TI Information — Selective Disclosure

DLP3030-Q1 Head-Up Display (HUD) Piccolo SPI

User's Guide

Literature Number: DLPU057
March 2018

Contents

List of Figures

List of Tables

Introduction

Trademarks

DLP is a registered trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

Overview

The Piccolo LED Controller is designed to cost effectively support automotive DLP® technology LED applications requiring precise control of color and brightness over a wide dimming range. The Piccolo LED Controller is a sub-system level reference design that leverages many unique features of the Piccolo TMS320F28023 MCU, plus some special companion functions designed into the automotive DLPC120 DMD controller ASIC.

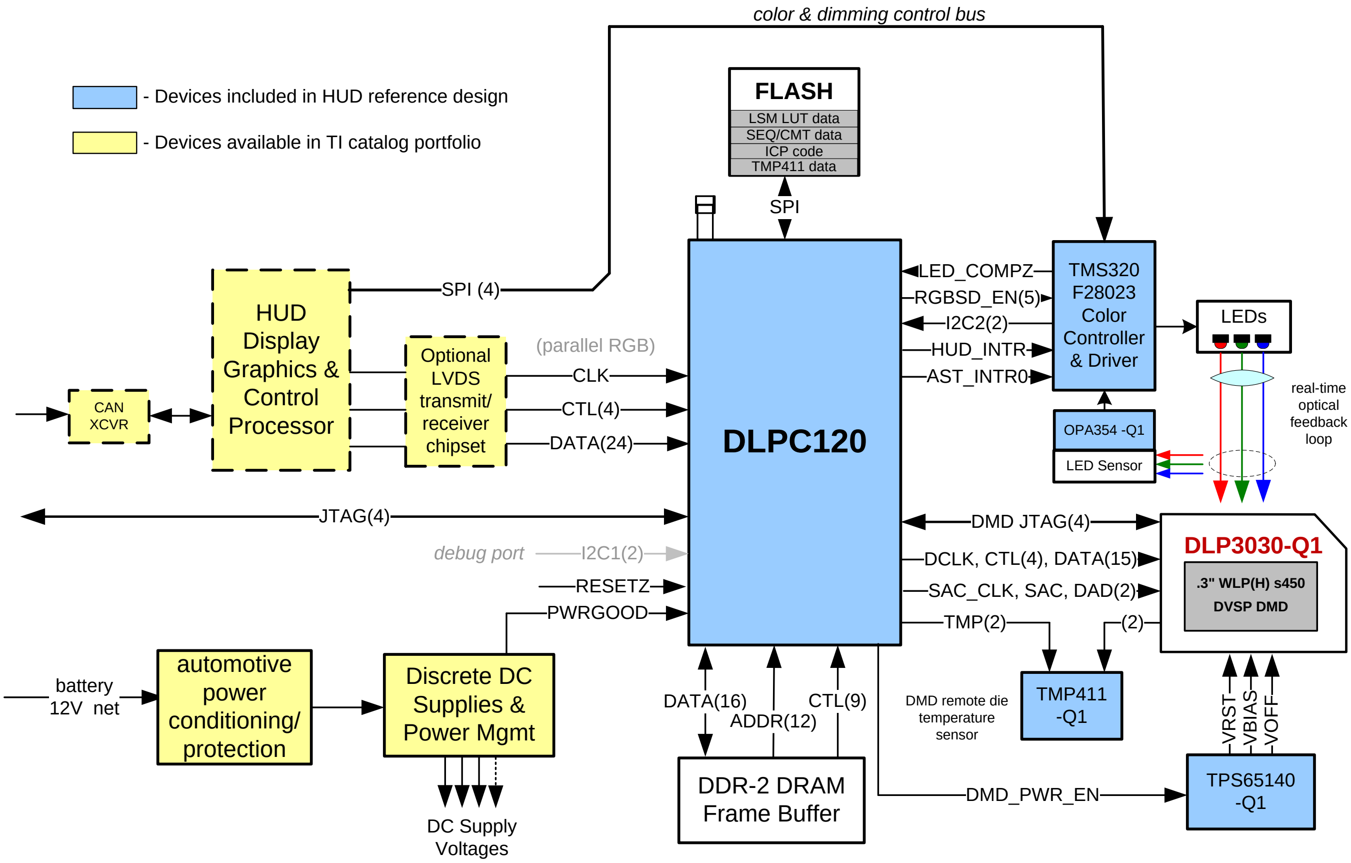

Figure 1. Piccolo LED Controller Block Diagram

The DLPC120 ASIC performs all image processing and display control to support DLP3000-Q1 or DLP3030-Q1 DMD. DLPC120 accepts parallel video input from the HUD Display Graphics and Control Processor which acts as the display systems master.

The Piccolo MCU receives brightness/dimming commands from the display controller via a SPI bus input. Piccolo SW interprets these commands and coordinates all adjustments within the sub-system. The Piccolo configures the DLPC120 over a dedicated I2C bus (I2C2 in Figure 1). The Piccolo SW also includes a bootloader, which enables updating the main application of the Piccolo MCU via SPI.

Purpose and Scope

The purpose of this document is to provide a command interface specification for the SPI interface of the Piccolo LED Controller.

Terms and Abbreviations

Abbreviations	Description
SPI	Serial Peripheral Interface
Slave	Piccolo TMS320F28023
Master	Display Host Controller acting as SPI Master

References

	Document Number	Document Name
1.	DLPA084	LED Driver for DLP3030-Q1 Displays Application Note
2.	DLPU061	Piccolo Software Programmers Guide for the DLPC120 ASIC
3.	DLPU055	I2C Programmer's Guide for DLPC120 ASIC

Chapter 1

DLPU057–March 2018

Serial Peripheral Interface (SPI)

1.1 Serial Peripheral Interface (SPI)

Piccolo LED Controller hosts a 4-wire SPI slave interface that serves as the communication port to the display controller. The SPI interface supports high-speed serial data transfers up to 400 kHz. Following four pins are used for SPI communication:

- SPISOMI:SPI slave-output/master-input pin
- SPISIMO:SPI slave-input/master-output pin
- SPISTEn:SPI slave transmit-enable pin
- SPICLK:SPI serial-clock pin

The SPISTEn pin operates as the slave-select pin. A high (inactive) signal causes the slave SPI serial shift register to stop and its serial output pin to be put into the high-impedance state. The SPISIMO input serves as the serial data input for the port (from the master to Piccolo); the SPISOMI output serves as the serial data output (from Piccolo to master).

The SPICLK input serves as the serial data clock for both the input and output data. Data is latched at the SPISIMO input on the rising edge of SPICLK, while data is clocked out of the SPISOMI output on the falling edge of SPICLK.

1.2 Block Diagram

The following diagram shows an outline of connection between Piccolo (SPI Slave) and Display Controller (SPI Master).

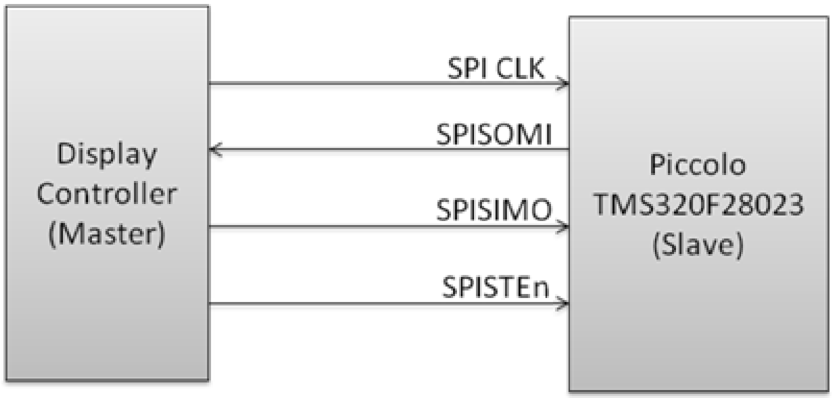

Figure 1-1. Block Diagram

1.3 Signal Format

The signal specifications are as given below:

Baud Rate	100 kHz (this could be higher but has not been tested completely with other rates)
Mode	Data written on falling edge, read on rising edge. The clock polarity is set to high ('1') and the clock phase is set to normal ('0' – no half cycle delay).
Data Length	8 Bit (1 byte)
Bit Order	MSB First (within a byte)
Byte Order	Least Significant Byte First (for multi byte data)
Byte Spacing	1 ms

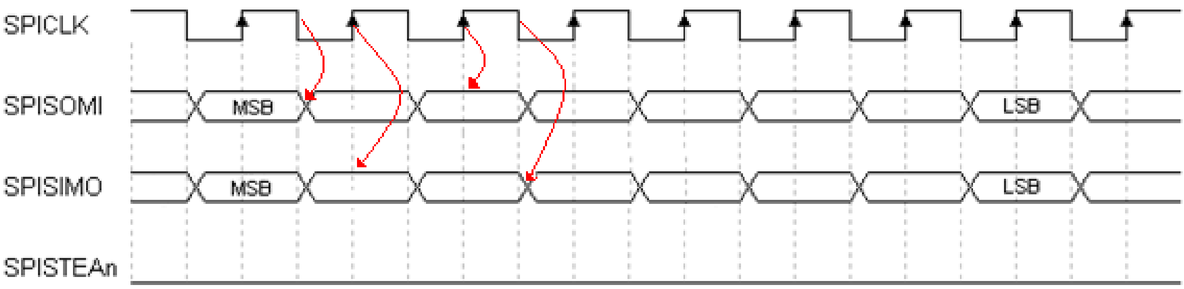

Figure 1-2. Signal Format

1.4 SPI Packet Structure

All the commands are sent as packets. Only the Master can initiate a packet transaction. A command packet consists of START_CHARACTER, 2 byte header (command byte, Length), variable data payload and a checksum byte. These are explained in the subsections below.

The SPI Packet structure sent by the Master is shown in Figure 1-3.

Byte 0	Byte 1	Byte 2	Byte 3	…	Byte n+2	Byte n+3
START CHARACTER	Command Byte	Length(n)	Data[1]	…	Data[n]	Checksum

Figure 1-3. SPI Packet Structure - Master

In Figure 1-3, *n* stands for number of bytes of Data for the command being transmitted. For example, a backlight command (see Section 3.2.1) will have the following structure:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Command Byte = 00	n = 2	Data[1] = FF	Data[2] = FF	Checksum = 00

Figure 1-4. Example of a Command Packet

The slave will keep sending dummy bytes (0xFF) back to the master for each byte of the command packet. Once the command packet is completely received, the slave prepares a response byte after validating the checksum and the length byte. To receive the response byte, the master should keep sending dummy bytes to the slave till it receives a non-0xFF byte from slave. The first non-0xFF byte from slave will be the response byte.

For a read command the read data follows the response byte and has the structure shown in Figure 1-6, where,

k – This is the time (in bytes) needed by the slave to respond to the master's request.

m – The number of data bytes returned by the slave. This is also called length and is transmitted as byte k+1.

The number of bytes from k transmitted by slave can vary from minimum of 1 byte (response byte only – in case of write commands, failed read commands) up to 258 (255 data bytes + Response + Length + Checksum) depending on the read command (unless master aborts the command).

Byte k-2	Byte k-1	Byte k	Byte k+1	Byte k+2	…	Byte k+m+1	Byte k+m+2
0xFF	0xFF	Response Byte	Length(m)	Data[1]	…	Data[m]	Checksum

Figure 1-5. SPI Packet Structure - Slave (Read)

As an example, the slave response to a backlight read command (see Section 3.2.3) is given below:

Byte k	Byte k+1	Byte k+2	Byte k+3	Byte k+4
Response Byte = 01	m = 2	Data[0] = FF	Data[1] = FF	Checksum = 01

Figure 1-6. Example of a SPI Read Response Packet from Slave

1.4.1 START CHARACTER

This byte indicates the start of a command packet send by the master. This byte resets the internal SPI state machine and any incomplete command will be aborted by the slave. The START CHARACTER is thrown away and not used in any computations. More details on START CHARACTER can be found in Section 1.5.

1.4.2 Command Byte

In the command byte, the least significant bit indicates whether it is a write or read command. For this bit, a 1 indicates a read operation, while a 0 indicates a write operation. The remaining seven bits of the command byte are the unique ID of the command targeted by the write or read operation.

Command Byte							
7	6	5	4	3	2	1	0
Command ID							R/Wn

Figure 1-7. SPI Command Byte

1.4.3 Length Byte

Length Byte represents the number of bytes of data following the length byte. The length does not include checksum, command byte and the length byte.

Length Byte							
7	6	5	4	3	2	1	0
Length (number of bytes of data)							

Figure 1-8. SPI Length Byte

1.4.4 Data Bytes

The data related to the current command. The data length and content can vary depending on the command.

1.4.5 Checksum

This byte is used for verifying the integrity of the command packet. It is the sum of all the bytes in the current packet (from Command Byte to Data[n]).

For the command packet send by master,

Checksum = (Byte$_1$ + Byte$_2$ + … + Byte$_n$ + Byte$_{n+1}$ + Byte$_{n+2}$) MOD 0x100

Where n = length of data (see Figure 1-3).

Please note that START_CHARACTER is not included in the checksum.

The MOD function is defined as,

R = a mod b

Where,

a = dividend

b = divisor

R = Remainder on division of a by b.

For the response packet send by slave for a read command (Figure 1-6), the checksum is computed as the sum of all bytes from Response Byte to Data[m]. This is shown below:

Checksum = (Byte$_k$ + Byte$_{k+1}$ + … + Byte$_{k+m+1}$) MOD 0x100

Where m = length of data returned by slave

k = byte index of response byte.

1.4.6 Response Byte

Response byte is used as an acknowledgment to the display controller indicating whether the command was received successfully. The following are the return codes sent as response by the slave.

0xFF – Dummy (Should not be treated as a response byte)

0x00 – (Reserved)

0x01 – Success

0x02 – Checksum Error

0x03 – Invalid Command (indicates invalid command ID)

0x04 – Command Not Available (Command ID is valid, but not allowed in this mode)

0x05 – Length Mismatch (Expected Length for this command is different)

0x06 – Reserved

0x07 – Write Command Execution Failed

0x08 – Read Command Execution Failed

0x09-0xFE – (Reserved)

Slave will send Dummy byte (0xFF) as response until the command completes execution or it detects an error. The status is returned in the response byte as well as flagged in the system status. All reserved bytes above indicate an error in SPI protocol.

1.5 Error Checking Mechanisms

The checksum byte (Section 1.4.5) in the SPI protocol helps in identifying any errors caused during transmission and execution. It immediately informs the master of any errors using the response byte (Section 1.4.6). However, just by looking at a byte transmitted over SPI bus there is no way the slave can identify what type (Command Byte or Length or Data or Checksum) of byte it is. To solve this problem, we need a method to identify the start of a transaction.

