Lepton™ Software Interface Description Document (IDD)

August 28, 2014

Document Number: 110-0144-03

Version 0.3.53

Table of Contents

List of Tables

List of Figures

1 Document Description

1.1 Revision History

Rev. #	Date	Comments
031	27 February 2014	Initial release associated with Camera Version 0.3.1
035	22 April 2014	Update for Camera Version 0.3.35
0350	1 August 2014	Updated for version 0.3.50
0353	28 August 2014	Updated for release 0.3.53

1.2 Scope

This interface description document (IDD) defines software interface requirements and software commands available to a Host.

1.3 References

The following documents form a part of this specification to the extent specified herein.

1.3.1 FLIR Systems Documents

102-PS245-99	Lepton Datasheet

1.3.2 External Documents

UM10204	I2C-Bus Specification and User Manual

1.3.3 Acronyms / Abbreviations

AGC	Automatic Gain Control
BIT	Built -In Test
CCI	Command and Control Interface
CMD	Command
CRC	Cyclic Redundancy Check
FFC	Flat Field Correction
FPA	Focal Plane Array
I2C	Inter-Integrated Circuit – a multi-master serial single-ended computer bus invented by Philips
LSB	Least Significant Byte
LUT	Look-Up Table
MSB	Most Significant Byte
ROI	Region of Interest
RX	Receive
SN	Serial Number
SPI	Serial Peripheral Interface
SW	Software
TBD	To Be Determined
TWI	Two-Wire Interface supporting I2C
TX	Transmit

2 Communications Protocol

Lepton supports Host command and control over a Two-Wire Interface (CCI/TWI). The SDK provides layering to isolate the operations and Lepton protocol from the Data link physical transport.

2.1 CCI/TWI Register Protocol

The Lepton camera module supports a command and control interface (CCI) hosted on a Two-Wire Interface (TWI) similar to I2C. The interface consists of a small number of registers through which a Host issues commands to, and retrieves responses from the Lepton camera module. See Figure 1.

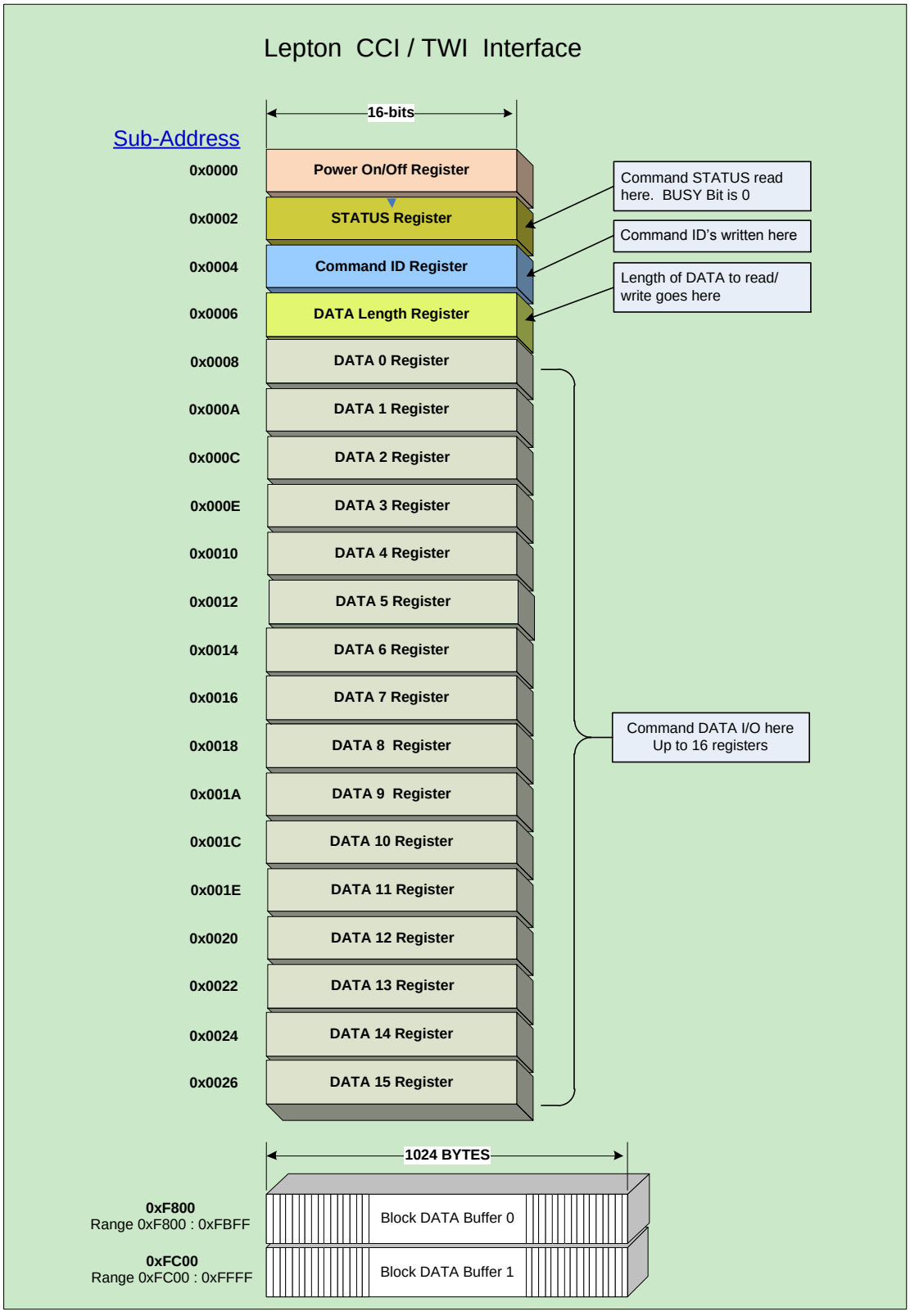

Figure 1 Lepton CCI/TWI Registers

Typical transmission requires the sequence of:

1. Polling the status register until camera is ready for a new command (Busy bit clear).
2. Writing data to send to the camera if required into the DATA Registers or block Data buffer.
3. Writing the number of data words written (16-bit data words) to the Data Length Register.
4. Writing the desired command ID to the Command Register.
5. Polling the Status Register to determine when the command is completed (busy bit cleared).
6. Read the success code from the status register.
7. Retrieve any responses as required from the Data registers or block Data buffer.

There are three basic operations capable of being commanded via the CCI. The first is a “get” or read of data, the second is a “set” or write of data and the third is a “run” or execution of a routine. A typical get sequence is illustrated in Figure 2, a typical set in Figure 3, and a typical run in Figure 4.

