byte 5 | Number of configuration updates | MSByte |
| Byte 6 | Temperature used for configuration | LSByte |
| Byte 7 | Temperature used for configuration | MSByte |

### *3.5.5.21 Write Actuator Orientation (C8h)*

This command writes the actuator subframe orientation for pixel data and actuator positions.

#### 3.5.5.21.1 Write Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Actuator subframe order 1 |
| Byte 2 | Pixel subframe order without flip 2 |
| Byte 3 | Pixel subframe order with long axis flip3 |
| Byte 4 | Pixel subframe order with short axis flip4 |
| Byte 5 | Pixel subframe order with both axis flip5 |

1. Defines enumeration value for actuator subframe position. Available actuator subframe order: 0x1, 0x3, 0x6, 0xb, 0xc, 0x11, 0x14, 0x16
2. Defines enumeration value for pixel subframe position with no flips. Available pixel subframe order with no flip: 0x1, 0x3, 0x6, 0xb, 0xc, 0x11, 0x14, 0x16
3. Defines enumeration value for pixel subframe position with long axis flip. Available pixel subframe order with long axis flip: 0x1, 0x3, 0x6, 0xb, 0xc, 0x11, 0x14, 0x16
4. Defines enumeration value for pixel subframe position with short axis flip. Available pixel subframe order with short axis flip: 0x1, 0x3, 0x6, 0xb, 0xc, 0x11, 0x14, 0x16
5. Defines enumeration value for pixel subframe position with both axes flipped. Available pixel subframe order with both axis flip: 0x1, 0x3, 0x6, 0xb, 0xc, 0x11, 0x14, 0x16

### *3.5.5.22 Read Actuator Orientation (C9h)*

This command reads the actuator subframe order for pixel data and actuator positions.

#### 3.5.5.22.1 Read Parameters

This command has no parameters.

#### 3.5.5.22.2 Return Parameters

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Actuator subframe order 1 |
| Byte 2 | Pixel subframe order without flip 2 |
| Byte 3 | Pixel subframe order with long axis flip 3 |
| Byte 4 | Pixel subframe order with short axis flip 4 |
| Byte 5 | Pixel subframe order with both axis flip 5 |

1. Defines enumeration value for actuator subframe position. Available actuator subframe order: 0x1, 0x3, 0x6, 0xb, 0xc, 0x11, 0x14, 0x16
2. Defines enumeration value for pixel subframe position with no flips. Available pixel subframe order with no flip: 0x1, 0x3, 0x6, 0xb, 0xc, 0x11, 0x14, 0x16
3. Defines enumeration value for pixel subframe position with long axis flip. Available pixel subframe order with long axis flip: 0x1, 0x3, 0x6, 0xb, 0xc, 0x11, 0x14, 0x16
4. Defines enumeration value for pixel subframe position with short axis flip. Available pixel subframe order with short axis flip: 0x1, 0x3, 0x6, 0xb, 0xc, 0x11, 0x14, 0x16

5. Defines enumeration value for pixel subframe position with both axes flipped. Available pixel subframe order with both axis flip: 0x1, 0x3, 0x6, 0xb, 0xc, 0x11, 0x14, 0x16

## 3.6 Flash Update Commands

This section describes the commands required to update internal pattern data stored in flash. The following steps must be followed to successfully update pattern data:

1. Generate your pattern data using the DLP® Display and Light Control EVM GUI. The .bin file generated by the "Save Pattern Data" button in GUI contains the pattern data to be sent through I2C.
2. Ensure internal patterns are not currently running.
3. Set flash data type to pattern data block using *Write Flash Data Type Select (DEh)* command.
4. Check whether the generated .bin file can fit within the currently existing pattern data flash block using the *Read Flash Update PreCheck (DDh)* command.
5. Erase existing pattern data using *Write Erase Flash Data (E0h)* command.
6. Set flash data length to 1024 bytes using the *Write Flash Data Length (DFh)* command.
7. Write the first 1024 bytes of pattern data using *Write Flash Start (E1h)* command.
8. Write the remaining data in blocks of 1024 bytes using the *Write Flash Continue (E2h)* command.