1.5.1 SPI Transaction Start and Escape Character

To aid the Piccolo in finding the start of a transaction, a special START CHARACTER should be sent before the start of Command Byte (Byte 1) in the Command Packet. The hex character 0xA5 is treated as this special START CHARACTER. The START CHARACTER 0xA5 should not be present anywhere else in the byte stream sent by master.

If the master needs to send 0xA5 in the byte stream (not as a START CHARACTER), it should use the help of another special character 0x5A named as the ESCAPE CHARACTER. Sending 0x5A 0x00 consecutively will be treated as a single data byte 0xA5.

The following table illustrates how various combinations of bytes are interpreted.

Table 1-1. START and ESCAPE Character Examples

Bytes	Interpretation
0xA5	Start of transaction. Next Byte is Command Byte.
0x5A 0x00	Data Byte 0xA5
0x5A 0x5A	Data Byte 0x5A
0x5A	Data Byte 0xYY, when YY is the hex byte immediately following 0x5A and 0xYY is non-zero.

The ESCAPE CHARACTER (0x5A) can come in as a replacement for any byte in the command packet explained in Figure 1-3.

The START CHARACTER can be used to abort any existing command. Whenever 0xA5 is received by the slave, it clears any unfinished command and starts afresh treating the next byte as Command byte.

The following figures illustrate the complete structure of a write and read commands.

Write	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	...	Byte n+2	Byte n+3	Byte n+4	Byte n+5	Byte n+6
Host (Master)	**0xA5**	CMD	LEN	D1	D2	...	Dn	CKSUM	0	0	0
Piccolo (Slave)	FF	FF	FF	FF	FF	...	FF	FF	FF	FF	Response

Figure 1-9. SPI Write Transaction

Read	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	...	Byte n+2	Byte n+3	Byte n+4	Byte n+5
Host (Master)	**0xA5**	CMD	LEN	D1	D2	...	Dn	CKSUM	0	0
Piccolo (Slave)	FF	FF	FF	FF	FF	...	FF	FF	FF	FF

Read	Byte n+6	Byte n+7	Byte n+8	Byte n+9	...	Byte n+m+7	Byte n+m+8
Host (Master)	0	0	0	0	...	0	0
Piccolo (Slave)	Response	LEN	D1	D2	...	Dm	CKSUM

Figure 1-10. SPI Read Transaction

Please note that the start and escape characters are not applicable to data sent by slave.

1.5.2 Checksum Computation With Special Characters

The checksum and data length computed will not take into account the presence of START CHARACTER and any ESCAPE CHARACTERs present in the byte stream. START CHARACTERs are not included when the checksum is computed. ESCAPE CHARACTERs are replaced with interpreted data for checksum computation.

For example, if master needs to send backlight value 0xA55A (42330) (see Section 3.2.1 for description of Backlight command), the original data without start and escape characters look like:

CMD	LEN	D0	D1	CKSUM
0x00	0x02	0xA5	0x5A	0x01

Since the byte stream contains 0xA5 and 0x5A they need to be properly masked. Besides, the START CHARACTER should also be send before the actual CMD Byte. Note that the checksum did not change (the START and ESCAPE characters were not added to the checksum).

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
START	**CMD**	**LEN**	**D0**	**D0**	**D1**	**D1**	**CKSUM**
0xA5	0x00	0x02	0x5A	0X00	0x5A	0x5A	0x01

On receiving the above byte stream, Piccolo treats Byte 3 and Byte 4 together as a single byte which has value 0xA5 and Byte 5 and Byte 6 together as a single byte which has value 0x5A.

More examples can be found in Section 4.2.

1.5.3 SPI Flow Control Diagrams

After forming the SPI command packet the master should add the START and any ESCAPE characters required before transmission. The control flow for this postprocessing stage is given in Figure 1-11.

Similarly a preprocessing stage should run in slave to decode the transmitted bytes to remove any START characters and transform ESCAPE character sequences to respective data bytes. The respective control flow in slave is given in Figure 1-12.

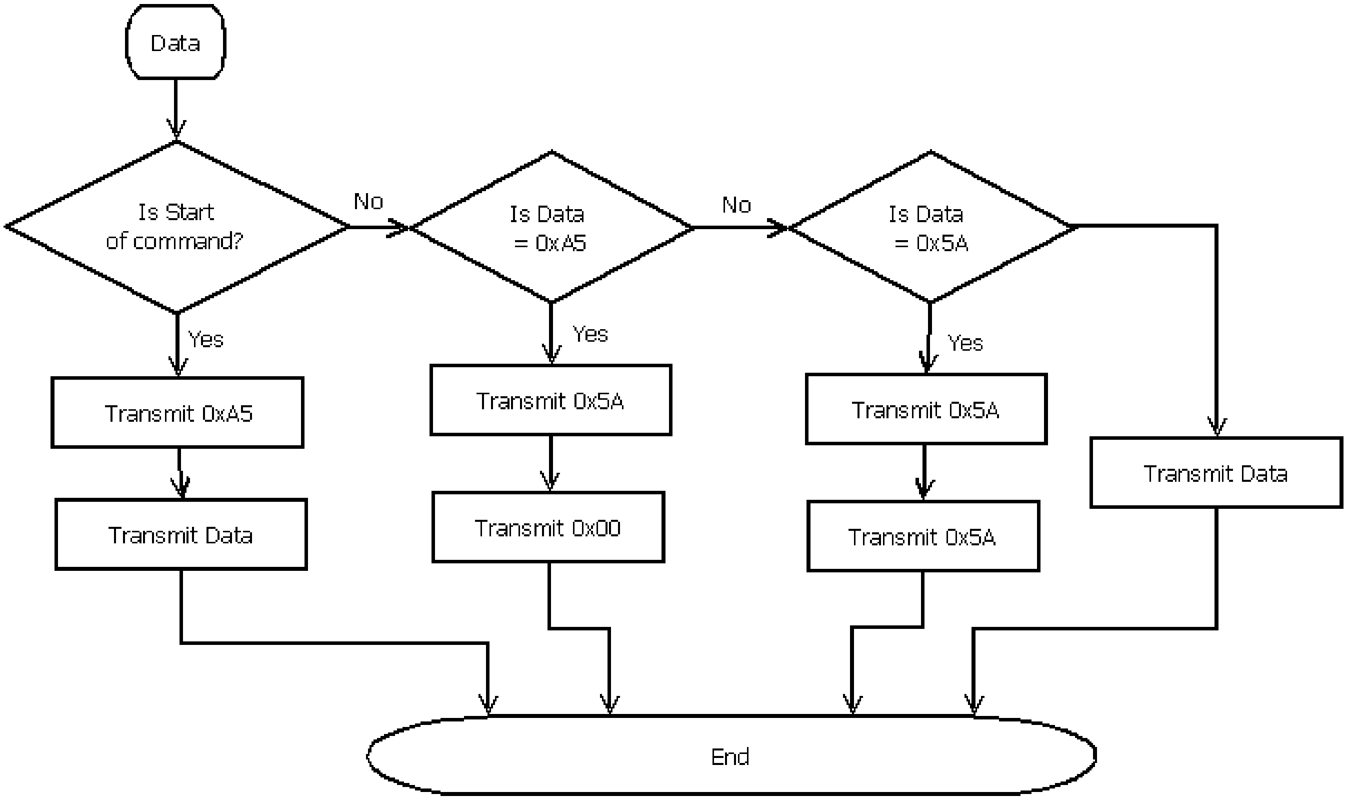

Figure 1-11. SPI Master Postprocessing Stage

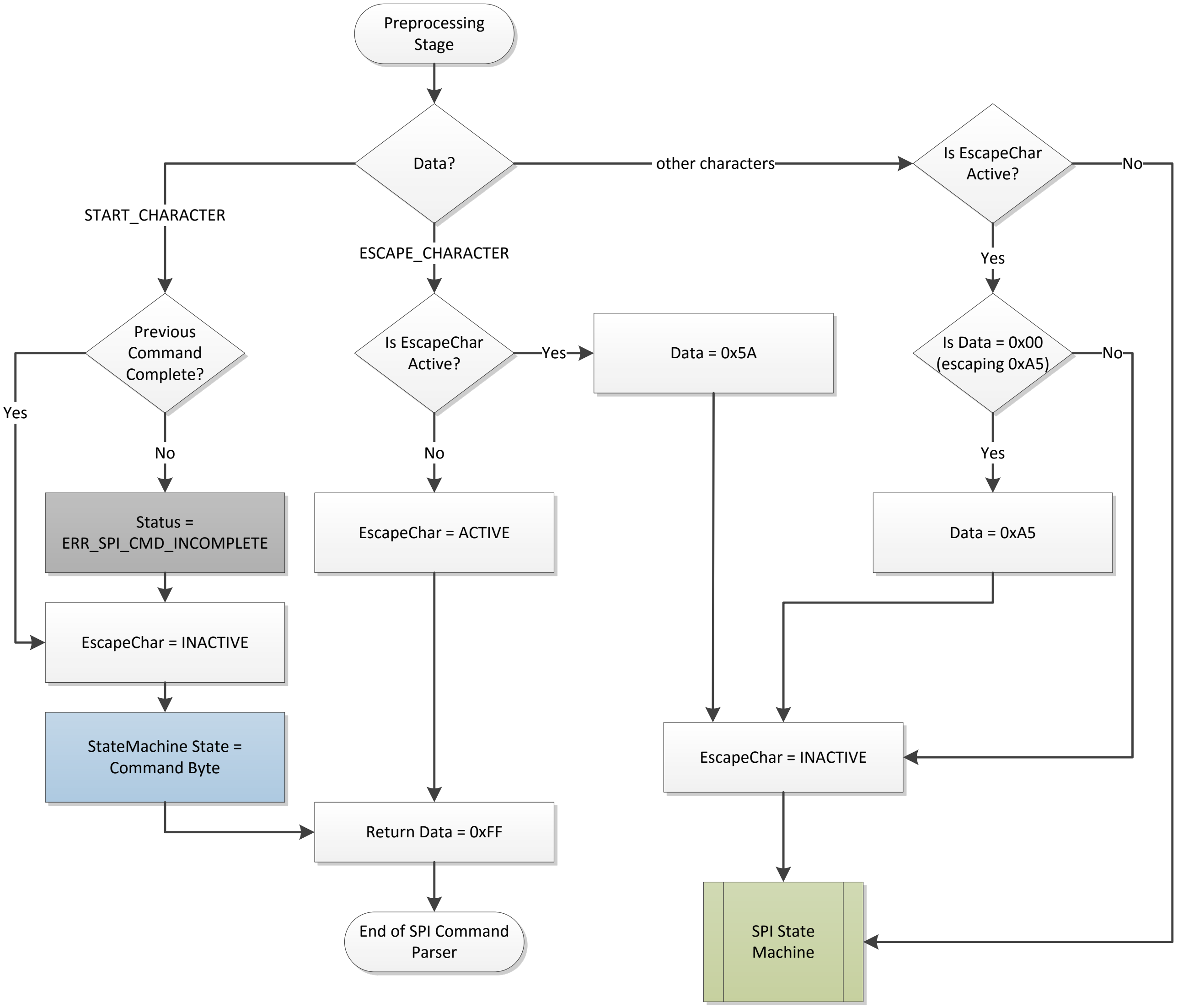

Figure 1-12. SPI Slave Preprocessing Stage

SPI Command Specification – Bootloader

2.1 SPI Command Specification – Bootloader

The following sections define the list of commands and their description, as supported by the Piccolo Bootloader SPI interface.

Note: The command specification is subject to correction or change as required.

Table 2-1. List of Bootloader SPI Commands

Write/Read	Commands	7-bit Address (Command ID)	Write Permissions	Section
	Reserved	*00h - 31h*		
Read	Bootloader SW Version	32h	Read-Only	3.1
Read	Bootloader SW Status	33h	Read-Only	3.2
	Reserved	*34h-70h*		
Write/Read	Piccolo Flash Read	71h	Always	3.3
	Reserved	*72h – 79h*		
Read	Toggle Mode	7Ah	Always	3.4
Write/Read	Piccolo Program Software	7Bh	Always	0
	Reserved	*7Ch*		
	<Reserved - Should not be used>	*7Dh*		
Read	Piccolo Program Mode	7Eh	Read-Only	3.6
	<Reserved - Should not be used>	*7Fh*		

2.2 Bootloader Software Version (32h)

This command requests a read of the Bootloader Software Version.

Byte	Description
1	Command Byte (65h)
2	Length (00h)
3	Checksum (65h)

2.2.1 Read Response

This is the data response for a valid request to read the Bootloader Software Version.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3	Major
4	Minor
5	Build (lsb)
6	Build (msb)
7	Checksum (Sum of bytes 1 to 6)

Version Data is of the form *Major.Minor (Build).*

2.3 Bootloader Software Status (33h)

This command requests a read of the Bootloader Software Status.

Byte	Description
1	Command Byte (67h)
2	Length (00h)
3	Checksum (67h)

Note: The Piccolo SW status command can be used to get more information when a command fails with response byte indicating Write/Read Execution Failure. The command can also be sent to get information about the current Piccolo SW state.

2.3.1 Read Response

This is the data response for a valid request to read the Bootloader Software Status. Status is automatically cleared upon read.

Note: The status bits are subject to change during development.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3 - 6	Status – See below
7	Checksum (Sum of bytes 1 to 6)

msb	**Byte 3**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) -	SPI Overrun Detect	b(3) -	SPI Incomplete Command
b(6) -	*Reserved*	b(2) -	SPI Command Not Available
b(5) -	*Reserved*	b(1) -	SPI Invalid Data
b(4) -	SPI Read Data Invalid	b(0) -	SPI Invalid Command

msb	**Byte 4**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) -	*Reserved*	b(3) -	*Reserved*
b(6) -	*Reserved*	b(2) -	*Reserved*
b(5) -	Data out of Range	b(1) -	*Reserved*
b(4) -	*Reserved*	b(0) -	*Reserved*

msb	**Byte 5**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) -	*Reserved*	b(3) -	Flash sector erase failed
b(6) -	Timer Error	b(2) -	*Reserved*
b(5) -	Unhandled Interrupt Received	b(1) -	*Reserved*
b(4) -	Flash Programming failed	b(0) -	*Reserved*

msb	**Byte 6**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) -	SPI – Escape character detected	b(3) -	*Reserved*
b(6) -	SPI – Length Mismatch/Other errors	b(2) -	*Reserved*
b(5) -	SPI – Ignored some bytes	b(1) -	*Reserved*
b(4) -	SPI – Checksum Mismatch	b(0) -	*Reserved*

2.4 Binary Flash Read/Set Binary Read Start Address (71h)

The Binary flash read function is used to read the piccolo flash data in binary format. This is useful to read out the main application data. The function is versatile and allows the reading of any part of the flash from any address. This is done by setting the start address by using the Set Binary Read start address function and then, using binary flash read with the required number of words as argument.

2.4.1 Write

This command is used to set flash binary read start address. The next binary read performed will be from the start address.