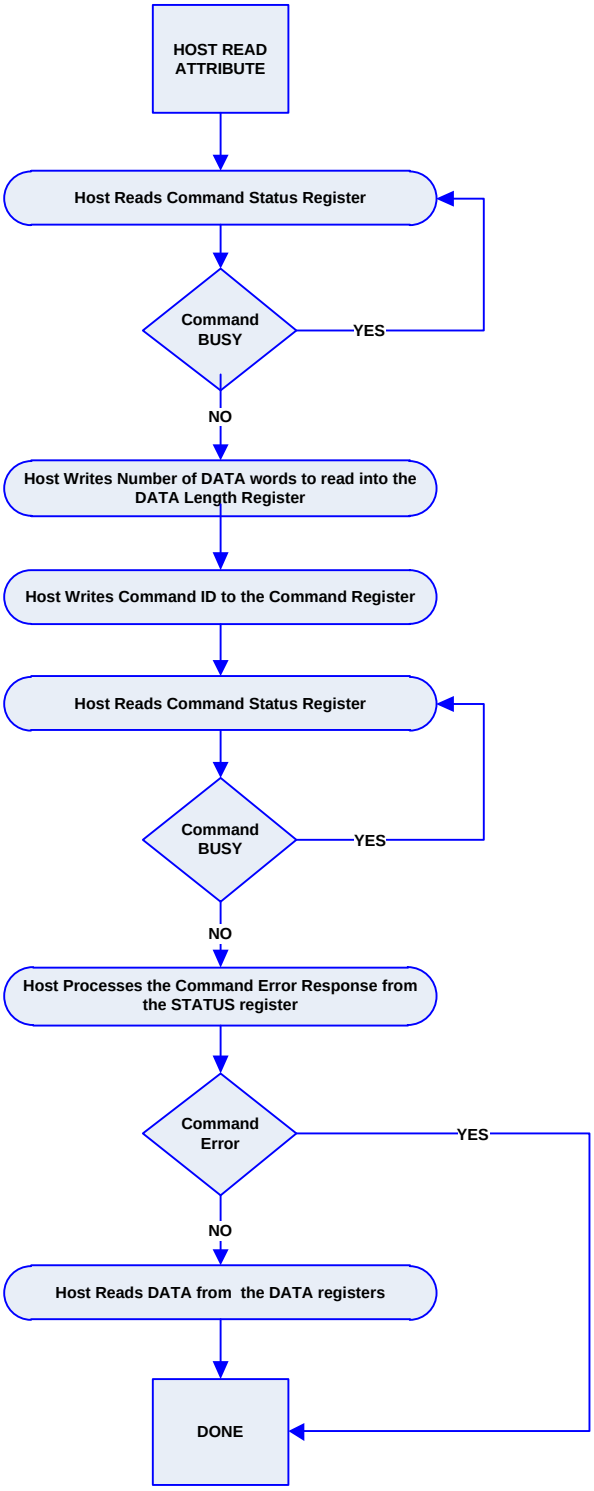

Figure 2 Lepton CCI/TWI Get or Read Attribute Sequence

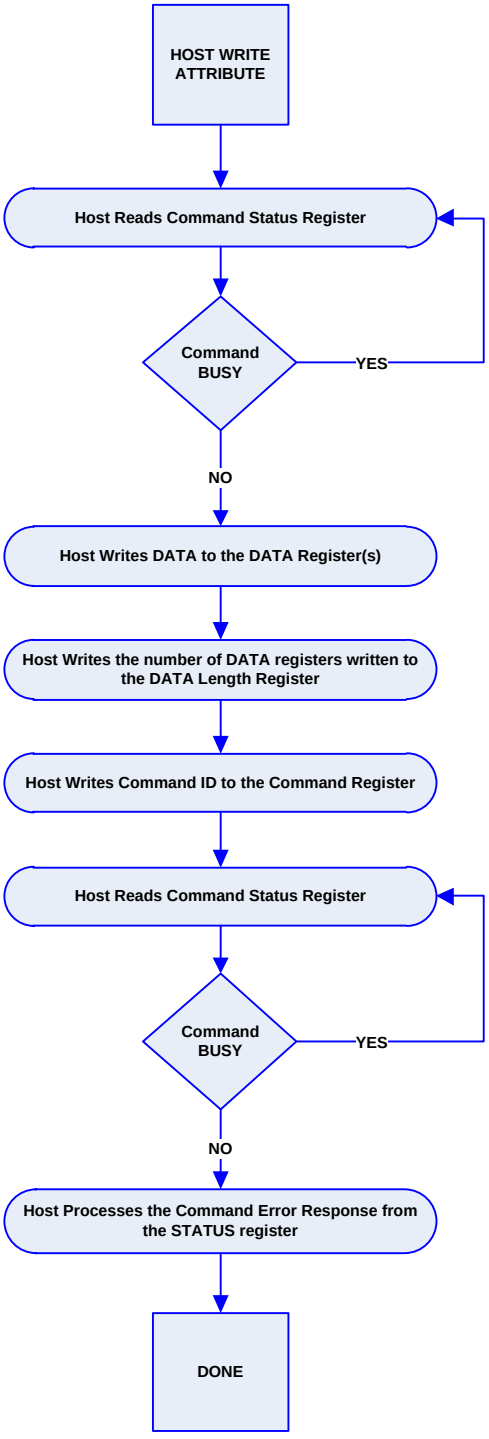

Figure 3 Lepton CCI/TWI Set or Write Sequence

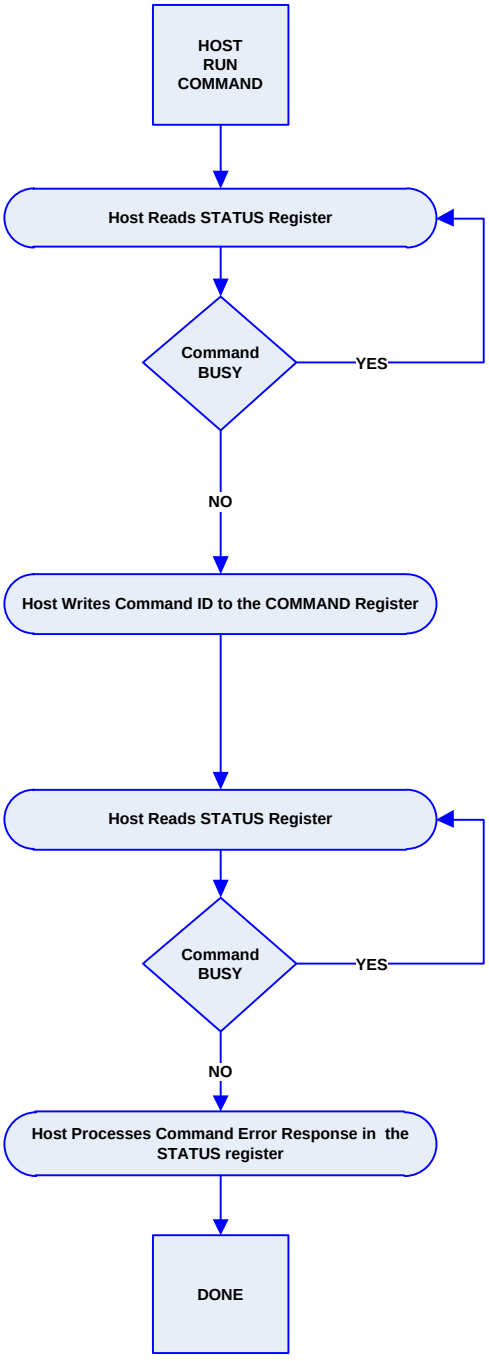

Figure 4 Lepton CCI/TWI Run Command Sequence

2.1.1 CCI/TWI Interface

The CCI/TWI interface is similar to the I2C standard; however Lepton registers are all 16-bits wide and consequently only 16-bit transfers are allowed. This is illustrated in Figure 5. Device parameters are listed in Table 1.

Table 1 CCI/TWI Device Parameters

Device ID	0x2A (7-bit addressing)
Transfer DATA Bit Width	16-bits
Clock	100Kbaud, 400Kbaud & 1M baud.

2.1.1.1 Reading from the Camera

Reading DATA from the Camera using the CCI/TWI interfaces follows the I2C standard except the DATA is all 16-bit wide. All Camera CCI/TWI Registers are 16-bits wide, and the larger DATA buffer is organized as 512 x 16-bits. The Camera's DATA Length Register always specifies lengths as a number of 16-bit DATA being transferred. The Camera supports access to random locations in which the transmission includes the starting Register address in the transmission, access to the current address, and address auto-increment. Figure 5 illustrates typical CCI/TWI Register Read access transmission. The Camera accepts the Repeated START condition to combine specifying the register address with register access in a single transmission. Alternatively, one can separate a write transmission to set the current address, then issue READ transmissions that start at this current address. A read is stopped by sending a Not Acknowledge signal followed by a Stop sequence. A sequential read can be stopped after reading the last byte by sending a Not Acknowledge signal followed by a Stop sequence

Single READ from random location – 16-Bit words

Figure 5 CCI/TWI Single READ from random location reads 16-bit DATA

Set Base Address Register current location to random location

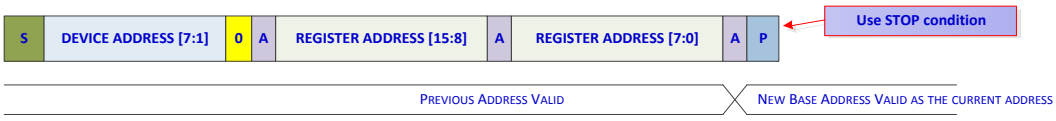

Figure 6 CCI/TWI Setting the Camera's CCI/TWI current address

Sequential READ from current location – byte at a time, 16-Bit words

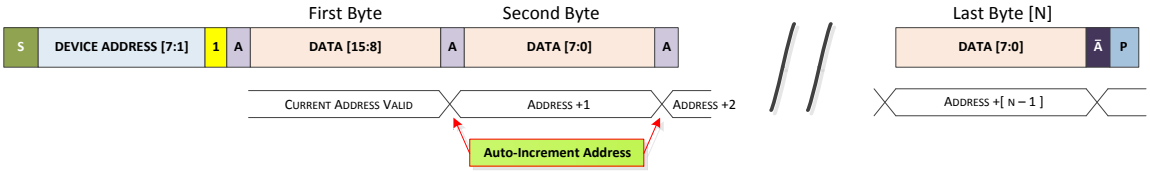

Figure 7 CCI/TWI Reading sequentially from the Camera's CCI/TWI current address

2.1.1.2 Writing to the Camera

Writing DATA to the Camera using CCI/TWI interfaces follows the I2C standard except the DATA are all 16-bits wide. All Camera CCI/TWI Registers are 16-bits wide, and a larger DATA buffer is organized as 512 x 16-bits. The Camera's DATA Length Register always specifies lengths as the number of 16-bit DATA words being transferred.