**Note**

If the size of the pattern data is not a multiple of 1024 bytes, the last Write Flash Continue command has less than 1024 bytes. In that case, use the *Write Flash Data Length (DFh)* command to update the Flash Data Length prior to the last Write Flash Continue command.

Note that the flash commands described in this section cannot be used within batch files.

### 3.6.1 Read Flash Update PreCheck (DDh)

This command is used to verify that a pending flash update is appropriate for the specified flash data type.

#### *3.6.1.1 Read Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Flash Build Data Size (LSB) |
| Byte 2 | Flash Build Data Size |
| Byte 3 | Flash Build Data Size |
| Byte 4 | Flash Build Data Size (MSB) |

#### *3.6.1.2 Return Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Flash PreCheck Result |

**Byte 1 - Flash PreCheck Result**

b(0) Package Size Error
'0': No Error
'1': Error

b(7:1) *Reserved*

1. This command is used in conjunction with the *Write Flash Data Type Select (DEh)* command. This command is sent after the flash data type has been selected, but before any other flash operation. The purpose is to verify that the desired flash update is compatible and fits within the existing flash space for the current flash configuration.
2. The Flash Build Data Size specifies the size of the flash update package in bytes.
3. A Package Size error indicates that the flash package is too large to fit into the specified location. For example, this bit is set if the size of the new internal pattern data is larger than that of the data currently present in flash.
4. If an error is returned by this command, the user is responsible for correcting the error before updating the flash. If the user chooses to ignore this error and proceed with updating the flash, the system allows this. In this case, the user is responsible for any problems or system behaviors that arise thereafter.

### 3.6.2 Write Flash Data Type Select (DEh)

This command is used to specify the type of data that is written to the Flash. In DLPC143x the only flash data type supported for update through I2C is internal pattern data.

#### *3.6.2.1 Write Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Flash Data Type (D0h for pattern data) |
| Byte 2 | (00h for pattern data) |
| Byte 3 | (00h for pattern data) |
| Byte 4 | (00h for pattern data) |

### 3.6.3 Write Flash Data Length (DFh)

This command is used to specify the length of the data that is written to the Flash in bytes.

#### *3.6.3.1 Write Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Flash Data Length (LSB) |
| Byte 2 | Flash Data Length (MSB) |

Default: 0000h

1. Flash data length must be in multiples of four bytes.
2. The flash data length applies to each write transaction, not to the length of the data type selected.
3. The maximum data length allowed for each write transaction is 1024 bytes.

### 3.6.4 Write Erase Flash Data (E0h)

This command erases the specified block of data in Flash.

#### *3.6.4.1 Write Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Signature: Value = AAh |
| Byte 2 | Signature: Value = BBh |
| Byte 3 | Signature: Value = CCh |
| Byte 4 | Signature: Value = DDh |

1. When this command is executed, the system erases all sectors associated with the data type specified by the *Write Flash Data Type Select* (Section 3.6.2) command. As such, this command does not make use of the Flash Data Length parameter.
2. The signature bytes are used to minimize unintended flash erases. The command OpCode and four signature bytes must be received correctly before this command is recognized and executed.

**Note**

Since the process of erasing Flash sectors can take a significant amount of time, the Flash Erase Complete status bit in the *Read Short Status* command (Section 3.4.1.1) is to be checked periodically (not continuously) to determine when this task has been completed. This bit is set at the start of the erase process and is cleared when the erase process is complete. Flash writes are not to be started before the erase process has been completed.

### 3.6.5 Write Flash Start (E1h)

This command is used to start writing data to Flash.

#### *3.6.5.1 Write Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Data Byte 1 |
| Byte 2 | Data Byte 2 |
| Byte 3 | Data Byte 3 |
| Byte 4 | Data Byte 4 |
| Byte 5 … n | Data Byte 5 … n |

1. The *Write Flash Data Length* command must be used to specify how many bytes of data are sent by this command.
2. The *Write Flash Start* command is used to write up to 1024 bytes of data starting at the first address of the data type selected. If more than 1024 bytes are to be written, the *Write Flash Continue* command must be used. Up to 1024 bytes of data can be written with each *Write Flash Continue* command, which starts at the end of the last data written.
3. The Flash Error bit of the *Read Short Status* command indicate if the Flash update was successful. This bit sets for an error at the end of each write transaction, however, once an error has been detected, this bit remains in the error state until a new data type is selected (selecting a new data type clears this bit). This allows the user the option of checking the status between each write transaction, or at the end of the update of a specific data type. Once a write transaction has started, the flash status (and this error bit) is not accessible until the write transaction has completed.