Byte	**Description**
1	Command Byte (E2h)
2	Length (04h)
3	Read Start Address (lsb)
4	Read Start Address
5	Read Start Address
6	Read Start Address (msb)
7	Checksum (Sum of bytes 1 to 6)

2.4.2 Read

This command requests a binary read of the Piccolo Flash. The number of words to be read is given as input.

Byte	**Description**
1	Command Byte (E3h)
2	Length (01h)
3	Number of words (1 - 127)
4	Checksum (Sum of bytes 1 to 3)

2.4.3 Read Response

This is the data response for a valid read command.

Byte	**Description**
1	Response Byte (01h)
2	Length (255h)
3 to 257	Binary Flash Data
258	Checksum (Sum of bytes 1 to 257)

The values of all bytes after the required number of words are set to 0.

2.5 Toggle Mode (7Ah)

2.5.1 Read

This command verifies the main application and jumps to the main application if the verification is successful. The jump occurs after the command execution and acknowledgment is complete.

Byte	Description
1	Command Byte (F5h)
2	Length (05h)
3	Toggle to application(01h)
4-7	Toggle Mode Write Signature (FF00FF00h)
8	Checksum (Sum of bytes 1 to 7)

2.5.2 Read Response

This is the data response for a valid request to read the toggle mode read signature.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3-6	Toggle Mode Bootloader Read Signature (43218765h)
7	Checksum (Sum of bytes 1 to 6)

2.6 Program Piccolo Software (7Bh)

This command is used to program Piccolo's main application. It has four functions: erase, start address and length, program, and checksum validation. A secondary op-code is used to decide which of the four actions to take.

Steps to program:

1. Erase – Erase the sectors to be programmed.
2. Start Address and Length – Set the address from which programming is to be started and length of the data to be programmed.
3. Program – Send data to be programmed (maximum 127 words). The number of bytes to be programmed should be even.
4. To program the next region, go to step 2.
5. Validate Data – Validate application.

Byte 3 is secondary op-code. It can have following values:

Secondary op-code	Action	Write/Read
00h	Erase	Write
01h	Start Address and Length	Write
02h	Program	Write
03h	Validate Data	Read

Figure 2-1. Program Piccolo Secondary Op-code

2.6.1 Erase

This command is used to erase flash sectors. It doesn't allow erasing sector A as it has bootloader code.

Byte	Description
1	Command Byte (F6h)
2	Length (02h)
3	Secondary Op-code – Erase (00h)
4	Sector Mask
5	Checksum (Sum of bytes 1 to 4)

msb	**Byte 4**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b(7) -	*Sector H*	b(3) -	*Sector D*
b(6) -	*Sector G*	b(2) -	*Sector C*
b(5) -	*Sector F*	b(1) -	*Sector B*
b(4) -	*Sector E*	b(0) -	*Reserved*

Figure 2-2. Program Piccolo Erase Sector Mask

2.6.2 Start Address and Length

This command sets the start address and length for the data to be programmed.

For every contiguous region, set the start address and length of the region. The length is the number of 16-bit words to be programmed in that region.

Byte	Description
1	Command Byte (F6h)
2	Length (09h)
3	Secondary Op-code - Start Address and Length (01h)
4	Start Address (lsb)
5	Start Address
6	Start Address
7	Start Address (msb)
8	Region Length (lsb)
9	Region Length
10	Region Length
11	Region Length (msb)
12	Checksum (Sum of bytes 1 to 11)

2.6.3 Program

This command programs the binary data starting from the location next to the last programmed location. The length of binary data should be multiple of 2 ranging from 2 bytes to 254 bytes. Also, it checks if the command is trying to program beyond the region specified by the *Start Address and Length* settings before programming the data.

Byte	Description
1	Command Byte (F6h)
2	Length (3 to 255)
3	Secondary Op-code – Program (02h)
4 to 257	Binary Flash Data
258	Checksum (Sum of bytes 1 to 257)

Note: Length (byte 2) is number of write bytes. Its value should be number of bytes of binary flash data + 1.

2.6.4 Read

This command requests status of application signature and checksum validation for the programmed application. It returns ‘01h’ if application is correct and the checksum is validated, else it returns ‘00h’

Byte	Description
1	Command Byte (F7h)
2	Length (01h)
3	Secondary Op-code – Validate Data (03h)
4	Checksum (Sum of bytes 1 to 3)

2.6.5 Read Response

This is the data response for a valid read command.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	Checksum verification status (00h or 01h)
4	Checksum (Sum of bytes 1 to 3)

2.7 Piccolo Program Mode (7Eh)

This command requests a read of the Piccolo Program Mode.

Byte	Description
1	Command Byte (FDh)
2	Length (00h)
3	Checksum (FDh)

2.7.1 Read Response

This is the data response for a valid request to read the Piccolo Program Mode.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	Program Mode(01h)
4	Checksum (Sum of bytes 1 to 3)

msb	**Byte 3**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

B (7:1)	Reserved
b (0) -	Piccolo Program Mode 0 – Main Application 1 – Bootloader

Figure 2-3. Program Mode Byte Format

2.8 Stay in Bootloader

Purpose of this command is to indicate to the bootloader that the control should stay in bootloader and that jump to main application should not be performed automatically. Bootloader waits for some time (10ms) after initialization, to check if this command is received.

This command is to be sent without the command packet. That means, there is no Start Character, Command Byte or Checksum bytes sent over SPI. The command is a repeated 4-byte pattern. The bytes are to be sent until a valid 4-byte response pattern is received back from piccolo. When the host receives the 4-byte response pattern, the host will stop sending the 4-byte pattern.

The bootloader will monitor for this 4-byte pattern. When it detects the pattern, it sends back a 4-byte response pattern with the next 4 bytes. Detection of this pattern indicates to the bootloader that it should not jump to the main application and wait for further SPI commands.

2.8.1 Write Bytes

Byte	Description
(4i+1)- (4i+4)	Stay in Bootloader Write Signature (18273645h)

Note: ‘i’ stands for the i^{th} iteration of the pattern. There is no associated command byte, length or checksum.

2.8.2 Read Response

Byte	Description
1-4	Stay in Bootloader Read Signature (AA55AA55h)

Note: The response bytes are sent as a response to the successful detection of write pattern. There is no associated length, data or checksum.

Chapter 3

DLPU057–March 2018

SPI Command Specification – Main Application

3.1 SPI Command Specification – Main Application

The following sections define the list of commands and their description as supported by the Piccolo Main Application SPI interface.

The following code is used to describe permissions:

Calibration Mode Permissions

CN: Calibration and Normal Mode

CO: Calibration only

NO: Normal Only

ASIC-In-Reset state Permissions

RA: Reset and Active states

RO: Reset only

AO: Active Only

Master On/Off Permissions

OO: Master on and off

ON: On only

OF: OFF Only

NA: Not Applicable

Some commands are included for test purposes but can be removed in the final application. These commands are marked with an **X** in the development only column.

Note: The command specification is subject to correction or change as required.

Table 3-1. List of Main Application Commands

Write/Read	Commands	7-bit Address (Command ID)	Write Permissions	Read Permissions	Section	Development Only
Write/Read	Backlight	00h	NO, RA, ON	CN, RA, ON	Section 3.2	
Write/Read	Master On/Off	01h	CN, AO, OO	CN, RA, OO	Section 3.3	
Write/Read	DMD Park	02h	CN, AO, OO	CN, RA, OO	Section 3.4	
	Reserved	*03h - 24h*				
Write/Read	Splash Control Mode	25h	CN, AO, ON	CN, AO, ON	Section 3.5	
Write/Read	DMD Drive Strength	26h	CN, AO, ON	CN, AO, ON	Section 3.6	X
Write/Read	Heater PWM Parameters	27h	CN, AO, ON	CN, AO, ON	Section 3.7	X
Write/Read	Bezel Offset	28h	CN, AO, ON	CN, AO, ON	Section 3.8	
	Reserved	*29h – 2Fh*				

Table 3-1. List of Main Application Commands (continued)

Write/Read	Switch SPI Bus	2Fh	CN, RA, OO	CN, RA, OO	Section 3.9	
Read	ASIC BIST Results	30h	NA	CN, RA, ON	Section 3.10	
Read	ASIC Initialization Type	31h	NA	CN, RA, ON	Section 3.11	X
Read	Piccolo SW Version	32h	NA	CN, RA, OO	Section 3.12	
Read	Piccolo SW Status	33h	NA	CN, RA, OO	Section 3.13	
Write/Read	ASIC Register	34h	CN, AO, ON	CN, AO, ON	Section 3.14	
Write/Read	VAC Mode	35h	CN, RA, ON	CN, RA, ON	Section 3.15	X
Read	Operating Mode	36h	NA	CN, AO, ON	Section 3.16	
Read	PWM Sensitivity	37h	NA	CN, RA, ON	Section 3.17	X
Read	Piccolo SW Secondary Status	38h	NA	CN, RA, OO	Section 3.18	
Read	Extra Information Keys	39h	NA	CN, RA, ON	Section 3.19	
Read	Extra Information Values	3Ah	NA	CN, RA, ON	Section 3.20	
	Reserved	*3Bh-3Fh*				
Write/Read	Dimming LUT Group and Gamma Index	40h	CO, AO, ON	CN, RA, ON	Section 3.21	
Read	Dimming LUT Group Information	41h	NA	CN, RA, ON	Section 3.22	
Read	CMT/Gamma Information	43h	NA	CN, RA, ON	Section 3.23	
	Reserved	*44h-4Dh*				
Read	Command List Address	4Eh	NA	CN, RA, ON	Section 3.24	
Read	Generic Command List Type	4Fh	NA	CN, RA, ON	Section 3.25	
Read	Command List Numbers	50h	NA	CN, RA, ON	Section 3.26	
Write/Read	Command List Info/ Execute Command List	51h	CN, AO, ON	CN, RA, ON	Section 3.27	
Write/Read	Front End Video BIST Pixels	53h	CN, RA, ON	CN, RA, ON	Section 3.28	
Write/Read	Front End Video BIST	54h	CN, AO, ON	CN, RA, ON	Section 3.29	
Write/Read	External Video Detect BIST	55h	CN, AO, ON	CN, RA, ON	Section 3.30	
	Reserved	*56h-5Fh*				
Write/Read	Low Pass Temperature Filter Constants	60h	CN, RA, ON	CN, RA, ON	Section 3.31	X
Write/Read	Temperature Compensation	61h	CN, RA, ON	CN, RA, ON	Section 3.32	X
Read	LED Voltage and Current	62h	NA	CN, RA, ON	Section 3.33	
Read	DMD Temperature	63h	NA	CN, AO, ON	Section 3.34	

Table 3-1. List of Main Application Commands (continued)

Write/Read	Enable Calibration Mode	64h	CN, RA, ON	Always	Section 3.35	
Write/Read	Red LED PWM	65h	CO, RA, ON	CN, RA, ON	Section 3.36	
Write/Read	Green LED PWM	66h	CO, RA, ON	CN, RA, ON	Section 3.37	
Write/Read	Blue LED PWM	67h	CO, RA, ON	CN, RA, ON	Section 3.38	
Write/Read	Current Limit PWM	68h	CO, RA, ON	CN, RA, ON	Section 3.39	
Write/Read	Sensor Gain	69h	CO, RA, ON	CN, RA, ON	Section 3.40	
Write/Read	Command Table Index	6Ah	CO, RA, ON	CN, RA, ON	Section 3.41	
Read	Sensor Gain Map	6Bh	NA	CN, RA, ON	Section 3.42	
Read	Adapter Voltages	6Ch	NA	CN, RA, ON	Section 3.43	
Read	Configuration Format Version	6Dh	NA	CN, RA, OO	Section 3.44	
Read	Calibration Format Version	6Eh	NA	CN, RA, OO	Section 3.45	
Read	Calibration Data Version	6Fh	NA	CN, RA, OO	Section 3.46	
Write	Program Calibration Data to Flash	70h	CO, RA, ON	CN, RA, ON	Section 3.47	
Write/Read	Piccolo Flash Read	71h	CN, RA, ON	CN, RA, ON	Section 3.48	
Write/Read	PWM Period/Read Info	72h	CO, RA, ON	CN, RA, ON	Section 3.49	
Read	PWM Scale Factor	73h	NA	CN, RA, ON	Section 3.50	X
Read	Read ASIC Flash	74h	NA	CN, AO, ON	Section 3.51	
Read	ASIC Flash Info	75h	NA	CN, RA, ON	Section 3.52	
	Reserved	*76h – 77h*				
Read	Power Rail Voltages	78h	NA	CN, RA, ON	Section 3.53	
Write/Read	Voltage Supervision	79h	CN, RA, ON	CN, RA, ON	Section 3.54	X
Read	Toggle Mode	7Ah	CN, RA, OO	CN, RA, OO	Section 3.55	
	Reserved	*7Bh*				
Write/Read	Piccolo IIC Clock Rate	7Ch	CN, AO, ON	CN, RA, ON	Section 3.56	X
	Reserved	*7Dh*				
Read	Piccolo Program Mode	7Eh	CN, RA, OO	CN, RA, OO	Section 3.57.1	
	Reserved	*7Fh*				

Note 1: Details of Normal Mode and Cal Mode are explained in section Section 3.35.

3.2 Backlight (00h)

3.2.1 Write

This command is used to set the dimming level which is expressed as a percentage of full scale of factory calibrated maximum brightness.

Byte	Description
1	Command Byte (00h)
2	Length (02h)
3	Backlight Level (lsb)
4	Backlight Level (msb)
5	Checksum (Sum of bytes 1 to 4)

The figure below shows the bit order and weighing for the Backlight Level relative to the full scale.

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}

Figure 3-1. Bit Weight Definition for Backlight Level

The Backlight Level can range from almost 1 (when all bits are set to 1) to $1/2^{16}$. The maximum brightness will be multiplied by this value to set the backlight. In other words, a value of 65535 (all bits set to 1) indicates the brightest value of the system.

The range (65535 to 0) is same as (1 to 0) when the binary representation is treated as unsigned 0.16 fixed point number. As such, the software uses the (65535 to 0) as the backlight range for all purposes.

A backlight value of 35000 should be sent as:

Byte 3 = Backlight (lsb) = lsb 8 bits (35000) = 0xB8

Byte 4 = Backlight (msb) = msb 8 bits (35000) = 0x88

3.2.2 Read

This command requests a read of the Backlight value set by the system.

Byte	Description
1	Command Byte (01h)
2	Length (00h)
3	Checksum (01h)

3.2.3 Read Response

This is the data response for a valid request to read the Backlight.

Byte	Description
1	Response Byte (01h)
2	Length (02h)
3	Backlight Level (lsb)
4	Backlight Level (msb)
5	Checksum (Sum of bytes 1 to 4)

The Backlight value returned is of the same format as given in Figure 3-1.

3.3 Master ON/OFF (01h)

3.3.1 Write

This command is used to turn on/off the system software functions. All SPI commands except Master ON/OFF (01h) are disabled when turned OFF.