The Camera supports access to a random 16-bit aligned location in which the starting register address is specified in the transmission with post-access address auto-increment for sequential reads or writes. The Camera also supports access to the current address with post-access address auto-increment. Typical register writes are illustrated in Figure 8, and sequential writes are illustrated in Figure 9.

Single WRITE to random location – 16-bit words

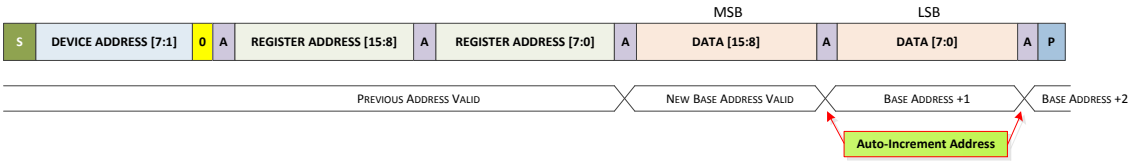

Figure 8 CCI/TWI Single WRITE to random location writes 16-bit DATA

Sequential WRITE to random location – byte at a time, 16-Bit words

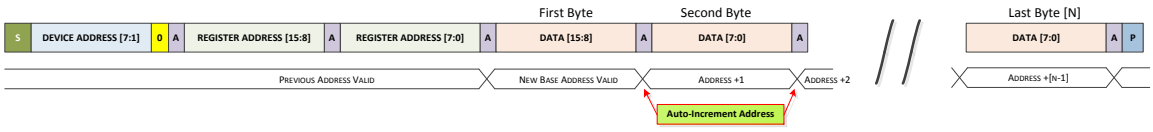

Figure 9 CCI/TWI Writing sequentially

2.1.2 CCI/TWI Command Register

The Lepton Command Register is a 16-bit register located at Register Address 0x0004. This register is used to issue a command to the Lepton Camera. Writing a value to this register initiates the camera's command processing. It is important to make sure the Command BUSY bit in the Lepton Status Register (Register Address 0x0002) indicates that the Camera is ready to accept a new command (BUSY bit cleared) before initiating a new command; otherwise the Camera communication may become compromised, necessitating a restart or reboot of the Camera.

The Command Register Word register bit definitions are illustrated in Figure 10. The Command Register Word is composed of 4 fields, each described in more detail in the sections that follow:

1. A Module ID designating which camera subsystem to access (see Table 2)
2. A Command ID that specifies a unique element or command base, for that subsystem.
3. A command type designating the command is one of Get or Set data type or Run type (see Table 3).

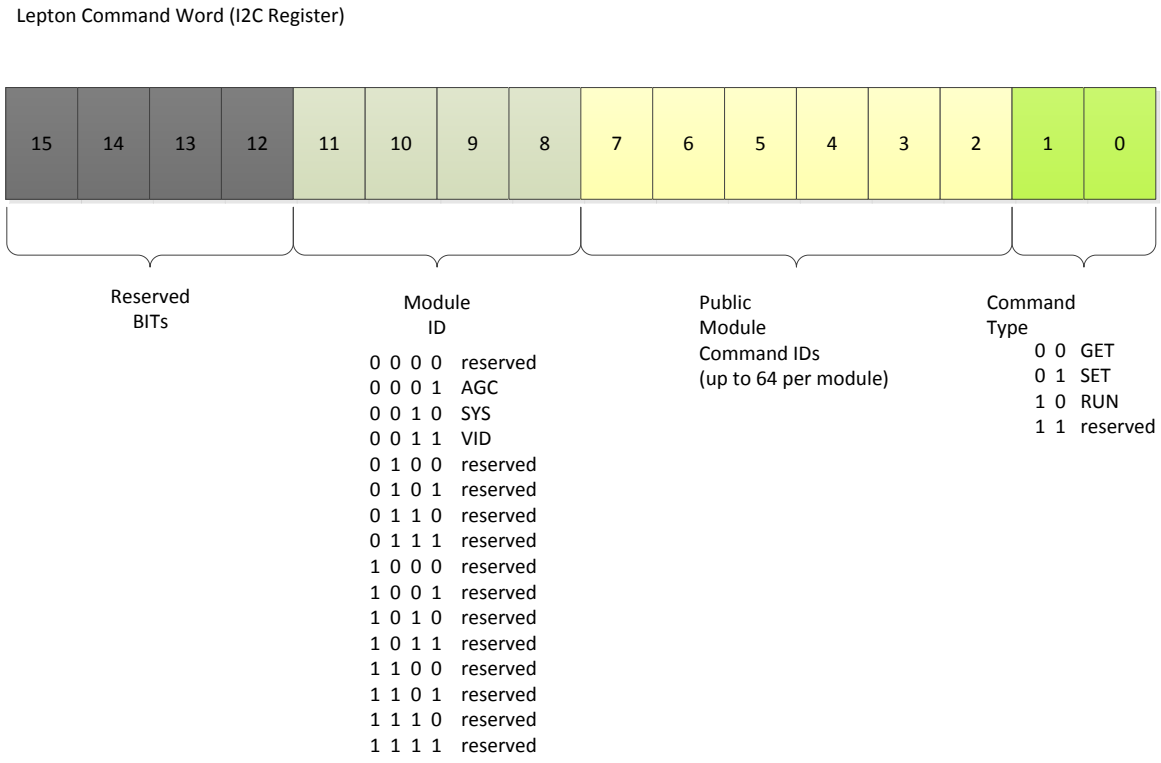

Figure 10 Lepton Command Word Format

2.1.2.1 Module ID

The Lepton Camera Module ID designates which camera module to address. The Camera modules encapsulate properties or attributes and methods of a camera sub-system. Currently, Lepton defines 3 sub-systems and the SDK exposes their associated module as follows:

- AGC – Automatic Gain Control, affects image contrast and quality
- SYS – System information

- VID – Video processing control

The Module IDs and their location in the Lepton command word are illustrated in Figure 10.

2.1.2.2 Command ID

For each of the Lepton Camera modules, a unique Lepton Command ID identifies an element of the module, either an attribute or property, or an action. Each Camera module exposes up to 64 Command IDs assigned to attributes and/or methods of that module.

2.1.2.3 Command Type

A command type specifies what the command does.

- 0x00 Get a module property or attribute value
- 0x01 Set a module property or attribute value
- 0x02 Run – execute a camera operation exposed by that module

2.1.3 CCI/TWI Status Register

The Status register, located at Register Address 0x0002 and illustrated in Figure 11, is used to communicate command status and camera boot status. Whenever a Host issues a command to the Camera by writing to the Command Register, the Camera automatically asserts (sets to 1) the command BUSY bit (Bit 0) in the Status register. When the command is completed, the response code is written into the upper 8-bits of the Status register (Bits 15-8). Then the Camera de-asserts (sets to 0) the BUSY bit to signal the Host the command is complete. See Figure 12 for the possible responses from the Camera to a command.

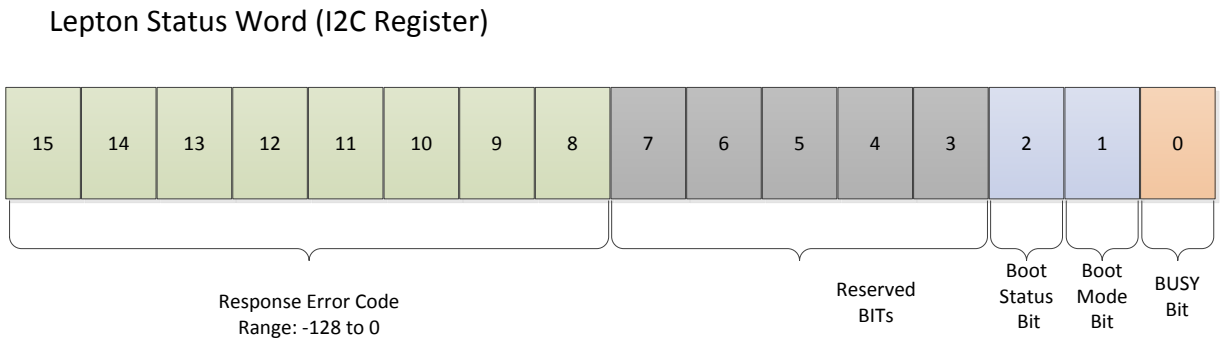

Figure 11 CCI/TWI Status Register Definition

2.1.3.1 Boot Status Bit (Bit 2)

If the Camera successfully boots up, this bit is set to 1. If this bit is 0, then the Camera has not booted. A host can monitor this bit to learn when the Camera has booted.