### 3.6.6 Write Flash Continue (E2h)

This command is used if more than 1024 bytes of data has to be written to Flash.

#### *3.6.6.1 Write Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Data Byte 1 |
| Byte 2 | Data Byte 2 |
| Byte 3 | Data Byte 3 |
| Byte 4 | Data Byte 4 |
| Byte 5 … n | Data Byte 5 … n |

1. The *Write Flash Data Length* command must be used to specify how many bytes of data are sent by this command.
2. The *Write Flash Start* command is used to write up to 1024 bytes of data starting at the first address of the data type selected. If more than 1024 bytes are to be written, the *Write Flash Continue* command must be used. Up to 1024 bytes of data can be written with each *Write Flash Continue* command, which starts at the end of the last data written.
3. The Flash Error bit of the *Read Short Status* command indicates if the Flash update was successful. This bit is set for an error at the end of each write transaction, however, once an error has been detected, this bit remains in the error state until a new data type is selected (selecting a new data type clears this bit). This allows the user the option of checking the status between each write transaction, or at the end of the update of a specific data type. Once a write transaction has started, the flash status (and this error bit) is not accessible until the write transaction has completed.

### 3.6.7 Read Flash Start (E3h)

This command is used to start reading data from Flash.

#### *3.6.7.1 Read Parameters*

The command has no read parameters.

#### *3.6.7.2 Return Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Data Byte 1 |
| Byte 2 | Data Byte 2 |
| Byte 3 | Data Byte 3 |
| Byte 4 | Data Byte 4 |
| Byte 5 … n | Data Byte 5 … n |

1. The *Write Flash Data Length* command must be used to specify how many bytes of data are sent by this command.
2. The *Read Flash Start* command is used to read up to 256 bytes of data starting at the first address of the data type selected. If more than 256 bytes are to be read, the *Read Flash Continue* command must be used. Up to 256 bytes of data can be read with each *Read Flash Continue* command, which starts at the end of the last data written.

### 3.6.8 Read Flash Continue (E4h)

This command is used if more than 256 bytes of data has to be read from Flash.

#### *3.6.8.1 Read Parameters*

The command has no read parameters.

#### *3.6.8.2 Return Parameters*

| Parameter Bytes | Description |
|---|---|
| Byte 1 | Data Byte 1 |
| Byte 2 | Data Byte 2 |
| Byte 3 | Data Byte 3 |
| Byte 4 | Data Byte 4 |
| Byte 5 … n | Data Byte 5 … n |

1. The *Write Flash Data Length* command must be used to specify how many bytes of data are sent by this command.
2. The *Read Flash Start* command is used to read up to 256 bytes of data starting at the first address of the data type selected. If more than 256 bytes are to be read, the *Read Flash Continue* command must be used. Up to 256 bytes of data can be read with each *Read Flash Continue* command, which starts at the end of the last data written.

# 4 SPI Data Transmission

This section covers the formatting for proper SPI data transmission from a SPI transmitter to the FPGA on a DLPC1438 system. The topics for this section include:

1. SPI Pixel Video Specifications (see Section 4.1)
2. SPI Data Formatting (see Section 4.2)

## 4.1 SPI Pixel Video Specifications

The external print data stream to the parallel data pins on the DLPC1438 needs to be properly formatted. The front end video source is to have the following settings:

- Horizontal back porch: 645
- Horizontal front porch: 645
- Vertical back porch: 107
- Vertical front porch: 107
- Pixels per line: 1280
- Lines per frame: 720

## 4.2 SPI Data Formatting

The incoming data sent over the SPI interface must be formatted in the following method for proper interpretation from the FPGA:

**Table 4-1. SPI Data Stream Format**

| Layout: | Command | Row/Col Index | Dummy | Length 1 | Data Field 2 | CRC16 3 |
|---|---|---|---|---|---|---|
| **Bits:** | 0 - 7 | 8 - 39 | 40 - 47 | 48 - 79 | 80 -336/16.777297E6 | 337/16.777298E6 - 369/16.777329E6 |

1. The length field is the hexadecimal number of bytes of the data field.
2. The data field has a minimum size of 256 bits and a maximum size of 16,777,216 bits.
3. The CRC16 is a 16 bit redundancy check. These bits are always be at the end of the datastream.