Byte	Description
1	Command Byte (02h)
2	Length (01h)
3	Master On/Off
4	Checksum (Sum of bytes 1 to 3)

For Master On ‘1’ is to be sent and for Master Off ‘0’ is to be sent as the 3rd byte.

3.3.2 Read

This command requests a read of the state of the system.

Byte	Description
1	Command Byte (03h)
2	Length (00h)
3	Checksum (03h)

3.3.3 Read Response

This is the data response for a valid request to read the system state.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	System State (ON/OFF)
4	Checksum (Sum of bytes 1 to 4)

System State of 1 Indicates Master ON and 0 indicates Master OFF.

3.4 DMD Park (02h)

3.4.1 Write

This command can be used to park or un-park the DMD. If the Park byte = 1, the DMD is parked. If the Park byte = 0, the DMD is un-parked.

Byte	Description
1	Command Byte (04h)
2	Length (01h)
3	Park
4	Checksum (Sum of bytes 1 to 3)

3.4.2 Read

This command returns whether the DMD is parked or unparked.

Byte	Description
1	Command Byte (05h)
2	Length (00h)
3	Checksum (05h)

3.4.3 Read Response

This response indicates whether the DMD is parked or un-parked.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	Park Status (See below)
4	Checksum (Sum of bytes 1 to 3)

Park Status	Description
0	DMD un-parked
2	DMD parked by SPI Park Command
4	DMD parked because the system is in bootloader mode
8	DMD parked by Master On/Off command

3.5 Splash Control Mode (25h)

3.5.1 Write

This command should be enabled when splash is displayed on the screen. The Piccolo SW uses a slightly different dimming strategy when splash is displayed.

Byte	Description
1	Command Byte (4Ah)
2	Length (01h)
3	Splash Control Mode (See below)
4	Checksum (Sum of bytes 1 to 3)

msb	Byte 3						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7:0) - Splash Control Mode
0 – Disable
1 – Enable

Figure 3-2. Splash Control Mode Byte Format

Note 1: Refer to **Piccolo Software Programmer's Guide for the DLPC120 ASIC (DLPU061)** for more details about Splash Control Mode.

3.5.2 Read

This command requests a read of the Splash Control Mode.

Byte	Description
1	Command Byte (4Bh)
2	Length (00h)
3	Checksum (4Bh)

3.5.3 Read Response

This is the data response for a valid request to read the Splash Control Mode.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	Splash Control Mode (See Figure 3-2)
4	Checksum (Sum of bytes 1 to 3)

3.6 DMD Drive Strength (26h)

3.6.1 Write

This command is used to set the DMD Drive Strength Index. Index values of 0, 1, 2 correspond to a drive strength of 6mA, 10mA, 12mA respectively.

Byte	Description
1	Command Byte (4Ch)
2	Length (01h)
3	DMD Drive Strength Index
4	Checksum (Sum of bytes 1 to 3)

3.6.2 Read

This command requests a read of the DMD Drive Strength Index. The values of the index correspond to the same drive strengths as mentioned in the write command.

Byte	Description
1	Command Byte (4Dh)
2	Length (00h)
3	Checksum (03h)

3.6.3 Read Response

This is the data response for a valid request to read the DMD Drive Strength Index.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	DMD Drive Strength Index
4	Checksum (Sum of bytes 1 to 3)

Note: For more details, refer to the **DLPC120-Q1 Programmer's Guide (DLPU055).**

3.7 Heater PWM Parameters (27h)

NOTE: The heater is not needed for DLP3030-Q1 chipset operation. A DLP3030-Q1 based PGU offers full performance over the entire operating temperature range without the use of the heater. Duty cycle should be left at 0.

3.7.1 Write

This command is used to set the Heater PWM Frequency (in kHz) and Duty Cycle (%).

Byte	Description
1	Command Byte (4Eh)
2	Length (03h)
3	Frequency in kHz (LSB)
5	Frequency (MSB)

Byte	Description
6	Duty Cycle (%)
7	Checksum (Sum of bytes 1 to 6)

3.7.2 Read

This command requests a read of the Heater PWM Parameters.

Byte	Description
1	Command Byte (4Fh)
2	Length (00h)
3	Checksum (03h)

3.7.3 Read Response

This is the data response for a valid request to read the Heater PWM Parameters.

Byte	Description
1	Response Byte (01h)
2	Length (03h)
3	Frequency in kHz (LSB)
4	Frequency (MSB)
5	Duty Cycle (%)
6	Checksum (Sum of bytes 1 to 5)

Note: For more details, refer to the **DLPC120-Q1 Programmer's Guide (DLPU055).**

3.8 Bezel Offset (28h)

3.8.1 Write

This command is used to set the horizontal and vertical bezel offset in pixels. The magnitude and sign are written separately. For the sign byte, ‘0’ corresponds to negative and ‘1’ corresponds to positive.

Byte	Description
1	Command Byte (50h)
2	Length (06h)
3	Magnitude of horizontal offset (LSB)
5	Magnitude of horizontal offset (MSB)
6	Sign of Horizontal offset
7	Magnitude of vertical offset (LSB)
8	Magnitude of vertical offset (MSB)
9	Sign of vertical offset
10	Checksum (Sum of bytes 1 to 9)

3.8.2 Read

This command requests a read of the bezel offset values.

Byte	Description
1	Command Byte (51h)
2	Length (00h)
3	Checksum (03h)

3.8.3 Read Response

This is the data response for a valid request to read the bezel offset values.

Byte	Description
1	Response Byte (01h)
2	Length (06h)
3	Magnitude of horizontal offset (LSB)
5	Magnitude of horizontal offset (MSB)
6	Sign of Horizontal offset
7	Magnitude of vertical offset (LSB)
8	Magnitude of vertical offset (MSB)
9	Sign of vertical offset
10	Checksum (Sum of bytes 1 to 9)

Note: For more details, refer to the **DLPC120-Q1 Programmer's Guide (DLPU055).**

3.9 Prepare ASIC Flash/Switch SPI Bus (2Fh)

3.9.1 Write

If ‘Enable’ is ‘1’, the command switches the SPI bus from Piccolo SPI bus, to DLPC120 App Flash SPI bus. It is used to program DLPC120 flash. It parks the DMD, resets DLPC120. The SPI switch occurs after the command execution and acknowledgment is complete.

If ‘Enable’ is ‘0’, the command brings back DLPC120 out of reset, initializes it and un-parks the DMD. This can be used if SPI switch was not successful.

Note 1: This command assumes that the electronics reference design has an electronic switch signal connected to the piccolo MCU.

Note 2: If SPI bus is successfully switched to DLPC120 App Flash, SPI communication with Piccolo will no longer be possible. To communicate to Piccolo over SPI, system needs to be reset.

Byte	Description
1	Command Byte (5Eh)
2	Length (01h)
3	Enable(00h or 01h)
4	Checksum (Sum of bytes 1 to 3)

3.9.2 Read

This command checks if the Piccolo software supports SPI switching.

Byte	Description
1	Command Byte (5Fh)
2	Length (00h)
3	Checksum (5Fh)

3.9.3 Read Response

This is the data response for a valid request to check if the Piccolo software supports SPI switching.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3-6	SPI Switch Signature (11001100h)
7	Checksum (Sum of bytes 1 to 6)

3.10 ASIC BIST Results (30h)

This command requests a read of the results of all the ASIC BIST's performed by the piccolo at system startup and the associated checksums/ Device ID's.

Byte	Description
1	Command Byte (61h)
2	Length (00h)
3	Checksum (61h)

3.10.1 Read Response

This is the data response for a valid request to read the ASIC BIST results.

Byte	Description
1	Response Byte (01h)
2	Length (0Dh)
3	BIST Results (Refer to Figure 3-3)
4	Flash BIST Checksum (lsb)
5	Flash BIST Checksum
6	Flash BIST Checksum
7	Flash BIST Checksum (msb)
8	DMD Device ID (lsb)
9	DMD Device ID
10	DMD Device ID
11	DMD Device ID (msb)
12	System BIST Checksum (lsb)
13	System BIST Checksum
14	System BIST Checksum
15	System BIST Checksum (msb)
16	Checksum (Sum of bytes 1 to 15)

msb	**Byte 1: BIST Results**						*lsb*
b7	*b6*	*b5*	*b4*	*b3*	*b2*	*b1*	*b0*
System BIST Result		*DMD JTAG BIST Result*		*Flash BIST Result*		*DDR2 BIST Result*	

b(0:1) - DDR2 BIST Result
00: FAIL
01: PASS
10: Unknown
11: Not Executed

b(2:3) - Flash BIST Result
00: FAIL
01: PASS
10: Unknown
11: Not Executed

b(4:5) - DMD JTAG BIST Result
00: FAIL
01: PASS
10: Unknown
11: Not Executed

b(6:7) - System BIST Result
00: INVALID
01: VALID
10: Unknown

11: Not Executed

Figure 3-3. ASIC BIST Results Byte Format

Note: For more details, refer to the **DLPC120-Q1 Programmer's Guide (DLPU055).**

3.11 ASIC Initialization Type (31h)

This command requests a read of the ASIC Initialization Type.

Byte	Description
1	Command Byte (63h)
2	Length (00h)
3	Checksum (63h)

3.11.1 Read Response

This is the data response for a valid request to read the ASIC Initialization Type.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	ASIC Initialization Type (See Figure 3-4)
4	Checksum (Sum of bytes 1 to 3)

msb	**Byte 3**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7:0) - ASIC Initialization Type
0 – FPGA
1 – External Termination (ODT-Off)
2 – On Die Termination (ODT-On)

Figure 3-4. ASIC Initialization Branch Byte Format

3.12 Piccolo Software Version (32h)

This command requests a read of the Piccolo Software Version.

Byte	Description
1	Command Byte (65h)
2	Length (00h)
3	Checksum (65h)

3.12.1 Read Response

This is the data response for a valid request to read the Piccolo Software Version.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3	Major
4	Minor
5	Build (lsb)
6	Build (msb)

7	Checksum (Sum of bytes 1 to 6)

Version Data is of the form *Major.Minor (Build).*

3.13 Piccolo Software Status (33h)

This command requests a read of the Piccolo Software Status.

Byte	Description
1	Command Byte (67h)
2	Length (00h)
3	Checksum (67h)

Note 1: Piccolo SW status command can be used to get more information when a command fails with response byte indication Write/Read Execution Failure. The command can also be send to get information about the Piccolo SW state at the requested point of time.

3.13.1 Read Response

This is the data response for a valid request to read the Piccolo Software Status. Status is automatically cleared upon read.

Note: The status bits are subject to change during development.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3 - 6	Status – See below
7	Checksum (Sum of bytes 1 to 6)

msb	**Byte 3**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) -	SPI Overrun Detect	b(3) -	SPI Incomplete Command
b(6) -	Temperature data not in ascending order	b(2) -	SPI Command Not Available
b(5) -	Temperature Table not found	b(1) -	SPI Invalid Data
b(4) -	Video BIST Execution Failed	b(0) -	SPI Invalid Command

msb	**Byte 4**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) -	Calibration Flash Signature/Checksum invalid	b(3) -	Dimming Queue Overflow
b(6) -	Calibration Table not found in flash	b(2) -	ASIC Initialization Failure
b(5) -	Data out of Range	b(1) -	ASIC I2C Read Failure
b(4) -	On-Die Termination based Initialization	b(0) -	ASIC I2C Write Failure

msb	**Byte 5**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) -	*Reserved*	b(3) -	Calibration – Flash sector erase failed
b(6) -	Timer Error	b(2) -	Calibration Table not supported
b(5) -	Unhandled Interrupt Received	b(1) -	Calibration – Incomplete Data
b(4) -	Calibration – Flash Programming failed	b(0) -	Mismatch between Calibration Data and Command List File

msb	**Byte 6**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) -	SPI – Escape character detected	b(3) -	HRPWM Scale Factor Optimization Error
b(6) -	SPI – Length Mismatch/Other errors	b(2) -	Temperature Related Error
b(5) -	SPI – Ignored some bytes	b(1) -	Invalid TMP411 Reading
b(4) -	SPI – Checksum Mismatch	b(0) -	50-50 Sequence Enforced

3.14 ASIC Register (34h)

3.14.1 Write

This command is used to write to HUD ASIC Register. *This command is for TI test purposes only. It is not intended for normal use.*

Warning: Improper use of this command has the potential to stop the system from working.

Byte	Description
1	Command Byte (68h)
2	Length (05h)
3	ASIC Register Address
4	Register Data (lsb)
5	Register Data
6	Register Data
7	Register Data (msb)
8	Checksum (Sum of bytes 1 to 7)

The 4 byte Register Data is written to the provided ASIC Register Address via I2C.

3.14.2 Read

This command requests a read of any ASIC Register.

Byte	Description
1	Command Byte (69h)
2	Length (01h)
3	ASIC Register Address
4	Checksum (Sum of bytes 1 to 3)

3.14.3 Read Response

This is the data response for a valid request to read any ASIC Register.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3	Register Data (lsb)
4	Register Data
5	Register Data
6	Register Data (msb)
7	Checksum (Sum of bytes 1 to 6)

3.14.4 ASIC Register Details

Detailed descriptions of supported ASIC registers are available as a separate document titled “DLPC120 IIC Programmer’s Guide”.

3.15 VAC Mode (35h)

3.15.1 Write

This command is used to enable/disable VAC. This command is available only in the calibration mode.

Byte	Description
1	Command Byte (6Ah)
2	Length (03h)
3	VAC enable/disable
4	0xFF (Dummy Byte)
5	0xFF (Dummy Byte)
6	Checksum (Sum of bytes 1 to 5)

Note: The 3rd byte is used to enable/disable VAC control. When set to 1 VAC is enabled and when reset to 0, it is disabled. Bytes 4 and 5 are dummy bytes which are used to maintain the same command structure as some of the previous versions.

3.15.2 Read

This command is used to read if VAC is enabled or disabled, and get the current VAC mode switch point. This command is available only in the calibration mode.

Byte	Description
1	Command Byte (6Bh)
2	Length (00h)
3	Checksum (6Bh)

3.15.3 Read Response

This is the data response for a valid request to read the VAC mode.

Byte	Description
1	Response Byte (01h)
2	Length (03h)
3	VAC enabled/disabled
4	VAC mode switch Point (LSB)
5	VAC mode switch Point (MSB)
6	Checksum (Sum of bytes 1 to 5)

3.16 Operating Mode (36h)

This command requests a read of the current operating mode.

Byte	Description
1	Command Byte (6Dh)
2	Length (00h)
3	Checksum (6Dh)

3.16.1 Read Response

This is the data response for a valid request to read the current operating mode.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	Operating mode
4	Checksum (Sum of bytes 1 to 6)

Note: the 3rd response byte contains 1 for Continuous Mode and 2 for Discontinuous Mode

3.17 PWM Sensitivity (37h)

This command requests a read of the PWM Sensitivity. PWM sensitivity is the minimum change in PWM input for each LED that can ensure one fine (HRPWM) pulse change in the corresponding output pulse width.