2.1.3.2 Boot Mode Bit (Bit 1)

For normal operation, this bit will be set to 1, indicating successful boot from internal ROM.

2.1.4 CCI/TWI Data Length Register

The DATA Length register, located at Register Address 0x0006, is used to specify the number of 16-bit words being transferred (or number of 16-bit DATA registers used in the transfer). For example, if a command is to transfer a 32-bit value to the Camera, the Host would set the Data Length register to 2 (two 16-bit registers used to transfer 32-bits).

2.1.5 CCI/TWI Data Registers

The DATA registers [0-15] , located at Register Addresses 0x0008 thru 0x0026, are used to transfer Data to and from the Camera. Each register is 16-bits wide and there are 16 independent registers. Auto-increment mode is used whenever reading and writing these registers. Thus if the first register (DATA 0) is specified, consecutive reads or writes are made to the next DATA register automatically using I2C multi-byte transfer mechanisms.

2.1.6 CCI/TWI Byte Order

Since the CCI/TWI interface transfers DATA in 16-bit words, byte order becomes important. The Lepton CCI/TWI interface only supports MSB first (Big Endian). Within each 16-bit word, bits 7-0 contain the MSB and bits 15:8 contain the LSB.

2.1.6.1 Multi-Word Transfers

When transmitting DATA that are larger than a single word (16-bits), the larger DATA is divided into multiple 16-bit words; each word is then placed into multiple DATA registers with the least significant word in the lower DATA register. Thus for a 32-bit transfer, a Host would place the lower 16-bits into DATA 0 (Least Significant Word first) and the upper 16-bits into DATA 1.

2.1.6.2 CCI/TWI Data Block Buffer

For transfers that exceed the 16 DATA registers, the camera provides a 1k Byte buffer. It is used for transferring larger blocks of DATA such as user-defined color look-up tables. These buffers are also addressed as 16-bit words, so the total length of a single buffer is 512 words. Access is treated as a multi-word transfer as well with the least significant words in the lower memory addresses. Auto-increment access is also supported.

2.2 CRC Handling

2.2.1 Message CRC Bytes

On all incoming and outgoing messages, a cyclical redundancy check (CRC) is calculated using CRC-CCITT-16 initialized to **0**. Polynomial = $x^{16} + x^{12} + x^5 + 1$ or 0x11021. The CRC is calculated using all previous bytes in the packet (i.e. bytes 0 through N).

Below is an example showing a CRC calculation for the single byte 0x6E.

Data = 0x6E = 01101110 (binary); Polynomial = 10001000000100001 (binary)

01101110000000000000000 [data is right-padded with 16 zeros]
10001000000100001

01101110000000000000000
010001000000100001

00101010000100001000000
0010001000000100001

00001000000110001100000
0001000100000100001

00001000000110001100000
0000100010000100001

00000000100011010110100
0000010001000000100001

00000000100011010110100
0000001000100000100001

00000000100011010110100
0000000100010000100001

00000000100011010110100 = **0x8D68**

2.3 Lepton SDK Error Codes

All Lepton SDK functions will return an error code. If the function is successful, the response is LEP_OK. Otherwise the return code will be one from the ***enum*** listed below in Figure 12.

```
/*
 * Represents the different result codes the camera can return.
 */
typedef enum Result
{
   FLR_OK                              = 0,     /*!< Camera ok */
   FLR_COMM_OK                         = FLR_OK, /*!< Camera comm ok (same as FLR_OK) */

   FLR_ERROR                           = -1,    /*!< Camera general error */
   FLR_NOT_READY                       = -2,    /*!< Camera not ready error */
   FLR_RANGE_ERROR                     = -3,    /*!< Camera range error */
   FLR_CHECKSUM_ERROR                  = -4,    /*!< Camera checksum error */
   FLR_BAD_ARG_POINTER_ERROR           = -5,    /*!< Camera Bad argument error */
   FLR_DATA_SIZE_ERROR                 = -6,    /*!< Camera byte count error */
   FLR_UNDEFINED_FUNCTION_ERROR        = -7,    /*!< Camera undefined function error */
   FLR_FUNCTION_NOT_SUPPORTED_ERROR    = -8,    /*!< Camera function not supported error */
   FLR_DATA_OUT_OF_RANGE_ERROR         = -9,    /*!< Camera Data out of range error */

   /* OTP access errors */
   FLR_OTP_WRITE_ERROR                 = -15,   /*!< Camera OTP write error */
   FLR_OTP_SEC_READ_ERROR              = -16,   /*!< single bit error detected (correctable) */
   FLR_OTP_DED_READ_ERROR              = -17,   /*!< single bit error detected (correctable) */

   FLR_OTP_NOT_PROGRAMMED_ERROR        = -18,   /*!< Flag read as non-zero */

   /* Operation Errors */
   FLR_DIV_ZERO_ERROR                  = -80,   /*!< Attempted div by zero */

   /* Communication Errors */
   FLR_COMM_PORT_NOT_OPEN              = -101,  /*!< Comm port not open */
   FLR_COMM_RANGE_ERROR                = -102,  /*!< Comm port range error */
   FLR_ERROR_CREATING_COMM             = -103,  /*!< Error creating comm */
   FLR_ERROR_STARTING_COMM             = -104,  /*!< Error starting comm */
   FLR_ERROR_CLOSING_COMM              = -105,  /*!< Error closing comm */
   FLR_COMM_CHECKSUM_ERROR             = -106,  /*!< Comm checksum error */
   FLR_COMM_NO_DEV                     = -107,  /*!< No comm device */
   FLR_COMM_TIMEOUT_ERROR              = -108,  /*!< Comm timeout error */
   FLR_COMM_ERROR_WRITING_COMM         = -109,  /*!< Error writing comm */
   FLR_COMM_ERROR_READING_COMM         = -110,  /*!< Error reading comm */
   FLR_COMM_COUNT_ERROR                = -111,  /*!< Comm byte count error */

   /* Other Errors */
   FLR_OPERATION_CANCELED              = -126,  /*!< Camera operation canceled */
   FLR_UNDEFINED_ERROR_CODE            = -127,  /*!< Undefined error */

   FLR_END_ERROR_CODES

} FLR_RESULT;
```

Figure 12 Lepton SDK Response Error Codes

3 Startup and Port Configuration

Using the Lepton SDK to communicate with the Lepton Camera requires opening a supported communication port before issuing any other calls. The port open operation specifies the desired baud rate for the port and returns a port descriptor for use with all other SDK APIs. A host needs to open a port for every port-camera connection they are supporting. Typically this is only once, but the SDK does not impose any limitations.

The port open operation also identifies the Device ID automatically freeing the Host application from needing to specify the Device ID. The port descriptor returns the selected Device ID.

C-SDK Commands	Description
`LEP_OpenPort()`	Opens a communications port if available. Supported Lepton communication ports are TWI and SPI. Only TWI is supported in the current release. (SPI support is planned for a later release.)

C SDK Interface:

```
LEP_RESULT LEP_OpenPort(LEP_UINT16          portID,
                        LEP_CAMERA_PORT_E   portType,
                        LEP_UINT16          portBaudRate,
                        LEP_CAMERA_PORT_DESC_T_PTR portDescPtr )


portID -   User defined value to identify a specific comm port.
           Useful when multiple cameras are attached to a single Host.

portBaudRate – Port-specific Units: kHz.   Supported TWI: 400
                                           Supported SPI: 20000 max (20 MHz)

/* Lepton physical transport interfaces
*/
typedef enum LEP_CAMERA_PORT_E_TAG
{
    LEP_CCI_TWI=0,
    LEP_CCI_SPI,
    LEP_END_CCI_PORTS
}LEP_CAMERA_PORT_E, *LEP_CAMERA_PORT_E_PTR;

/* Communications Port Descriptor Type
*/
typedef struct  LEP_CAMERA_PORT_DESC_T_TAG
{
    LEP_UINT16            portID;
    LEP_CAMERA_PORT_E     portType;
    LEP_UINT16            portBaudRate;

}LEP_CAMERA_PORT_DESC_T, *LEP_CAMERA_PORT_DESC_T_PTR;
```

3.1 Port Selection

The Lepton SDK provides a mechanism to communicate with specific communication ports. Communication ports are uniquely identified by the port descriptor returned from a successful port open operation. Each Lepton SDK function requires a valid port descriptor as a parameter to identify which port to issue the command to. Typically only one port is opened and this port descriptor is passed with each Lepton SDK call. It is readily possible to route commands to different cameras using each camera's unique port descriptor. This routing is performed in the device driver, not in the SDK.