   For example, the DLP team at TI uses the mkCrcFun function from crcmod within Python scripts. The parameters used to generate the function are:
   Polynomial: 0x18005
   Initial value: 0xFFFF
   Final XOR value: 0x0
   Input reflected: False
   Result reflected: False
   Forward or reverse: Forward
   The FPGA SPI CRC16 results can be checked with Section 3.3.16
4. These command parameters are all sent in little endian formatting.
5. Due to limitations in SPI transmitter buffer size, it may be necessary to send these commands in smaller portions. For instructions on doing so please refer to Section 4.2.1.

**Table 4-2. SPI Data Stream Commands**

| Command Code | Description |
|---|---|
| 00 | Idle |
| 04 | Data stream |

The SPI Data stream may replace the entire image buffer, or may optionally update a rectangular windowed portion of the image buffer. Column Start Index sets the column of the top left pixel of the rectangle to be updated, which will occur at column number 128 x Column Start Index. Row Start Index sets the row of the top left pixel of the rectangle to be updated, which will occur at 2 x Row Start Index. Column End Index sets the rightmost column of the rectangle to be updated, which will occur at column number 128 x Column End Index - 1. Length determines the bottom right corner of the rectangle to be updated, and must be set at height x width x number of bits per pixel.

For example, to replace the entire 2560 x 1440 frame of 8-bit values, use the following settings:

Column Start Index: 0

Column End Index: 19

Start Row Index: 0

Length: 3686400

For example, to replace the 1280 x 720 frame of 8-bit values, use the following settings:

Column Start Index: 5

Column End Index: 14

Start Row Index: 180

Length: 921600

**Table 4-3. SPI Data Stream Row And Column Index Bits (8 - 39)**

| Bits | Description | Valid Values with DLPC1438 |
|---|---|---|
| b(8:12) | Column Start Index, column of top left pixel = 128 x Column Start Index | 0 to 19 |
| b(13:17) | Column End Index, rightmost column = 128 x Column End Index - 1 | 0 to 19, must be greater or equal than Column Start Index |
| b(18:28) | Start Row Index, row of top left pixel = 2 x Start Row Index | 0 to 719 |
| b(29:35) | All "0"s for dummy bits | |
| b(36:39) | All "1"s for data stream check | |

### 4.2.1 Dividing SPI Data

In the event of a command exceeding the front-end buffer size, the command can be reduced to a size that can be supported and sent in segments. In these segmented commands the command must always begin with the command opcode, the row and column index, and the dummy bits followed by the data. This can be repeated as necessary until the entirety of the command data has been sent with the CRC as the last two bytes.

This appears as follows:

**Table 4-4. SPI Data First Transmission**

| Command | Row/Col Index | Dummy Bits | Length 1 | Image Data 0 to send (x1) |
|---|---|---|---|---|
| 1 byte opcode | 4 bytes in little endian | 7 bits all "0"s | 4 bytes in little endian | Up to FPGA buffer size - 79 bits in little endian |

1. The length field is the hexadecimal number of bits of the entire data being sent for the image over all transmissions.

**Table 4-5. SPI Data Nth Transmission**

| Command | Row/Col Index | Dummy Bits | Image Data from (xN-1) to (xN) |
|---|---|---|---|
| 1 byte opcode | 4 bytes in little endian | 7 bits all "0"s | Up to SPI transmitter buffer size - 47 bits in little endian |

**Table 4-6. SPI Data Final Transmission**

| Command | Row/Col Index | Dummy Bits | Image Data (xN) to end | CRC16 |
|---|---|---|---|---|
| 1 byte opcode | 4 bytes in little endian | 7 bits all"0"s | Up to SPI transmitter buffer size - 63 bits in little endian | 2 bytes in little endian |