Byte	Description
1	Command Byte (6Fh)
2	Length (00h)
3	Checksum (6Fh)

3.17.1 Read Response

This is the data response for a valid request to read the PWM Sensitivity.

Byte	Description
1	Response Byte (01h)
2	Length (02h)
3	PWM Sensitivity (lsb)
4	PWM Sensitivity (msb)
5	Checksum (Sum of bytes 1 to 4)

3.18 Piccolo Software Secondary Status (38h)

This command requests a read of the Piccolo Software Secondary Status. The secondary status is an extension of the 32 bit status discussed in 4.11. This provides for 32 new status bits.

Byte	Description
1	Command Byte (71h)
2	Length (00h)
3	Checksum (71h)

3.18.1 Read Response

This is the data response for a valid request to read the Piccolo Software Secondary Status. Status is automatically cleared upon read.

Note: The secondary status bits are subject to change during development.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3 - 6	Secondary Status – See below
7	Checksum (Sum of bytes 1 to 6)

msb	**Byte 3**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) - *Reserved*
b(6) - *System Reset due to Voltage Monitoring*
b(5) - *Voltage Monitoring Enabled*
b(4) - *Reserved*
b(3) - *Reserved*
b(2) - *Reserved*
b(1) - Configuration File modified after calibration
b(0) - Calibration File modified after calibration

msb	**Byte 4**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) - *Reserved*
b(6) - *Reserved*
b(5) - *Reserved*
b(4) - *Reserved*
b(3) - *Reserved*
b(2) - *Reserved*
b(1) - *Reserved*
b(0) - *Reserved*

msb	**Byte 5**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) - *Reserved*
b(6) - *Reserved*
b(5) - *Reserved*
b(4) - *Reserved*
b(3) - *Reserved*
b(2) - *Reserved*
b(1) - *Reserved*
b(0) - *Reserved*

msb	**Byte 6**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7) - *Reserved*
b(6) - *Reserved*
b(5) - *Reserved*
b(4) - *Reserved*
b(3) - *Reserved*
b(2) - *Reserved*
b(1) - *Reserved*
b(0) - *Reserved*

3.19 Extra Information Keys (39h)

The Extra Information field is a part of the configuration file that can be used to store any information in the form of 32 bit key-value pairs. For example, the key 0xB217D12F is mapped to the address in the Piccolo where the ASIC Build Information is stored. This SPI command is used to get the corresponding key for a given index. An input of ‘0’ returns the number of keys, any other index will cause the piccolo to return Key [index - 1].

3.19.1 Read

This is the data response for a valid request to read the Extra Information Key.

Byte	**Description**
1	Command Byte (73h)
2	Length (04h)
3	Key Index (lsb)
4	Key Index
5	Key Index
6	Key Index (msb)
7	Checksum (Sum of bytes 1 to 6)

Note : This command needs valid calibration data to be present in flash for successful operation.

3.19.2 Read Response

This is the data response for a valid request to read the Extra Information Key.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3	Key [Key Index - 1] (lsb)
4	Key [Key Index - 1]
5	Key [Key Index - 1]
6	Key [Key Index - 1] (msb)
7	Checksum (Sum of bytes 1 to 6)

3.20 Extra Information Values (3Ah)

This command returns the value mapped to an Extra Information Key.

3.20.1 Read

This is the data response for a valid request to read the Extra Information Value.

Byte	Description
1	Command Byte (75h)
2	Length (04h)
3	Key(lsb)
4	Key
5	Key
6	Key(msb)
7	Checksum (Sum of bytes 1 to 6)

Note : This command needs valid calibration data to be present in flash for successful operation.

3.20.2 Read Response

This is the data response for a valid request to read the Extra Information Value.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3	Value (lsb)
4	Value
5	Value
6	Value (msb)
7	Checksum (Sum of bytes 1 to 6)

3.21 Dimming LUT Group and Gamma Index (40h)

3.21.1 Write

This command is used to select any of the Dimming LUT Groups and Gamma Indexes available in Calibration Data stored in Piccolo Flash. This command is available only in the calibration mode. After this command is sent, the effects of this command will not become apparent until calibration mode is disabled. Refer to Section 3.34 for enabling and disabling calibration mode.

Byte	Description
1	Command Byte (80h)
2	Length (02h)
3	Dimming LUT Group Index (Range 0-15)
4	Gamma Index (Range 0-15)
5	Checksum (Sum of bytes 1 to 4)

Note 1: This command needs valid calibration data to be present in flash for successful operation.

Note 2: The maximum value of Dimming LUT Group and Gamma indexes supported depends upon the calibration data and command list files. The maximum value supported can be obtained by using the command in read mode.

3.21.2 Read

This command requests a read of the Dimming LUT Group and Gamma Indexes.

Byte	Description
1	Command Byte (81h)
2	Length (00h)
3	Checksum (81h)

3.21.3 Read Response

This is the data response for a valid request to read the Dimming LUT Group Index.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3	Number of Dimming LUT Groups available
4	Number of Gamma Indexes available for the current Dimming LUT Group
5	Current Dimming LUT Group index
6	Current Gamma Index
7	Checksum (Sum of bytes 1 to 6)

Note 1: Byte 3 and Byte 4 indicate the maximum number of Dimming LUT Group and Gamma Indexes present in calibration data and command list file. The Dimming LUT Group and Gamma Index, used in write mode should be less than this maximum available value for successful operation.

3.22 Dimming LUT Group Information (41h)

3.22.1 Read

This command reads the Dimming LUT Group Information for the requested Dimming LUT Group Index.

Byte	Description
1	Command Byte (83h)
2	Length (01h)
3	Dimming LUT Group Index
4	Checksum (Sum of bytes 1 to 3)

Note 1: This command needs valid calibration data to be present in flash for successful operation.

Note 2: The Dimming LUT Group Index value should be less than Number of available Dimming LUT Group Indexes in calibration/command list file.

3.22.2 Read Response

This is the data response for a valid request to read the Dimming LUT Group Information.

Byte	Description
1	Response Byte (01h)
2	Length (35h)
3	Red Duty Cycle (lsb) : See Note 1
4	Red Duty Cycle (msb)
5	Green Duty Cycle (lsb) : See Note 1
6	Green Duty Cycle (msb)
7 - 37	Dimming LUT Group Name (as ASCII value letter by letter)
38	Checksum (Sum of bytes 1 to 37)

Note 1: The Red and Green Duty Cycles are multiplied by 100 before sending the response.

For example, if the duty cycles are Red = 35, Green = 45 and Blue = 20, then,

The transmitted Red Duty Cycle = 3500 (Byte 3 = 0xAC, Byte 4 = 0x0D), Transmitted Green Duty Cycle = 4500 (Byte 5 = 0x94, Byte 6 = 0x11)

Note 2: The Blue Duty Cycle is not transmitted back in the command. This can be easily computed as below:

Blue Duty Cycle = 100 - (Transmitted Red DC + Transmitted Green DC) / 100

Note 3: The Dimming LUT Group Name is transmitted letter by letter using the ASCII value of each letter as one byte. The ASCII value of the first letter is sent as the first byte, followed by the second letter, and so on.

3.23 CMT/Gamma Information (43h)

3.23.1 Read

This command reads the CMT/Gamma Information for the requested Dimming LUT Group and CMT/Gamma Index.

Byte	Description
1	Command Byte (87h)
2	Length (02h)
3	Dimming LUT Group Index
3	CMT/Gamma Index
4	Checksum (Sum of bytes 1 to 4)

Note 1: This command needs valid calibration data to be present in flash for successful operation.

Note 2: The CMT/Gamma Index value should be less than Number of available CMT/Gamma indexes for the corresponding Dimming LUT Group in calibration/command list file. To read the number of Gamma indexes available for any Dimming LUT Group, use this command with arguments as the Dimming LUT Group and gamma index 0 (as index 0 is valid for all Dimming LUT Groups).

3.23.2 Read Response

This is the data response for a valid request to read the CMT/Gamma Information.

Byte	Description
1	Response Byte (01h)
2	Length (32h)
3	Number of CMT/Gamma Indexes available for the requested Dimming LUT Group Index
4-34	ASCII name of the CMT/Gamma as defined in Calibration File
35	Checksum (Sum of bytes 1 to 34)

3.24 Command List Address (4Eh)

3.24.1 Read

This command reads the start address of the Splash (Type = 1), Test Pattern (Type = 2), External Video (Type = 3) or Generic (Type = 4 + Generic Type Index) command list specified as per the argument. Generic command lists are further classified into Generic Types. To get the address of the type N (starting from 0) the 3rd byte must be (4 + N).

Byte	Description
1	Command Byte (9Dh)
2	Length (02h)
3	Command List Type (1,2,3, or (4 + Generic Type Index))
4	Command List Index
5	Checksum (Sum of bytes 1 to 4)

Note 1: This command needs valid calibration data to be present in flash for successful operation.

3.24.2 Read Response

This is the data response for a valid request to read the number Command List start address.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3	Start Address (lsb)
4	Start Address
5	Start Address
6	Start Address (msb)
7	Checksum (Sum of bytes 1 to 6)

Note: Any invalid type (less than 1 or greater than 3 + Number of Generic Types) will cause the command to fail and flag a ‘data out of range’ error.

3.25 Generic Command List Type (4Fh)

3.25.1 Read

This command reads the Type of the Generic Command List given a Type Index. Generic Command List Type is used to group Generic Command Lists based on functionality. For example, all Command Lists that are related to flipping images may have the type as ‘Flip’. The Piccolo Software uses Command Lists of type ‘Internal’ to execute the System BIST (refer to Piccolo software for the **Piccolo Software Programmer's Guide for the DLPC120 ASIC (DLPU061)**) for more details on System BIST.

Byte	Description
1	Command Byte (9Fh)
2	Length (01h)
3	Generic Command List Type Index
4	Checksum (Sum of bytes 1 to 3)

Note 1: This command needs valid calibration data to be present in flash for successful operation.

3.25.2 Read Response

This is the data response for a valid request to read the Generic Command List type. The type is returned in the form of ASCII values of each character one by one from the start to the end of the string.

Byte	Description
1	Response Byte (01h)
2	Length (1Fh)
3 - 33	ASCII Value of Each Character of the Command List Type (from first to last)
34	Checksum (Sum of bytes 1 to 33)

3.26 Command List Numbers (50h)

3.26.1 Read

This command reads the number of available Splash (Type 1), Test Pattern (Type 2), External Video (Type 3), or Generic (Type = 4 + Generic Type Index) command lists as per the argument. It also returns the number of Generic Command List Types available.

Byte	Description
1	Command Byte (A1h)
2	Length (01h)
3	Command List Type (1/2/3/(4 + Generic Type Index))
4	Checksum (Sum of bytes 1 to 3)

Note 1: This command needs valid calibration data to be present in flash for successful operation.

3.26.2 Read Response

This is the data response for a valid request to read the number of Command Lists.

Byte	Description
1	Response Byte (01h)
2	Length (02h)
3	Number of command lists
4	Number of Generic Command List Types
5	Checksum (Sum of bytes 1 to 4)

Note: Any invalid type (less than 1 or greater than 3 + Number of Generic Types) will cause the command to fail and flag a ‘data out of range’ error.

3.27 Read Command Table Info / Execute Command List (51h)

3.27.1 Write

This command is used to execute a Command List which is referenced by its type and index sent as arguments. Currently available types are Splash (Type 1), Test Pattern (Type 2), External Video (Type 3) and Generic (Type = 4 + Generic Type Index).

Byte	Description
1	Command Byte (A2h)
2	Length (02h)
3	Command List Type (1/2/3/(4 + Generic Type Index))
4	Command List Index
5	Checksum (Sum of bytes 1 to 4)

The Command List index must be less than the number of available Command Lists of the type.

This number can be read by using the Read Command List Numbers command (0x50)

3.27.2 Read

This command requests a read of the Command List details.

Byte	Description
1	Command Byte (A3h)
2	Length (02h)
3	Command List Type (1/2/3/(4 + Generic Type Index)))
4	Command List Index
5	Checksum (Sum of bytes 1 to 4)

3.27.3 Read Response

This is the data response for a valid request to read the Command List Info.

For types 1, 2, and 4:

Byte	Description
1	Response Byte (01h)
2	Length (31h)
3 - 30	Command List Name (ASCII values of each character)
31	Checksum (Sum of bytes 1 to 30)

For type 3:

Byte	Description
1	Response Byte (01h)
2	Length (0Bh)
3	Horizontal Resolution (lsb)
4	Horizontal Resolution (msb)
5	Vertical Resolution (lsb)
6	Vertical Resolution (msb)
7	Frequency
8	Output Horizontal Resolution (lsb)
9	Output Horizontal Resolution (msb)
10	Output Vertical Resolution (lsb)
11	Output Vertical Resolution (msb)
12	Checksum (Sum of bytes 1 to 11)

Note: Any invalid type (less than 1 or greater than 3 + Number of Generic Types) will cause the command to fail and flag a ‘data out of range’ error.

3.28 Front End Video BIST Pixels (53h)

3.28.1 Write

This command is used to set the x and y coordinates of the start pixel and end pixel of the region for which checksum is to be completed.

Byte	Description
1	Command Byte (A6h)
2	Length (08h)
3	Start Pixel x-coordinate (lsb)
4	Start Pixel x-coordinate (msb)
5	Start Pixel y-coordinate (lsb)
6	Start Pixel y-coordinate (msb)
7	End Pixel x-coordinate (lsb)

8	End Pixel x-coordinate (msb)
9	End Pixel y-coordinate (lsb)
10	End Pixel y-coordinate (msb)
11	Checksum (Sum of bytes 1 to 10)

3.28.2 Read

This command requests a read of the Front End Video BIST Pixels.

Byte	Description
1	Command Byte (A7h)
2	Length (00h)
3	Checksum (Sum of bytes 1 and 2)

3.28.3 Read Response

This is the data response for a valid request to read the Front End Video BIST Pixels.

Byte	Description
1	Response Byte (01h)
2	Length (08h)
3	Start Pixel x-coordinate (lsb)
4	Start Pixel x-coordinate (msb)
5	Start Pixel y-coordinate (lsb)
6	Start Pixel y-coordinate (msb)
7	End Pixel x-coordinate (lsb)
8	End Pixel x-coordinate (msb)
9	End Pixel y-coordinate (lsb)
10	End Pixel y-coordinate (msb)
11	Checksum (Sum of bytes 1 to 10)

Note: For more details, refer to the **DLPC120-Q1 Programmer's Guide (DLPU055).**

3.29 Front End Video BIST (54h)

3.29.1 Write

This command is used to execute the Front End Video BIST.

Byte	Description
1	Command Byte (A8h)
2	Length (00h)
3	Checksum (Sum of bytes 1 and 2)

3.29.2 Read

This command requests a read of the Front End Video BIST Result and Checksum.

Byte	Description
1	Command Byte (A9h)
2	Length (00h)
3	Checksum (Sum of bytes 1 and 2)

3.29.3 Read Response

This is the data response for a valid request to read the Front End Video BIST Result and Checksum.