4 SDK Camera Modules

The Lepton SDK partitions the software interfaces into independent sub-systems or modules. A module is a collection of interfaces supporting common camera elements, for example the AGC module presents interfaces that affect the video output contrast and brightness processing. Each module is identified by a unique ID; see Table 2. The individual interfaces within each module are also uniquely identified using a command ID Base. Modules present interfaces to retrieve (Get) or modify (Set) attributes or properties of that module. Some modules also provide operations or methods as well, these are called run commands. See Table 3.

Table 2 Lepton SDK Modules

Modules		
ID	**Name**	**Description**
0x100	AGC	Automatic Gain Control for image Brightness and Contrast
0x200	SYS	System Information
0x300	VID	Video Control

Table 3 Command Types

	Command Types			
	Get	**Set**	**Run**	**Invalid**
Type Value to Add to the Command ID Base	0x0	0x1	0x2	0x3

4.1 Data Types

Data types used in the Lepton SDK are defined in the file `LEPTON_Types.h`. Data widths are specified in the data type used, for example: LEP_UINT16 specifies an unsigned integer with a data width of 16-bits.

Enumeration bit-width is typically compiler-dependent; however in the Lepton SDK, the width of 32-bits is used and the value is a signed integer, thus the equivalent is a signed 32-bit integer. For all Lepton SDK functions that pass enumerations, the data size is two 16-bit words per enumeration.

When issuing commands to the Lepton camera, the data transmitted uses 16-bit registers making the data size granularity 16-bits. As such, when specifying the data size, it is always interpreted as the number of 16-bit words to transmit.

4.2 Command Format

As described in 2.1.2, Lepton commands are contained in a single 16-bit command word. This 16-bit command word consists of 4 fields:

1. A Module ID designating which camera subsystem to access (see Table 2),
2. A Command ID that specifies a unique element or command base, for that subsystem
3. A command type designating the command is one of Get or Get data type or an execution or run type (see Table 3).

4.3 Command Word Generation Example

To specify to the Camera which action to take, the Module ID is added with the Command ID base and with the Command Type and if required, a protection bit value, to synthesize the Command Word.

4.3.1 AGC, VID, and SYS Module Command ID Generation

AGC, VID, and SYS modules no not require a protection bit to be set before the camera will recognize it as a valid command so the protection bit value is 0x0000. For example, the AGC Module ID is 0x0100; the ACG Enable command ID Base is 0x00. To retrieve the current AGC enable state, issue a Get command specifying command type of 0x0. The AGC module protection bit not defined so the value is 0x0000. The Command ID is synthesized as follows: Module ID + Command ID Base + Type + Protection Bit value= Command ID. So in this example, 0x0100 + 0x00 + 0x0 + 0x0000 = 0x0100 and this is the Get AGC Enable State Command ID. To set the AGC enable state to enabled, the command type is 0x1 and thus the Command ID is 0x100 + 0x00 + 0x1 + 0x0000 = 0x0101.

4.4 SDK Module: AGC 0x100

This module provides command and control of the video output Automatic Gain Control (AGC) operation. The camera's video data may be processed to provide an optimum scene contrast using one of two policies: HEQ-Histogram Equalization, or by Linear Histogram stretching. This module provides commands to enable, select, and control the AGC processing.

4.4.1 AGC Enable and Disable

To turn AGC ON is to enable AGC processing. Disabling the AGC will turn the AGC processing OFF and the video data will not be optimized for scene contrast. This command sets and retrieves the AGC state.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_AGC_DISABLE	LEP_AGC_ENABLE	LEP_AGC_DISABLE	N/A	N/A

SDK Module ID: AGC **0x0100**

SDK Command ID: Base **0x00**
With Get **0x00**
With Set **0x01**

SDK Data Length: Get **2** size on an **enum** data type on a 32-bit machine
Set **2** size on an **enum** data type on a 32-bit machine

C-SDK Commands	Description
LEP_GetAgcEnableState()	Updates **agcEnableStatePtr** with the Camera's current AGC enable state.
LEP_SetAgcEnableState()	Sets Camera's current AGC enable state to **agcEnableState**

C SDK Interface:

```
LEP_RESULT LEP_GetAgcEnableState(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                 LEP_AGC_ENABLE_E_PTR agcEnableStatePtr)

LEP_RESULT LEP_SetAgcEnableState(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                 LEP_AGC_ENABLE_E agcEnableState)

/* AGC Enable Enum
*/
typedef enum LEP_AGC_ENABLE_TAG
{
    LEP_AGC_DISABLE=0,
    LEP_AGC_ENABLE,
    LEP_END_AGC_ENABLE

}LEP_AGC_ENABLE_E, *LEP_AGC_ENABLE_E_PTR;
```

4.4.2 AGC ROI Select

The AGC algorithms utilize a histogram, which is collected from within a specified rectangular window or Region Of Interest (ROI). This region is defined by 4 parameters: start column, start row, end column, and end row. The region is adjustable from full window to a sub-window.

Dimension	Minimum Value	Maximum Value	Default Value	Units	Scale factor
start column	0	< = endCol	0	pixels	1
start row	0	< endRow	0	pixels	1
end column	>= startCol	79	79	pixels	1
end row	>= startRow	59	59	pixels	1

SDK Module ID: AGC **0x0100**

SDK Command ID: Base **0x08**
With Get **0x08**
With Set **0x09**

SDK Data Length: Get **4** size of `LEP_AGC_ROI_T` data type
Set **4** size of `LEP_AGC_ROI_T` data type

C-SDK Commands	Description
`LEP_GetAgcROI()`	Updates **`agcROIPtr`** with the Camera's current AGC ROI
`LEP_SetAgcROI()`	Sets Camera's current AGC ROI to **`agcROI`**

C SDK Interface:

```
LEP_RESULT LEP_GetAgcROI(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                         LEP_AGC_ROI_T_PTR agcROIPtr)

LEP_RESULT LEP_SetAgcROI(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                         LEP_AGC_ROI_T agcROI)

/* AGC ROI Structure
*/
typedef struct LEP_AGC_ROI_TAG
{
    LEP_UINT16 startCol;
    LEP_UINT16 startRow;
    LEP_UINT16 endCol;
    LEP_UINT16 endRow;

}LEP_AGC_ROI_T, *LEP_AGC_ROI_T_PTR;
```

4.4.3 AGC Histogram Statistics

The AGC algorithms use the image histogram as input. This attribute returns the current Histogram statistics of minimum intensity, maximum intensity, mean intensity, and the number of pixels processed within the defined AGC ROI. This command is Read-only.

Dimension	Minimum Value	Maximum Value	Units	Scale factor
minimum intensity	0	2^14 -1	pixels	1
maximum intensity	0	2^14 -1	pixels	1
mean intensity	0	2^14 -1	pixels	1
number of pixels	0	4800	pixels	1

SDK Module ID: AGC **0x0100**

SDK Command ID: Base **0x0C**
With Get **0x0C**

SDK Data Length: Get **4** size of `LEP_AGC_HISTOGRAM_STATISTICS_T` data type

C-SDK Commands	Description
`LEP_GetAgcHistogramStatistics()`	Updates **`agcHistogramStatisticsPtr`** with the Camera's current AGC Histogram statistics

C SDK Interface:

```
LEP_RESULT LEP_GetAgcHistogramStatistics(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                         LEP_AGC_HISTOGRAM_STATISTICS_T_PTR
                                             *agcHistogramStatisticsPtr)


/* AGC Histogram Statistics Structure
*/
typedef struct LEP_AGC_HISTOGRAM_STATISTICS_TAG
{
    LEP_UINT16  minIntensity;
    LEP_UINT16  maxIntensity;
    LEP_UINT16  meanIntensity;
    LEP_UINT16  numPixels;

}LEP_AGC_HISTOGRAM_STATISTICS_T, *LEP_AGC_HISTOGRAM_STATISTICS_T_PTR;
```

4.4.4 AGC HEQ Dampening Factor

This parameter is the amount of temporal dampening applied to the HEQ transformation function. An IIR filter of the form (*N*/256) * **previous** + ((256-*N*)/256) * **current** is applied , and the HEQ dampening factor represents the value N in the equation, i.e., a value that applies to the amount of influence the previous HEQ transformation function has on the current function. . The lower the value of N the higher the influence of the current video frame whereas the higher the value of N the more influence the previous damped transfer function has.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	256	64	N/A	1