Byte	Description
1	Response Byte (01h)
2	Length (5h)
3	Front End Video BIST result
4	Front End Video BIST checksum (lsb)
5	Front End Video BIST checksum
6	Front End Video BIST checksum
7	Front End Video BIST checksum (msb)
8	Checksum (Sum of bytes 1 to 7)

Front End BIST Result Key:
00: FAIL
01: PASS
10: Unknown
11: Not Executed

Note: For more details, refer to the **DLPC120-Q1 Programmer's Guide (DLPU055).**.

3.30 External Video Detect BIST (55h)

3.30.1 Write

This command is used to execute the External Video Detect BIST.

Byte	Description
1	Command Byte (AA)
2	Length (08h)
3	Execution Type
4	Enable Command List Execution
5	Pass Command List Type
6	Pass Command List Index
7	Fail Command List Type
8	Fail Command List Index
9	Maximum Frame Rate (fps)
10	Minimum Frame Rate (fps)
11	Checksum (Sum of bytes 1 to 10)

Figure 3-5 shows the description of all parameters for the write command.

Parameter	Description
Execution Type	Determines the action performed: 0: Disable BIST 1: Enable BIST and allow it to run continuously until disabled. Does not update status and mux reports 2: Runs BIST four times, updates associated mux reports and status and then disables BIST
Enable Command List Execution	Determines whether to execute command list on pass/fail. Must be set to 1 for Execution type = 1 0: Do not execute Command List 1: Execute Command List
Pass Command List Type	Type of the Command List to be executed when BIST passes. 1: Splash 2: Test Pattern 3: External Video 4: Generic
Pass Command List Index	Index of the Command List to be executed when BIST passes.

Fail Command List Type	Type of the Command List to be executed when BIST fails. 1: Splash 2: Test Pattern 3: External Video 4: Generic
Fail Command List Index	Index of the Command List to be executed when BIST fails.
Maximum Frame Rate (fps)	Maximum Frame rate (in frames per second)
Minimum Frame Rate (fps)	Minimum Frame rate (in frames per second)

Figure 3-5. Description of External Video Detect BIST Write Parameters

3.30.2 Read

This command requests a read of the External Video BIST Result and MUX reports.

Byte	Description
1	Command Byte (ABh)
2	Length (00h)
3	Checksum (Sum of bytes 1 and 2)

3.30.3 Read Response

This is the data response for a valid request to read the External Video BIST Result and MUX reports.

Byte	Description
1	Response Byte (01h)
2	Length (17h)
3	External Video Detect BIST result
4	Vsync report (lsb)
5	Vsync report
6	Vsync report
7	Vsync report (msb)
8	Pixel Clock report (lsb)
9	Pixel Clock report
10	Pixel Clock report
11	Pixel Clock report (msb)
12	Active Lines report (lsb)
13	Active Lines report
14	Active Lines report
15	Active Lines report (msb)
16	Active Pixels report (lsb)
17	Active Pixels report
18	Active Pixels report
19	Active Pixels report (msb)
20	Checksum (Sum of bytes 1 to 19)

Parameter	Description
External Video Detect BIST result	Result of the 4 External Video BIST's run Bytes (0,1): Result for BIST using Mux Select = Vsync Bytes (2,3): Result for BIST using Mux Select = Pixel Clock Bytes (4,5): Result for BIST using Mux Select = Active Line Bytes (6,7): Result for BIST using Mux Select = Active Pixels Result: 00 – BIST FAIL 01 – BIST PASS 10 – BIST RESULT UNKNOWN (BIST timed out) 11 – BIST NOT EXECUTED

Vsync report	Vsync Period in number of 78-MHz clock pulses*
Pixel Clock report	Number of Pixel Clocks in Test Period (1023 pulses of a 78MHz clock*)
Active Lines report	Number of Active Pixels in the first active line.
Active Pixels report	Number of Active Lines per active frame.
	*Frequency might vary depending on the PLL spreading

Figure 3-6. Description of External Video Detect BIST Read Response Parameters

Note: For more details, refer to the **DLPC120-Q1 Programmer's Guide (DLPU055).**.

3.31 Low Pass Filter Constants (60h)

The low pass filter is used to smoothen any sharp changes in temperature. The operation of the low pass filter is based on 2 constants: Strength and Quantization step. There are two stages of operation – Filtering and Quantization.

1. Filtering and determination of output temperature of filter:

   Let T1(n) be the last output temperature of the filter and T2(n + 1) the new reading from the TMP411.

   *New output temperature T1(n + 1) = g * T2(n + 1) - (1 - g)*T1(n)*

2. Quantization:

   Before using it for dimming, the output temperature can be quantized to a desired level as indicated by the ‘Quantization Step’ (Qstep) property of the Filter.

   If T1(n + 1) >Last used Active Temperature (round down)

   Active Temperature(Used for dimming) = T1(n + 1) - modulo(T1(n + 1), Qstep)

   Else (round up)

   Active Temperature(Used for dimming) = T1(n + 1) - modulo(T1(n + 1), Qstep) + Qstep

3.31.1 Write

This command is used to set the strength and quantization step of the low pass temperature filter.

Byte	Description
1	Command Byte (C0h)
2	Length (08h)
3	Low Pass Filter Strength (lsb)
4	Low Pass Filter Strength
5	Low Pass Filter Strength
6	Low Pass Filter Strength (msb)
7	Low Pass Filter Quantization Step (lsb)
8	Low Pass Filter Quantization Step
9	Low Pass Filter Quantization Step
10	Low Pass Filter Quantization Step (msb)
11	Checksum (Sum of bytes 1 to 10)

For more information on the working of the low pass filter refer to the Temperature Compensation section of the **Piccolo Software Programer's Guide for the DLPC120 ASIC User’s Guide (DLPU061)**.

3.31.2 Read

This command requests a read of the Low Pass Filter Constants.

Byte	Description
1	Command Byte (C1h)
2	Length (00h)
3	Checksum (C1h)

3.31.3 Read Response

This is the data response for a request to read the Low Pass Filter Constants.

Byte	**Description**
1	Response Byte (01h)
2	Length (08h)
3	Low Pass Filter Strength (lsb)
4	Low Pass Filter Strength
5	Low Pass Filter Strength
6	Low Pass Filter Strength (msb)
7	Low Pass Filter Quantization Step (lsb)
8	Low Pass Filter Quantization Step
9	Low Pass Filter Quantization Step
10	Low Pass Filter Quantization Step (msb)
11	Checksum (Sum of bytes 1 to 10)

Note: The Filter Coefficients are floating point numbers. Please refer to Chapter 5 for the read and write format for such numbers.

3.32 Temperature Compensation (61h)

3.32.1 Write

This command is used to set up Temperature Compensation Function. This function needs pre-calibrated data to be present to function correctly.

Byte	**Description**
1	Command Byte (C2h)
2	Length (03h)
3	Function Enable : See Below
4	Frequency (0 - 7)
5	Custom Temperature : See Below
6	Checksum

msb	**Byte 3: Function Enable**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0
Reserved				Measurement Mode			Enable

b(0) - Function Enable
0: Disable Temperature Compensation
1: Enable Temperature Compensation

b(3:1) - Measurement Mode
0 – *Reserved (Do not use)*
1 – User defined temperature input
2 – Temperature measured from TMP411
3 to 7 – *Reserved (Do not use)*

b(7:4) - *Reserved*

Figure 3-7. Temperature Compensation (Byte 3)

msb		**Byte 4: Reserved**						*lsb*
b7		b6	b5	b4	b3	b2	b1	b0
	Reserved (Should be set to zero)							

b(7:0) - *Reserved (Should be set to zero)*

Figure 3-8. Temperature Compensation (Byte 4)

msb	**Byte 5: Custom Temperature**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0
Custom Temperature							

b(7:0) - Custom Temperature
The value in Celsius temperature scale should be offset by 100
Eg: 0°C should be send as 100
–35°C should be send as 65 etc

Figure 3-9. Temperature Compensation (Byte 5)

This command supports temperatures in the range –100°C to 155°C. To support transmitting the temperature in a single byte, an offset of 100 is added to the temperature post which the range becomes 0 (–100°C) to 255 (155°C).

Note 1: Measurement Mode determines the temperature used in the compensation calculations.

Note 2: In User defined mode, the temperature value supplied in Byte 5 is used directly.

Note 3: In TMP411 mode, the software uses the temperature measured by TMP411 sensor for the calculations.

Note 4: Details of Temperature Compensation algorithm is available in the **LED Driver for DLP3030-Q1 Displays Application Note**.

Note 5: Frequencies of 0 to 7 are permitted. 0 corresponds to 1 Hz, and 7 to 8 Hz.

3.32.2 Read

This command requests a read of Temperature Compensation Settings.

Byte	Description
1	Command Byte (C3h)
2	Length (00h)
3	Checksum

3.32.3 Read Response

This is the data response for a valid request to read the temperature compensation settings.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3	Function Enable : See Figure 23
4	Frequency – 1 Hz
5	Custom Temperature
6	Active Temperature : See below
7	Checksum (Sum of bytes 1 to 6)

Active Temperature is the temperature at which the Temperature Compensation algorithm runs when enabled.

Active temperature is formatted the same manner as Custom Temperature.

msb	**Byte 5: Active Temperature**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0
Active Temperature							

b(7:0) - Active Temperature
The value returned is offset by 100 degree Celsius
Eg: 100 means 0°C
65 means –35°C etc

Figure 3-10. Active Temperature Byte Format

3.33 LED Voltage and Current (62h)

This command returns the measured LED Voltage and Current.

3.33.1 Read

This command requests a read of the LED Voltage and Current.

Byte	**Description**
1	Command Byte (C5h)
2	Length (00h)
3	Checksum (C5h)

3.33.2 Read Response

This is the data response for a request to read the LED Voltage and Current.

Byte	**Description**
1	Response Byte (01h)
2	Length (08h)
3	LED Voltage (lsb)
4	LED Voltage
5	LED Voltage
6	LED Voltage (msb)
7	LED Current (lsb)
8	LED Current
9	LED Current
10	LED Current (msb)
11	Checksum (Sum of bytes 1 to 10)

The LED Voltages and Currents are floating point numbers. Please refer to Section 5.3 for the format in which it is received.

3.34 DMD Temperature (63h)

3.34.1 Read

This command reads the DMD temperature in K10 format.

Byte	**Description**
1	Command Byte (C7h)
2	Length (00h)
3	Checksum (C6h)

3.34.2 Read Response

This is the data response for a request to read the DMD Temperature.

Byte	Description
1	Response Byte (01h)
2	Length (02h)
3	Temperature in K10 (lsb)
4	Temperature in K10 (msb)
5	Checksum (Sum of bytes 1 to 4)

3.34.3 Example: K10 Format:

Let us assume the received value of byte 3 to be 0xA4 and that of byte 4 to be 0x0B. Hence the temperature in K10 format is 0x0BA4. In decimal form the temperature is 2890 K10. To convert it into Celsius scale we divide by ten and subtract 273 from the result.

Temperature (in °C) = (Temperature (in k10) / 10) – 273

= 298 – 273 = 25 C

3.35 Enable Calibration Mode (64h)

3.35.1 Write

Piccolo software has two modes – Normal Mode and Calibration Mode. Normal Mode is the regular mode when the system allows dimming operations. Calibration Mode or Cal Mode for short can be considered as a factory operating mode, where the user has direct access to LED PWMs and other low-level functions.

This command is used to change between calibration and normal modes.

Byte	Description
1	Command Byte (C8h)
2	Length (01h)
3	Calibration Mode (See below)
4	Checksum (Sum of bytes 1 to 3)

msb	Byte 3						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7:0) -	Calibration Mode 0 – Disable Calibration Mode 1 – Enable Calibration Mode

Figure 3-11. Calibration Mode Byte Format

Note 1: This command can be sent in both normal and calibration mode. The system starts up in normal mode by default.

3.35.2 Read

This command requests a read of the operating mode (calibration/normal).

Byte	Description
1	Command Byte (C9h)
2	Length (00h)
3	Checksum (C9h)

3.35.3 Read Response

This is the data response for a valid request to read the operating Mode.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	Calibration Mode (See Figure 3-11)
4	Checksum (Sum of bytes 1 to 3)

3.36 Red LED PWM (65h)

3.36.1 Write

This command is used to set the Red LED PWM Level. This command can be used only in calibration mode (see Section 3.34 to enable calibration mode).

Byte	Description
1	Command Byte (CAh)
2	Length (02h)
3	Red LED PWM (lsb)
4	Red LED PWM (msb)
5	Checksum (Sum of bytes 1 to 4)

3.36.2 Read

This command requests a read of the Red LED PWM Level.

Byte	Description
1	Command Byte (CBh)
2	Length (00h)
3	Checksum (CBh)

3.36.3 Read Response

This is the data response for a valid read command.

Byte	Description
1	Response Byte (01h)
2	Length (02h)
3	Red LED PWM (lsb)
4	Red LED PWM (msb)
5	Checksum (Sum of bytes 1 to 4)

3.37 Green LED PWM (66h)

3.37.1 Write

This command is used to set the Green LED PWM Level. This command can be used only in calibration mode (see Section 3.34 to enable calibration mode).

Byte	Description
1	Command Byte (CCh)
2	Length (02h)
3	Green LED PWM (lsb)

4	Green LED PWM (msb)
5	Checksum (Sum of bytes 1 to 4)

3.37.2 Read

This command requests a read of the Green LED PWM Level.

Byte	Description
1	Command Byte (CDh)
2	Length (00h)
3	Checksum (CDh)

3.37.3 Read Response

This is the data response for a valid read command.

Byte	Description
1	Response Byte (01h)
2	Length (02h)
3	Green LED PWM (lsb)
4	Green LED PWM (msb)
5	Checksum (Sum of bytes 1 to 4)

3.38 Blue LED PWM (67h)

3.38.1 Write

This command is used to set the Blue LED PWM Level. This command can be used only in calibration mode (see Section 3.34 to enable calibration mode).

Byte	Description
1	Command Byte (CEh)
2	Length (02h)
3	Blue LED PWM (lsb)
4	Blue LED PWM (msb)
5	Checksum (Sum of bytes 1 to 4)

3.38.2 Read

This command requests a read of the Blue LED PWM Level.

Byte	Description
1	Command Byte (CFh)
2	Length (00h)
3	Checksum (CFh)

3.38.3 Read Response

This is the data response for a valid read command.

Byte	Description
1	Response Byte (01h)
2	Length (02h)
3	Blue LED PWM (lsb)

4	Blue LED PWM (msb)
5	Checksum (Sum of bytes 1 to 4)

3.39 Current Limit PWM (68h)

3.39.1 Write

This command is used to set the Current Limit PWM Level. This command can be used only in calibration mode (see Section 3.34 to enable calibration mode).

Byte	Description
1	Command Byte (D0h)
2	Length (02h)
3	Current Limit PWM (lsb)
4	Current Limit PWM (msb)
5	Checksum (Sum of bytes 1 to 4)

3.39.2 Read

This command requests a read of the Current Limit PWM Level.

Byte	Description
1	Command Byte (D1h)
2	Length (00h)
3	Checksum (D1h)

3.39.3 Read Response

This is the data response for a valid read command.

Byte	Description
1	Response Byte (01h)
2	Length (02h)
3	Current Limit PWM (lsb)
4	Current Limit PWM (msb)
5	Checksum (Sum of bytes 1 to 4)

3.40 Sensor Gain (69h)

3.40.1 Write

This command is used to set the sensor gain. This command is available only in the calibration mode (see Section 3.35 to enable calibration mode).

Byte	Description
1	Command Byte (D2h)
2	Length (01h)
3	Sensor Gain
4	Checksum (Sum of bytes 1 to 3)

The sensor gain can only have values ‘0’, ‘1’, ‘2’ and ‘3’ as it is an input to the two select bits of a 4 to 1 Mux. The actual gain applied is hardware dependent and is stored in the calibration file. It can be obtained by using the Sensor Gain Map as described in Section 3.42.

3.40.2 Read

This command requests a read of the sensor gain.

Byte	Description
1	Command Byte (D3h)
2	Length (00h)
3	Checksum (D3h)

3.40.3 Read Response

This is the data response for a valid request to read the sensor gain.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	Sensor Gain
4	Checksum (Sum of bytes 1 to 3)

The read command will return a value in the range 0 – 3. The real applied gain can be obtained by using the Sensor Gain Map Command.

3.41 Command Table Index (6Ah)

3.41.1 Write

This command is used to set the ASIC command table index registers. This command is available only in the calibration mode (see Section 3.34 to enable calibration mode).

Byte	Description
1	Command Byte (D4h)
2	Length (01h)
3	LDC Index (See below)
4	Checksum (Sum of bytes 1 to 3)

msb	**Byte 3**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7:0) - LDC Index

Figure 3-12. Command Table Index Byte Format

Note 1: This command needs valid calibration data to be present in Piccolo flash memory for proper operation.

Note 2: This command will compute the SEQ Index and CMT Index based on loaded calibration file and other governing settings. The command will then write to all three command table index registers (LDC Index, SEQ Index and CMT Index).

Note 3: Details of the ASIC Registers are available in “ASIC IIC Guide” document.

3.41.2 Read

This command requests a read of the command table index registers of the HUD ASIC.

Byte	Description
1	Command Byte (D5h)
2	Length (00h)
3	Checksum (D5h)

3.41.3 *Read Response*

This is the data response for a valid request to read the command table index.

Byte	Description
1	Response Byte (01h)
2	Length (03h)
3	LDC Index (See Figure 28
4	SEQ Index
5	CMT Index
6	Checksum (Sum of bytes 1 to 5)

3.42 Sensor Gain Map (6Bh)

This command requests a read of the Sensor Gain Mapping present in the calibration data, which maps the Mux input to the actual gain applied.

Byte	Description
1	Command Byte (D7h)
2	Length (00h)
3	Checksum (D7h)

3.42.1 *Read Response*

This is the data response for a valid request to read the Sensor Gain Mapping.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3	Actual Gain for Mux Input 0 (Integer)
4	Actual Gain for Mux Input 1 (Integer)
5	Actual Gain for Mux Input 2 (Integer)
6	Actual Gain for Mux Input 3 (Integer)
7	Checksum (Sum of bytes 1 to 6)

3.43 Adapter ADC Voltages (0x6C)

This command gets the voltages of the three Adapter ADCs connected to channels A3, A6 and A7. On receiving this command, the Piccolo triggers ADC conversions to poll these channels.

3.43.1 *Read*

This command requests a read of the Adapter ADC voltages.

Byte	Description
1	Command Byte (D9h)
2	Length (00h)
3	Checksum (D9h)

3.43.2 Read Response

This is the data response for a request to read Adapter ADC voltages.

Byte	Description
1	Response Byte (01h)
2	Length (0Ch)
3	Adapter ADC channel A3 Voltage (lsb)
4	Adapter ADC channel A3 Voltage
5	Adapter ADC channel A3 Voltage
6	Adapter ADC channel A3 Voltage (msb)
7	Adapter ADC channel A6 Voltage (lsb)
8	Adapter ADC channel A6 Voltage
9	Adapter ADC channel A6 Voltage
10	Adapter ADC channel A6 Voltage (msb)
11	Adapter ADC channel A7 Voltage (lsb)
12	Adapter ADC channel A7 Voltage
13	Adapter ADC channel A7 Voltage
14	Adapter ADC channel A7 Voltage (msb)
15	Checksum (sum of bytes 1 to 14)

Note: The Adapter ADC Voltages are floating point numbers. Please refer to Section 5.3 for the read format for such numbers.

3.44 Configuration Format Version (6Dh)

This command requests a read of the Configuration Data Format Version supported by Piccolo Software.

Byte	Description
1	Command Byte (DBh)
2	Length (00h)
3	Checksum (DBh)

Note 1: This command works even when there is no valid calibration data present in Piccolo Internal Flash.

3.44.1 Read Response

This is the data response for a valid request to read the Configuration Data Format Version.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3	Format Version Character ASCII (lsb)
4	Format Version Character ASCII
5	Format Version Character ASCII
6	Format Version Character ASCII (msb)
7	Checksum (Sum of bytes 1 to 6)

3.44.2 Description of Configuration Format

Configuration Format Version is used to identify the internal memory format changes done to the Configuration Data File. This helps ensure that the internal Configuration data is interpreted by the Piccolo SW correctly. The calibration format is defined as a 4 character string (eg: 0008). Taking 0008 as the example, the command returns the Format Version as 0x38, 0x30, 0x30, 0x30 as its data bytes.

Please refer to **Piccolo Software Programmer's Guide for the DLPC120 ASIC (DLPU061)** for more details about Calibration Format Version.

3.45 Calibration Format Version (6Eh)

This command requests a read of the Calibration Data Format Version supported by Piccolo Software.

Byte	Description
1	Command Byte (DDh)
2	Length (00h)
3	Checksum (DDh)

Note 1: This command works even when there is no valid calibration data present in Piccolo Internal Flash.

3.45.1 Read Response

This is the data response for a valid request to read the Calibration Data Format Version.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3	Format Version Character ASCII (lsb)
4	Format Version Character ASCII
5	Format Version Character ASCII
6	Format Version Character ASCII (msb)
7	Checksum (Sum of bytes 1 to 6)

3.45.2 Description of Calibration Format

Calibration Format Version is used to identify the internal memory format changes done to the Calibration Data File. This helps ensure that the internal calibration data is interpreted by the Piccolo SW correctly. The calibration format is defined as a 4 character string (eg: 0006). Taking 0006 as the example, the command returns the Format Version as 0x36, 0x30, 0x30, 0x30 as its data bytes.

Please refer to **Piccolo Software Programmer's Guide for the DLPC120 ASIC (DLPU061)** for more details about Calibration Format Version.

3.46 Calibration Data Version (6Fh)

This command requests a read of the Version of Calibration Data stored in Piccolo flash and the identifier for the ASIC flash file.

Byte	Description
1	Command Byte (DFh)
2	Length (00h)
3	Checksum (DFh)

Note 1: This command needs valid calibration data to be present in flash for successful operation.

3.46.1 Read Response

This is the data response for a valid request to read the Calibration Data Version and the ASIC Flash File Id.

Byte	Description
1	Response Byte (01h)
2	Length (08h)
3	Calibration Data Version (lsb)
4	Calibration Data Version
5	Calibration Data Version

6	Calibration Data Version (msb)
7	ASIC Flash File Id (lsb)
8	ASIC Flash File Id
9	ASIC Flash File Id
10	ASIC Flash File Id (msb)
11	Checksum (Sum of bytes 1 to 10)

3.47 Program Calibration Data to Flash (70h)

3.47.1 Write

This command is used to program the Calibration Data to flash. This command supports variable length. Since the SPI Protocol supports only 255 bytes of data per command, the calibration data to program should be split across multiple commands to transmit the complete data.

This command is only supported in Calibration Mode.

Byte	Description
1	Command Byte (E0h)
2	Length
3	Command Flag (See below)
4	Data
..	Data
..	Checksum

msb	**Byte 3: Command Flag**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

b (7:0) -

0 – This command contains the complete data
1 – This command contains the beginning of data
2 – This command contains intermediate data
3 – This command contains the last data

Figure 3-13. Command Flag

The command flag indicates the position of the data sent by the master.

If the calibration data is less than 254 bytes, command flag ‘0’ can be used to program the flash in a single step. In this mode, the internal flash sector which stores the calibration data will be erased and programmed with the received data. The software will then verify the validity of the data post which it will be used for new dimming computations. Appropriate status flags will be set if the programmed data is found to be invalid.

For calibration data exceeding 254 bytes, this command should be issued multiple times with the command flag set to ‘1’ for first 254 bytes, ‘2’ for all intermediate bytes and ‘3’ for the last set of bytes (less than or equal to 254).

When Command flag = ‘1’, the software erases the internal flash sector allotted for calibration data and stores the received data.

When Command flag = ‘2’, the software will program the received data to that location in flash where it stopped in the last step.

When Command flag = ‘3’, the received data will first be programmed to flash in the same way as command flag = ‘2’. After this, the validity of the calibration data is checked. The command will fail if the calibration data programmed is not valid.

The verification of new calibration data is done only after overwriting existing contents of flash with new data. Therefore, utmost care should be taken in forming the calibration data to program to flash. It is recommended to use TI provided tools to prepare and program calibration data to Piccolo flash. The system will not work without presence of proper calibration data.

Command Flag	Beginning (1)	Intermediate (2)	Last (3)
Data Bytes	0-254	255-509	510-764

3.48 Binary Flash Read/Set Binary Read Start Address (71h)

The Binary flash read function is used to read the piccolo flash data in binary format. This is useful to read out the currently used calibration data. The function is versatile and allows the reading of any part of the flash from any address. This is done by setting the start address by using the Set Binary Read start address function and then, using binary flash read with the required number of words as argument.

3.48.1 Write

This command is used to set flash binary read start address. The next binary read performed will be from the start address.

Byte	**Description**
1	Command Byte (E2h)
2	Length (04h)
3	Read Start Address (lsb)
4	Read Start Address
5	Read Start Address
6	Read Start Address (msb)
7	Checksum (Sum of bytes 1 to 6)

3.48.2 Read

This command requests a binary read of the Piccolo Flash. The number of words to be read is given as input.

Byte	**Description**
1	Command Byte (E3h)
2	Length (01h)
3	Number of words (1 - 127)
4	Checksum (Sum of bytes 1 to 3)

3.48.3 Read Response

This is the data response for a valid read command.

Byte	**Description**
1	Response Byte (01h)
2	Length (255)
3 to 257	Binary Flash Data
258	Checksum (Sum of bytes 1 to 257)

The values of all bytes after the required number of words are set to 0.

3.49 Set PWM Period/Read PWM Info (72h)

3.49.1 Write

This command should be used only in Calibration mode. This command sets the PWM Period in terms of coarse pulses to a maximum of 1200 pulses. Trying to set 0 or a higher value than 1200 will result in a ‘Data out of range’ error being flagged.

Byte	Description
1	Command Byte (E4h)
2	Length (02h)
3	PWM Period (lsb)
4	PWM Period (msb)
5	Checksum (Sum of bytes 1 to 4)

3.49.2 Read

This command requests a read of the PWM related information.

Byte	Description
1	Command Byte (E5h)
2	Length (00h)
3	Checksum (E5h)

3.49.3 Read Response

This is the data response for a valid request to read PWM related information.

Byte	Description
1	Response Byte (01h)
2	Length (08h)
3	PWM Period (lsb)
4	PWM Period (msb)
5	PWM Frequency(kHz) * 100 (lsb)
6	PWM Frequency(kHz) * 100
7	PWM Frequency(kHz) * 100
8	PWM Frequency(kHz) * 100 (msb)
9	Maximum Resolution (lsb)
10	Maximum Resolution (msb)
11	Checksum (Sum of bytes 1 to 10)

3.50 PWM Scale Factor (73h)

This command requests a read of the current, maximum and minimum HRPWM MEP (Micro Edge Positioning) scale factor since the most recent reset/power on. The scale factor is the number of fine (HRPWM) pulses per coarse pulse.

Byte	Description
1	Command Byte (E7h)
2	Length (00h)
3	Checksum (E7h)

3.50.1 Read Response

This is the data response for a valid request to read the PWM Scale Factor.

Byte	Description
1	Response Byte (01h)
2	Length (06h)
3	Current PWM MEP Scale Factor (lsb)
4	Current PWM MEP Scale Factor (msb)
5	Minimum PWM MEP Scale Factor (lsb)
6	Minimum PWM MEP Scale Factor (msb))
7	Maximum PWM MEP Scale Factor (lsb)
8	Maximum PWM MEP Scale Factor (msb)
9	Checksum (Sum of bytes 1 to 8)

3.51 ASIC Flash Read (74h)

This command requests a read of the ASIC Flash Data. The number of bytes required is sent as input. This command is used to read the User Information from the ASIC flash. This command starts reading from the address set by the ASIC Flash Read Setup Command (Section 3.51).

Byte	Description
1	Command Byte (E9h)
2	Length (01h)
3	Number of bytes to read
4	Checksum (Sum of bytes 1 to 3)

3.51.1 Read Response

This is the data response for a valid request to read the ASIC Flash Data.

Byte	Description
1	Response Byte (01h)
2	Length (Requested number of bytes provided as input)
3 to (Requested number of bytes + 2)	ASIC Flash Data
Requested number of bytes + 3	Checksum (Sum of bytes 1 to Requested number of bytes + 2)

3.52 ASIC Flash Read Setup (75h)

3.52.1 Write

This command sets the Start Address from which the ASIC Flash Read (Section 3.50) command will start reading, and Read Length. Though each ASIC Flash Read command has its own requested length, this Read Length is used to tell the ASIC when to stop the read operation. The Read Length specifies the total number of bytes to be read from the Flash (which will be read using repetitive ASIC Flash Read commands with their own lengths, limited to a maximum of 255 bytes per command). The Piccolo keeps track of the number of bytes read (sum of read lengths of the individual ASIC Flash Read Commands). When the number of bytes read becomes equal to the Read Length set through this command, the Piccolo performs certain register writes to disable the read operation.

Byte	Description
1	Command Byte (EAh)
2	Length (08h)
3	Start Address (lsb)
4	Start Address
5	Start Address
6	Start Address (msb)
7	Read Length (lsb)

8	Read Length
9	Read Length
10	Read Length (msb)
11	Checksum (Sum of bytes 1 to 10)

3.52.2 Read

This command requests a read of the ASIC Flash Read Info.

Byte	Description
1	Command Byte (EBh)
2	Length (00h)
3	Checksum (EBh)

3.52.3 Read Response

This is the data response for a valid request to read ASIC Flash Read Info.