SDK Module ID: AGC **0x0100**

SDK Command ID: Base **0x24**
With Get **0x24**
With Set **0x25**

SDK Data Length: Get **1** size of `LEP_UINT16` data type
Set **1** size of `LEP_UINT16` data type

C-SDK Commands	Description
`LEP_GetAgcHeqDampingFactor()`	Updates **`agcHeqDampingFactorPtr`** with the Camera's current HEQ dampening factor
`LEP_SetAgcHeqDampingFactor()`	Sets Camera's current HEQ dampening factor to **`agcHeqDampingFactor`**

C SDK Interface:

```
LEP_RESULT LEP_GetAgcHeqDampingFactor(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                      LEP_UINT16 *agcHeqDampingFactorPtr)

LEP_RESULT LEP_SetAgcHeqDampingFactor(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                      LEP_UINT16 gcHeqDampingFactor)
```

4.4.5 AGC HEQ Clip Limit High

This parameter defines the maximum number of pixels allowed to accumulate in any given histogram bin. Any additional pixels in a given bin are clipped. The effect of this parameter is to limit the influence of highly-populated bins on the resulting HEQ transformation function.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	4800	4800	pixels	1

SDK Module ID: AGC **0x0100**

SDK Command ID: Base **0x2C**
With Get **0x2C**
With Set **0x2D**

SDK Data Length: Get **1** size of `LEP_UINT16` data type
Set **1** size of `LEP_UINT16` data type

C-SDK Commands	Description
`LEP_GetAgcHeqClipLimitHigh()`	Updates **`agcHeqClipLimitHighPtr`** with the Camera's current HEQ level high value
`LEP_SetAgcHeqClipLimitHigh()`	Sets Camera's current HEQ level high value to **`agcHeqClipLimitHigh`**

C SDK Interface:

```
LEP_RESULT LEP_GetAgcHeqClipLimitHigh(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                      LEP_UINT16 *agcHeqClipLimitHighPtr)

LEP_RESULT LEP_SetAgcHeqClipLimitHigh(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                      LEP_UINT16 agcHeqClipLimitHigh)
```

4.4.6 AGC HEQ Clip Limit Low

This parameter defines an artificial population that is added to every non-empty histogram bin. In other words, if the Clip Limit Low is set to *L*, a bin with an actual population of *X* will have an effective population of *L* + *X*. y empty bin that is nearby a populated bin will be given an artificial population of *L*. The effect of higher values is to provide a more linear transfer function; lower values provide a more non-linear (equalized) transfer function.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	1024	512	pixels	1

SDK Module ID: AGC **0x0100**

SDK Command ID: Base **0x30**
With Get **0x30**
With Set **0x31**

SDK Data Length: Get **1** size of `LEP_UINT16` data type
Set **1** size of `LEP_UINT16` data type

C-SDK Commands	Description
`LEP_GetAgcHeqClipLimitLow()`	Updates **`agcHeqClipLimitLowPtr`** with the Camera's current HEQ level Low value
`LEP_SetAgcHeqClipLimitLow()`	Sets Camera's current HEQ level Low value to **`agcHeqClipLimitLow`**

C SDK Interface:

```
LEP_RESULT LEP_GetAgcHeqClipLimitLow(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                     LEP_UINT16 *agcHeqClipLimitLowPtr)

LEP_RESULT LEP_SetAgcHeqClipLimitLow(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                     LEP_UINT16 agcHeqClipLimitLow)
```

4.4.7 AGC HEQ Empty Counts

This parameter specifies the maximum number of pixels in a bin that will be interpreted as an empty bin. Histogram bins with this number of pixels or less will be processed as an empty bin.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	2^14 -1	2	pixels	1

SDK Module ID: AGC **0x0100**

SDK Command ID: Base **0x3C**
With Get **0x3C**
With Set **0x3D**

SDK Data Length: Get **1** size of `LEP_UINT16` data type
Set **1** size of `LEP_UINT16` data type

C-SDK Commands	Description
`LEP_GetAgcHeqEmptyCount()`	Updates **`emptyCountPtr`** with the Camera's current HEQ transfer function's bin empty count
`LEP_SetAgcHeqEmptyCount()`	Sets Camera's current HEQ transfer function's bin empty count to **`emptyCount`**

C SDK Interface:

```
LEP_RESULT LEP_GetAgcHeqEmptyCount(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                   LEP_AGC_HEQ_EMPTY_COUNT_T_PTR emptyCountPtr)

LEP_RESULT LEP_SetAgcHeqEmptyCount(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                   LEP_AGC_HEQ_EMPTY_COUNT_T emptyCount)
```

4.4.8 AGC HEQ Output Scale Factor

This parameter specifies the output format for HEQ as either 8-bits (values range 0..255), or 14-bit (values range from 0..16383).

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_AGC_SCALE_TO_8_BITS	LEP_AGC_SCALE_TO_14_BITS	LEP_AGC_SCALE_TO_8_BITS	N/A	N/A

SDK Module ID: AGC **0x0100**

SDK Command ID: Base **0x44**
With Get **0x44**
With Set **0x45**

SDK Data Length: Get **2** size on an **enum** data type on a 32-bit machine
Set **2** size on an **enum** data type on a 32-bit machine

C-SDK Commands	Description
LEP_GetAgcHeqScaleFactor()	Updates **scaleFactorPtr** with the Camera's current AGC HEQ Output Scale Factor
LEP_SetAgcHeqScaleFactor()	Sets Camera's current AGC HEQ Output Scale Factor to **scaleFactor**

C SDK Interface:

```
LEP_RESULT LEP_GetAgcHeqScaleFactor (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                     LEP_AGC_HEQ_SCALE_FACTOR_E_PTR scaleFactorPtr)

LEP_RESULT LEP_SetAgcHeqScaleFactor (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                     LEP_AGC_HEQ_SCALE_FACTOR_E scaleFactor)

/* AGC Output Scale Factor Structure
*/
typedef enum LEP_AGC_SCALE_FACTOR_E_TAG
{
   LEP_AGC_SCALE_TO_8_BITS = 0,
   LEP_AGC_SCALE_TO_14_BITS,

   LEP_AGC_END_SCALE_TO
}LEP_AGC_HEQ_SCALE_FACTOR_E, *LEP_AGC_HEQ_SCALE_FACTOR_E_PTR;
```

4.4.9 AGC Calculation Enable State

This parameter controls the camera AGC calculations operations. If enabled, the current video histogram and AGC policy will be calculated for each input frame. If disabled, then no AGC calculations are performed and the current state of the ITT is preserved. For smooth AGC on /off operation, it is recommended to have this enabled.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_AGC_DISABLE	LEP_AGC_ENABLE	LEP_AGC_DISABLE	N/A	N/A

SDK Module ID: AGC **0x0100**

SDK Command ID: Base **0x48**
With Get **0x48**
With Set **0x49**

SDK Data Length: Get **2** size of `ENUM` data type
Set **2** size of `LEP_UINT16` data type

C-SDK Commands	Description
`LEP_GetAgcCalcEnableState()`	Updates `agcCalculationEnableStatePtr` with the Camera's current AGC Calculation enable state
`LEP_SetAgcCalcEnableState()`	Sets Camera's current AGC Calculation enable state to `agcCalculationEnableState`

C SDK Interface:

```
LEP_RESULT LEP_GetAgcCalcEnableState( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                      LEP_AGC_ENABLE_E_PTR agcCalculationEnableStatePtr )

LEP_RESULT LEP_SetAgcCalcEnableState( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                      LEP_AGC_ENABLE_E agcCalculationEnableState )

    /* AGC Enable Enum
    */
    typedef enum LEP_AGC_ENABLE_TAG
    {
        LEP_AGC_DISABLE=0,
        LEP_AGC_ENABLE,
        LEP_END_AGC_ENABLE

    }LEP_AGC_ENABLE_E, *LEP_AGC_ENABLE_E_PTR;
```

4.5 SDK Module: SYS 0x200

This module provides information and status of the camera system. This includes the camera serial number, current camera status, a method to ping the camera to verify communication, and Telemetry row enable and location control.

4.5.1 SYS Ping Camera

This function sends the ping command to the camera. The camera will respond with LEP_OK if command received correctly.

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x00**
With Run **0x02**

SDK Data Length: Run **0** size a run command argument is zero

C-SDK Commands	Description
`LEP_RunSysPing()`	Issues a ping command to the Camera to check if communication is up.

C SDK Interface:

```
LEP_RESULT LEP_RunSysPing(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr);
```

4.5.2 SYS Status

This command returns the system status: System Ready, System Initializing, System in Low-Power Mode, System Going into Standby, and FFC in Progress.