Byte	Description
1	Response Byte (01h)
2	Length (0Ch)
3	Start Address (lsb)
4	Start Address
5	Start Address
6	Start Address (msb)
7	Read Length (lsb)
8	Read Length
9	Read Length
10	Read Length (msb)
11	Bytes Read (lsb)
12	Bytes Read
13	Bytes Read
14	Bytes Read (msb)
15	Checksum (Sum of bytes 1 to 14)

3.53 Power Rail Voltages (0x78)

This command gets the voltages of the 1.2-V, 1.8-V, 2.5-V and 3.3-V rails. For this command to work correctly these rails must be connected to ADC Inputs A0, B3, B6 and B7 respectively. The command also returns if the system has not been brought out of reset due to one or more of the power rail voltages being out of tolerance. If voltage monitoring is enabled, the Piccolo SW regularly measures the power rail voltages. In this case, the command reads the last measured voltage. When voltage monitoring is disabled, this command triggers ADC conversions to poll the voltage on the four rails.

3.53.1 Read

This command requests a read of the power rail voltages.

Byte	Description
1	Command Byte (F1h)
2	Length (00h)
3	Checksum (F1h)

3.53.2 Read Response

This is the data response for a request to read the power rail voltages.

Byte	Description
1	Response Byte (01h)
2	Length (11h)
3	1.2-V Rail Voltage (lsb)
4	1.2-V Rail Voltage
5	1.2-V Rail Voltage
6	1.2-V Rail Voltage (msb)
7	1.8-V Rail Voltage (lsb)
8	1.8-V Rail Voltage
9	1.8-V Rail Voltage
10	1.8-V Rail Voltage (msb)
11	2.5-V Rail Voltage (lsb)
12	2.5-V Rail Voltage
13	2.5-V Rail Voltage
14	2.5-V Rail Voltage (msb)
15	3.3-V Rail Voltage (lsb)
16	3.3-V Rail Voltage
17	3.3-V Rail Voltage
18	3.3-V Rail Voltage (msb)
19	System Reset state
20	Checksum (sum of bytes 1 to 19)

Note: The Power Rail Voltages are floating point numbers. Please refer to Section 5.3 for the read format for such numbers.

System state ‘0’ indicates normal operation and ‘1’ indicates reset state.

3.54 Voltage Supervision (or Voltage Monitoring) (0x79)

This command is used to enable or disable voltage supervision. For this feature to work properly, it is essential that the 1.2-V, 1.8-V, 2.2-V and 3.3-V power rails are connected to ADC inputs A0, B3, B6 and B7 respectively. Also, GPIO 5 must be connected to the reset pin of the ASIC.

3.54.1 Write

This command is used to set enable/disable voltage supervision.

Byte	Description
1	Command Byte (F2h)
2	Length (01h)
3	Enable/Disable (‘1’/’0’)
4	Checksum (Sum of bytes 1 to 3)

3.54.2 Read

This command requests a read of the enable state of Voltage Supervision.

Byte	Description
1	Command Byte (F3h)
2	Length (00h)
3	Checksum (F3h)

3.54.3 Read Response

This is the data response for a valid request to read the enable state of Voltage Supervision.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	Enable state (‘0’ – Disabled, ‘1’ - Enabled)
4	Checksum (Sum of bytes 1 to 3)

3.55 Toggle Mode (7Ah)

3.55.1 Read

This command parks the DMD, resets the system and then jumps to bootloader. The jump occurs after the command execution and acknowledgment is complete.

Byte	Description
1	Command Byte (F5h)
2	Length (05h)
3	Toggle to bootloader(00h)
3-6	Toggle Mode Write Signature (FF00FF00h)
7	Checksum (Sum of bytes 1 to 6)

3.55.2 Read Response

This is the data response for a valid request to read the toggle mode read signature.

Byte	Description
1	Response Byte (01h)
2	Length (04h)
3-6	Toggle Mode Application Read Signature (12345678h)
7	Checksum (Sum of bytes 1 to 6)

3.56 Piccolo IIC Clock Rate (7Ch)

3.56.1 Write

This command is used to set the Piccolo IIC Master Clock Rate.

Byte	Description
1	Command Byte (F8h)
2	Length (01h)
3	Clock Rate (See below)
4	Checksum (Sum of bytes 1 to 3)

msb	**Byte 3**	*lsb*
Reserved		b0

b0 - IIC Clock Rate
0 – 400 kHz
1 – 100 kHz

Figure 3-14. IIC Clock Rate Byte Format

3.56.2 Read

This command requests a read of the Piccolo IIC Master Clock Rate.

Byte	Description
1	Command Byte (F9h)
2	Length (00h)
3	Checksum (F9h)

3.56.3 Read Response

This is the data response for a valid request to read the Piccolo IIC Master Clock Rate.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	Clock Rate (See Figure 3-14)
4	Checksum (Sum of bytes 1 to 3)

3.57 Piccolo Program Mode (7Eh)

This command requests a read of the Piccolo Program Mode.

Byte	Description
1	Command Byte (FDh)
2	Length (00h)
3	Checksum (FDh)

3.57.1 Read Response

This is the data response for a valid request to read the Piccolo Program Mode.

Byte	Description
1	Response Byte (01h)
2	Length (01h)
3	Program Mode(00h)
4	Checksum (Sum of bytes 1 to 3)

msb	**Byte 3**						*lsb*
b7	b6	b5	b4	b3	b2	b1	b0

B (7:1) Reserved

b (0) - Piccolo Program Mode
0 – Main Application
1 – Bootloader

Figure 3-15. Program Mode Byte Format

Chapter 4
DLPU057–March 2018

Examples

4.1 Examples

This section has some examples of SPI transmissions for some commands to help the reader understand the protocol. The following color codes are used to identify the byte type being transmitted.

A5	Start Character
5A	Escape Character (background changes depending on byte type)
byte	Escaped Data (background changes depending on byte type)
🟨	Command
🟧	Length
🟫	Data
🟦	Checksum
🟥	Response
⬜	Dummy Bytes

Figure 4-1. Example Legend

4.2 Write Backlight 0xFFFF

Checksum = (0x00 + 0x02 + 0xFF + 0xFF) MOD 0x100 = 0x00

Observe that the start character 0xA5 is not included in the checksum computation.

	0	1	2	3	4	5	6	7
Host (Master)	**A5**	**00**	**02**	**FF**	**FF**	**00**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**01**

4.3 Write Backlight 0xA523 (Special Char in Data)

For transmitting Data 0xA523, the 0xA5 is converted to escape sequence 0x5A 0x00 (Bytes 11 and 12). The checksum however uses 0xA5 in its calculation.

Checksum = (0x00 + 0x02 + 0xA5 + 0x23) MOD 0x100 = 0xCA

	8	9	10	11	12	13	14	15	16
Host (Master)	**A5**	**00**	**02**	**5A**	**00**	**23**	**CA**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**01**

4.4 Write Backlight 0xFA5A (Special Char in Data)

The second data byte 0x5A is escape character. To transmit this, send 0x5A twice (bytes 21 and 22). The checksum computation is always based on actual data; and therefore will use only one 0x5A in its computation.

Checksum = (0x00 + 0x02 + 0xFA + 0x5A) MOD 0x100 = 0x56

	17	18	19	20	21	22	23	24	25
Host (Master)	**A5**	**00**	**02**	**FA**	**5A**	**5A**	**56**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**01**

4.5 Write Backlight 0xE96F (Special Char in Checksum)

Special characters can come in any section of the command. This example shows special character in checksum area of a command.

On computing checksum for this command, we get 0x5A as shown below:

Checksum = (0x00 + 0x02 + 0xE9 + 0x6F) MOD 0x100 = 0x5A

To transmit 0x5A as checksum, we need to send two bytes (bytes 31 and 32) as shown below.

	26	27	28	29	30	31	32	33	34
Host (Master)	**A5**	**00**	**02**	**E9**	**6F**	**5A**	**5A**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**01**

4.6 Write Backlight 0x9013 (Special Char in Checksum)

This shows another example where in the computed checksum equals 0xA5, the value of START character.

Checksum = (0x00 + 0x02 + 0x90 + 0x13) MOD 0x100 = 0xA5

We transmit 0x5A 0x00 as bytes 40 and 41 as the checksum.

	35	36	37	38	39	40	41	42	43
Host (Master)	**A5**	**00**	**02**	**90**	**13**	**5A**	**00**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**01**

4.7 Write – Failure – Checksum Mismatch

This example shows that the slave response changes to 0x02 if there is a checksum mismatch.

Checksum = (0x00 + 0x02 + 0xAB + 0xCD) MOD 0x100 = 0x7A.

In this example, the checksum received by slave is 0xEF; hence it returns Checksum Mismatch as its response.

	44	45	46	47	48	49	50	51
Host (Master)	**A5**	**00**	**02**	**AB**	**CD**	**EF**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**02**

4.8 Write – Failure – Invalid Cmd

The following example tries to send an invalid command (0x21) in write mode. The slave responds with 0x03 (Invalid Command).

	52	53	54	55	56	57	58
Host (Master)	**A5**	**42**	**01**	**9F**	**E2**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**03**

4.9 Write – Failure – Command Not Available

This example tries to write to the status register. The command address is valid, but a write operation is invalid. Slave responds with 0x04 (Command Not Available). The same response is also returned by slave if a calibration command is sent while in normal mode or if a normal mode command is sent while in calibration mode. Read of all commands (unless a command supports only write), is available irrespective of normal/calibration mode.

	59	60	61	62	63	64	65	66	67	68
Host (Master)	**A5**	**66**	**04**	**FF**	**FF**	**FF**	**FF**	**66**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**04**

4.10 Write – Failure – Length Mismatch

This example tries to send 4 bytes of data for the backlight command, instead of 2 bytes. The slave responds with 0x05 (Length Mismatch). The backlight command will be ignored and will not be executed.

	69	70	71	72	73	74	75	76	77	78
Host (Master)	**A5**	**00**	**04**	**AB**	**00**	**CD**	**12**	**8E**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**05**

4.11 Write – Other Errors

The slave response only indicates whether the transmitted command was received correctly and if the slave understands the command. Other errors such as data out-of-range occurring during execution phase of a command will be flagged only in the system status. Slave response will not include these errors.

For example, the value 0x02 is invalid for *Enable Calibration Mode* (Section 3.34) command. Though the slave will respond with 0x01 (Command received successfully), during execution, it will flag “Data Out of Range” bit in the System Status. The command execution fails in such a case.

	79	80	81	82	83	84	85
Host (Master)	**A5**	**C8**	**01**	**02**	**CB**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**07**

4.12 Read Backlight

This example reads the backlight value from the slave. This example assumes the currently active backlight value is 0xFA5A. The start and escape characters are not applicable to data sent by slave. In other words, slave will not add a START character 0xA5 before every transaction. Also, any special characters in the byte stream of slave will be transmitted without any change (will not use escape characters).

Also note that, if Length = 0x00, there are no data bytes. The byte after the length byte will be treated as the checksum byte.

The read checksum includes the response byte.

Read Checksum = (0x01 + 0x02 + 0x5A + 0xFA) MOD 0x100 = 0x57

The example is split into two lines for the sake of clarity.

	86	87	88	89	90	91
Host (Master)	**A5**	**01**	**00**	**01**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**

	92	93	94	95	96
Host (Master)	**00**	**00**	**00**	**00**	**00**
Piccolo (Slave)	**01**	**02**	**5A**	**FA**	**57**

4.13 Read ASIC Register

To read ASIC register, the master has to transmit the address of the ASIC register. This is one command where the transmit bytes are non-zero for a read command. The below figure shows reading of the Command Table Index register (0xC5) from the ASIC and the slave response assuming the current command table index value is 0x08.

Write Checksum = (0x69 + 0x01 + 0xC5) MOD 0x100 = 0x2F

Read Checksum = (0x01 + 0x04 + 0x08 + 0x00 + 0x00 + 0x00) MOD 0x100 = 0xD

	97	98	99	100	101	102	103
Host (Master)	**A5**	**69**	**01**	**C5**	**2F**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**

	104	105	106	107	108	109	110
Host (Master)	**00**	**00**	**00**	**00**	**00**	**00**	**00**
Piccolo (Slave)	**01**	**04**	**08**	**00**	**00**	**00**	**0D**

4.14 Read – Failure

A read failure can occur because of two reasons:

- Transmission errors like checksum mismatch, length mismatch, invalid command address etc
- Execution Failure – Data out-of-range, I2C failure etc

A read command will try to execute first before sending the response. Hence the slave may take more time to send the response for a read command. The response byte will indicate the failure status on encountering any error. In case of failure, only the response byte is transmitted by the slave. There is no read checksum or read bytes in such cases.

If the master tries to read additional bytes, the slave will respond with dummy bytes (0xFF). The slave will ignore all such bytes transmitted by master (bytes 119 to 121 in below figure) till it receives the next start of a command character (0xA5). If the slave ignores any bytes, it will be flagged in the system status bit “SPI Bytes Ignored”.

This example shows a case where the read command fails because of Length Mismatch in the transmitted data. The data length should be “00”. The slave response byte will indicate 0x05 (Length Mismatch).

	111	112	113	114	115	116	117
Host (Master)	**A5**	**01**	**02**	**FF**	**FF**	**00**	**00**
Piccolo (Slave)	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**

	118	119	120	121
Host (Master)	**00**	**00**	**00**	**00**
Piccolo (Slave)	**05**	**FF**	**FF**	**FF**

Writing and Reading a Floating Point Number

5.1 Writing and Reading a Floating Point Number

Every float value is sent and received as a set of four integers. This is done by using an integer pointer to access these values and splitting them as though they are integers. To get the actual float values, the 4 integers are concatenated into a 32-bit integer and a float pointer is used to access them. The value of the float type pointer gives the actual value of the low pass filter constants.

5.2 Example: Writing Floating Point Data (say 1.0):

Step 1: Use an integer pointer to access the floating point value of ‘Data’ (which contains 1.0)

Uint32* DataInt = (Uint32*) &Data;

Step 2: Split the value stored in the integer pointer into bytes.

The value of the integer pointer (*DataInt) is 0x3E800000.

Hence the 4 bytes are 0x3E, 0x80, 0x00 and 0x00 respectively

Step 3: Send each byte (LSB first) in the SPI command

Byte 1 = 0x00

Byte 2 = 0x00

Byte 3 = 0x80

Byte 4 = 0x3E

5.3 Example: Reading Floating Point Data:

Let us assume the received values of bytes be 0x00, 0x00, 0x80, 0x3E respectively.

Step 1: Concatenating them in the correct order we get 0x3E800000

Step 2: Store 0x3E800000 into a 32 bit integer (say Uint32 DataInt).

Step 3: Use a 32 bit float pointer to access the integer.

float32* DataFlt= (float32*) & DataInt

Step 4: The value of the float pointer contains the actual value of Data.

float32 Data = * DataFlt;

We can see that the value of Data is 1.0,

Note: The above examples are just a description of the procedure. Different processors may store the float values in different ways.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265