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x04**
With Get **0x04**

SDK Data Length: Get **4** size of the **LEP_STATUS_T data type**

C-SDK Commands	Description
LEP_GetSysStatus()	Updates **sysStatusPtr** with the Camera's current system status

C SDK Interface:

```
LEP_RESULT LEP_GetSysStatus(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                            LEP_STATUS_T_PTR sysStatusPtr)
```

```
typedef struct
{
   LEP_SYSTEM_STATUS_STATES_E  camStatus;
   LEP_UINT16                  commandCount;
   LEP_UINT16                  reserved;

}LEP_STATUS_T, *LEP_STATUS_T_PTR;


typedef enum LEP_SYSTEM_STATUS_STATES_E_TAG
{
    LEP_SYSTEM_READY=0,
    LEP_SYSTEM_INITIALIZING,
    LEP_SYSTEM_IN_LOW_POWER_MODE,
    LEP_SYSTEM_GOING_INTO_STANDBY,
    LEP_SYSTEM_FLAT_FIELD_IN_PROCESS,

    LEP_SYSTEM_END_STATES

}LEP_SYSTEM_STATUS_STATES_E, *LEP_SYSTEM_STATUS_STATES_E_PTR;
```

4.5.3 SYS FLIR Serial Number

This command returns the Lepton Camera's serial number as a 64-bit unsigned long integer (unsigned long long).

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x08**
With Get **0x08**

SDK Data Length: Get **4** size of the **LEP_UINT64 data type**

C-SDK Commands	Description
LEP_GetSysFlirSerialNumber()	Returns the Lepton Camera's serial number as a 64-bit unsigned long integer (unsigned long long).

C SDK Interface:

```
LEP_RESULT LEP_GetSysFlirSerialNumber(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                      LEP_SYS_FLIR_SERIAL_NUMBER_T_PTR sysSerialNumberBufPtr)


typedef LEP_UINT64  LEP_SYS_FLIR_SERIAL_NUMBER_T, *LEP_SYS_FLIR_SERIAL_NUMBER_T_PTR;
```

4.5.4 SYS Camera Uptime

This command returns the Lepton Camera's current uptime in milliseconds. The uptime is the time since the camera was brought out of Standby. The uptime counter is implemented as a 32-bit counter and as such will roll-over after the maximum count of 0xFFFFFFFF (1193 hours) is reached and restart at 0x00000000.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	4294967295	N/A	milliseconds	1

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x0C**
With Get **0x0C**

SDK Data Length: Get **2** size of the `LEP_UINT32` **data type**

C-SDK Commands	Description
`LEP_GetSysCameraUpTime()`	Updates **sysCameraUpTimePtr** with the Camera's current uptime in milliseconds

C SDK Interface:

```
LEP_RESULT LEP_GetSysCameraUpTime(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                  LEP_UINT32 *sysCameraUpTimePtr)
```

4.5.5 SYS AUX Temperature Kelvin

This command returns the Lepton Camera's AUX Temperature in Kelvin. This value is from a thermistor located on the Lepton housing.

Minimum Value	Maximum Value	Units	Scale factor
0	16383	Kelvin	100

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x10**
With Get **0x10**

SDK Data Length: Get **1** size of the **`LEP_SYS_AUX_TEMPERATURE_KELVIN_T` data type**

C-SDK Commands	Description
`LEP_GetSysAuxTemperatureKelvin()`	Returns the Lepton Camera's AUX Temperature in Kelvin

C SDK Interface:

```
LEP_RESULT LEP_GetSysAuxTemperatureKelvin(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                        LEP_SYS_AUX_TEMPERATURE_KELVIN_T_PTR auxTemperaturePtr);

typedef LEP_UINT16  LEP_SYS_AUX_TEMPERATURE_KELVIN_T, *LEP_SYS_AUX_TEMPERATURE_KELVIN_T_PTR;
```

4.5.6 SYS FPA Temperature Kelvin

This command returns the Lepton Camera's FPA Temperature in Kelvin.

Minimum Value	Maximum Value	Units	Scale factor
0	65535	Kelvin	100

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x14**
With Get **0x14**

SDK Data Length: Get **1** size of the **`LEP_SYS_FPA_TEMPERATURE_KELVIN_T` data type**

C-SDK Commands	Description
`LEP_GetSysFpaTemperatureKelvin()`	Returns the Lepton Camera's FPA Temperature in Kelvin

C SDK Interface:

```
LEP_RESULT LEP_GetSysFpaTemperatureKelvin(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                          LEP_SYS_FPA_TEMPERATURE_KELVIN_T_PTR fpaTemperaturePtr)

typedef LEP_UINT16  LEP_SYS_FPA_TEMPERATURE_KELVIN_T, *LEP_SYS_FPA_TEMPERATURE_KELVIN_T_PTR;
```

4.5.7 SYS Telemetry Enable State

This command returns the Telemetry Enabled State as an Enum.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_TELEMETRY_DISABLED	LEP_TELEMETRY_ENABLED	LEP_TELEMETRY_DISABLED	N/A	N/A

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x18**
With Get **0x18**
With Set **0x19**

SDK Data Length: Get **2** size of an Enum on a 32-bit machine

C-SDK Commands	Description
LEP_ GetSysTelemetryEnableState()	Returns the Lepton Camera's Telemetry Enable State
LEP_ SetSysTelemetryEnableState()	Sets the Lepton Camera's Telemetry Enabled State

C SDK Interface:

```
LEP_RESULT LEP_GetSysTelemetryEnableState(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                LEP_SYS_TELEMETRY_ENABLE_STATE_E_PTR enableStatePtr)

LEP_RESULT LEP_SetSysTelemetryEnableState(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                LEP_SYS_TELEMETRY_ENABLE_STATE_E enableState)
```

```
typedef enum LEP_SYS_TELEMETRY_ENABLE_STATE_E_TAG
{
    LEP_TELEMETRY_DISABLED=0,
    LEP_TELEMETRY_ENABLED,
    LEP_END_TELEMETRY_ENABLE_STATE

}LEP_SYS_TELEMETRY_ENABLE_STATE_E, *LEP_SYS_TELEMETRY_ENABLE_STATE_E_PTR;
```

4.5.8 SYS Telemetry Location

This command Sets and Gets the Telemetry Location

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_TELEMETRY_LOCATION_HEADER	LEP_TELEMETRY_LOCATION_FOOTER	LEP_TELEMETRY_LOCATION_FOOTER	N/A	N/A

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x1C**
With Get **0x1C**
With Set **0x1D**

SDK Data Length: Get **2** size of an Enum on a 32-bit machine

C-SDK Commands	Description
LEP_ GetSysTelemetryLocation()	Returns the location of Telemetry data as an enum
LEP_ SetSysTelemetryLocation()	Sets the location of Telemetry data as an enum

C SDK Interface:

```
LEP_RESULT LEP_GetSysTelemetryLocation(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_SYS_TELEMETRY_LOCATION_E_PTR telemetryLocationPtr)

LEP_RESULT LEP_SetSysTelemetryLocation(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_SYS_TELEMETRY_LOCATION_E telemetryLocation)
```

```
typedef enum LEP_SYS_TELEMETRY_LOCATION_E_TAG
{
    LEP_TELEMETRY_LOCATION_HEADER=0,
    LEP_TELEMETRY_LOCATION_FOOTER,
    LEP_END_TELEMETRY_LOCATION

}LEP_SYS_TELEMETRY_LOCATION_E, *LEP_SYS_TELEMETRY_LOCATION_E_PTR;
```

4.5.9 SYS Number of Frames to Average

This command Gets or Sets the number of frames to average when executing a Flat-Field Correction (FFC) (see **Error! Reference source not found.**).

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_SYS_FA_DIV_1	LEP_SYS_FA_DIV_128	LEP_SYS_FA_DIV_8	N/A	N/A

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x24**
With Get **0x24**
With Set **0x25**

SDK Data Length: Get **2** size of an Enum on a 32-bit machine
Set **2** size of an Enum on a 32-bit machine

C-SDK Commands	Description
LEP_SYS_GetFramesToAverage()	Gets the number of frames to average
LEP_SYS_SetFramesToAverage()	Sets number of frames to average

C SDK Interface:

```
LEP_RESULT LEP_SYS_GetFramesToAverage(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                      LEP_SYS_FRAME_AVERAGE_DIVISOR_E_PTR numFrameToAveragePtr);

LEP_RESULT LEP_SYS_SetFramesToAverage(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                      LEP_SYS_FRAME_AVERAGE_DIVISOR_E numFrameToAverage);
```

```
typedef enum LEP_SYS_FRAME_AVERAGE_DIVISOR_E_TAG
{
    LEP_SYS_FA_DIV_1 = 0,
    LEP_SYS_FA_DIV_2,
    LEP_SYS_FA_DIV_4,
    LEP_SYS_FA_DIV_8,
    LEP_SYS_FA_DIV_16,
    LEP_SYS_FA_DIV_32,
    LEP_SYS_FA_DIV_64,
    LEP_SYS_FA_DIV_128,
    LEP_SYS_END_FA_DIV

}LEP_SYS_FRAME_AVERAGE_DIVISOR_E, *LEP_SYS_FRAME_AVERAGE_DIVISOR_E_PTR;
```

4.5.10 SYS Camera Customer Serial Number

This command returns the Lepton Camera's Customer serial number as a 32-byte character string. The Customer Serial Number is a (32 byte string) identifier unique to a specific configuration of module; essentially a module Configuration ID. This serial number is unwritten in the current release.

This command requires the Host to allocate the memory buffer before calling this function. The address to this memory block should be passed in as **`sysSerialNumberPtr`**

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x28**
With Get **0x28**

SDK Data Length: Get **16** 32-byte string Data type

C-SDK Commands	Description
`LEP_GetSysCustSerialNumber()`	Updates **`sysSerialNumberPtr`** with the Camera's 32-byte serial number.

C SDK Interface:

```
LEP_RESULT LEP_GetSysCustSerialNumber( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_SYS_CUST_SERIAL_NUMBER_T_PTR sysSerialNumberPtr )

typedef LEP_CHAR8 *LEP_SYS_CUST_SERIAL_NUMBER_T, *LEP_SYS_CUST_SERIAL_NUMBER_T_PTR;
```

4.5.11 SYS Camera Video Scene Statistics

This command returns the current scene statistics for the video frame defined by the SYS ROI (see section 4.5.12). The statistics captured are scene mean intensity in counts, minimum and maximum intensity in counts, and the number of pixels in the ROI. Lepton scene intensities range from 0 to 16383. The range drops to 0 to 255 when in 8-bit AGC mode. Maximum number of pixels in the scene is 4800.

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x2C**
With Get **0x2C**

SDK Data Length: Get **4** Returns four 16-bit values

C-SDK Commands	Description
LEP_GetSysSceneStatistics()	Updates **sceneStatisticsPtr** with the Camera's current scene statistics.

C SDK Interface:

```
LEP_RESULT LEP_GetSysSceneStatistics( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                      LEP_SYS_SCENE_STATISTICS_T_PTR sceneStatisticsPtr )
```

```
typedef struct LEP_SYS_SCENE_STATISTICS_T_TAG
{
   LEP_UINT16   meanIntensity;
   LEP_UINT16   maxIntensity;
   LEP_UINT16   minIntensity;
   LEP_UINT16   numPixels;

} LEP_SYS_SCENE_STATISTICS_T, *LEP_SYS_SCENE_STATISTICS_T_PTR;
```

4.5.12 SYS Scene ROI Select

The camera supports processing of pixels contained within a specified rectangular window or Region of Interest (ROI) to calculate scene statistics (See 4.5.11). This region is defined by 4 parameters: start column, start row, end column, and end row. The region is adjustable to a sub-window.

Dimension	Minimum Value	Maximum Value	Default Value	Units	Scale factor
start column	0	< end column	0	pixels	1
start row	0	< end row	0	pixels	1
end column	> start column	79	79	pixels	1
end row	> start row	59	59	pixels	1

SDK Module ID: VID **0x0200**

SDK Command ID: Base **0x30**
With Get **0x30**
With Set **0x31**

SDK Data Length: Get **4** size of `LEP_VID_FOCUS_ROI_T` data type
Set **4** size of `LEP_VID_FOCUS_ROI_T` data type

C-SDK Commands	Description
`LEP_GetSysSceneRoi()`	Updates `sceneRoiPtr` with the Camera's current Scene ROI
`LEP_ SetSysSceneRoi()`	Sets Camera's current Scene ROI to `sceneRoi`

C SDK Interface:

```
LEP_RESULT LEP_GetSysSceneRoi(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                              LEP_SYS_VIDEO_ROI_T_PTR sceneRoiPtr)

LEP_RESULT LEP_SetSysSceneRoi(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                              LEP_SYS_VIDEO_ROI_T sceneRoi)
```

```
/* SYS Scene ROI Structure
*/
typedef struct LEP_SYS_VIDEO_ROI_T_TAG
{
   LEP_UINT16 startCol;
   LEP_UINT16 startRow;
   LEP_UINT16 endCol;
   LEP_UINT16 endRow;

} LEP_SYS_VIDEO_ROI_T, *LEP_SYS_VIDEO_ROI_T_PTR;
```

4.5.13 SYS Thermal Shutdown Count

This command returns the current number of frames remaining before a thermal shutdown is executed once the camera temperature exceeds a high-temperature threshold (around 80 degrees C). Once the camera detects the camera exceeded the thermal threshold, this counter begins to count down until zero. When the count reaches ZERO, the camera will shut itself down. A host can use this value to determine when the camera shuts down due to thermal conditions. The default value of 270 is just over 10 seconds at 26 Hz video.

Dimension	Minimum Value	Maximum Value	Default Value	Units	Scale factor
thermalCounts	0	65535	270	pixels	1

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x34**
With Get **0x34**

SDK Data Length: Get **1** Returns one 16-bit value

C-SDK Commands	Description
`LEP_GetSysThermalShutdownCount()`	Updates `thermalCountsPtr` with the Camera's current thermal shut down count value.

C SDK Interface:

```
LEP_RESULT LEP_GetSysThermalShutdownCount(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                          LEP_SYS_THERMAL_SHUTDOWN_COUNTS_T_PTR thermalCountsPtr)


typedef LEP_UINT16  LEP_SYS_THERMAL_SHUTDOWN_COUNTS_T, *LEP_SYS_THERMAL_SHUTDOWN_COUNTS_T_PTR
```

4.5.14 SYS Shutter Position Control

This command is used to manually control the position of the attached shutter if one exists. If there is an attached shutter, then this command will return its current position. If there is no shutter attached, it will return LEP_SYS_SHUTTER_POSITION_UNKNOWN.

Minimum Value	Maximum Value	Default Value	Units	Scale factor
LEP_SYS_SHUTTER_POSITION_UNKNOWN	LEP_SYS_SHUTTER_POSITION_BRAKE_ON	LEP_SYS_SHUTTER_POSITION_UNKNOWN	N/A	1

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x38**
With Get **0x38**
With Set **0x39**

SDK Data Length: Get **2** size of an Enum on a 32-bit machine
Set **2** size of an Enum on a 32-bit machine

C-SDK Commands	Description
LEP_GetShutterPosition()	Updates **shutterPositionPtr** with the Camera's attached shutter current position.
LEP_SetShutterPosition()	Sets the Camera's attached shutter current position to **shutterPosition**

C SDK Interface:

```
LEP_RESULT LEP_GetShutterPosition(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                  LEP_SYS_SHUTTER_POSITION_E_PTR shutterPositionPtr)

LEP_RESULT LEP_SetShutterPosition(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                  LEP_SYS_SHUTTER_POSITION_E shutterPosition)
```

```
typedef enum LEP_SYS_SHUTTER_POSITION_E_TAG
{
   LEP_SYS_SHUTTER_POSITION_UNKNOWN = -1,
   LEP_SYS_SHUTTER_POSITION_IDLE = 0,
   LEP_SYS_SHUTTER_POSITION_OPEN,
   LEP_SYS_SHUTTER_POSITION_CLOSED,
   LEP_SYS_SHUTTER_POSITION_BRAKE_ON,
   LEP_SYS_SHUTTER_POSITION_END

}LEP_SYS_SHUTTER_POSITION_E,  *LEP_SYS_SHUTTER_POSITION_E_PTR;
```

4.5.15 SYS FFC Mode Control

This command controls the FFC mode and shutter control during an FFC. FFC modes allow for manual control, automatic control based upon time or temperature changes, and external control. If a shutter is attached this command controls the shutter activity profile.

SDK Module ID: VID **0x0200**

SDK Command ID: Base **0x3C**
With Get **0x3C**
With Set **0x3D**

SDK Data Length: Get **20** size of `LEP_SYS_FFC_SHUTTER_MODE_OBJ_T` data type
Set **20** size of `LEP_SYS_FFC_SHUTTER_MODE_OBJ_T` data type

C-SDK Commands	Description
`LEP_GetFfcShutterModeObj()`	Updates `shutterModeObjPtr` with the Camera's current FFC mode and shutter control
`LEP_SetFfcShutterModeObj()`	Sets Camera's current FFC mode and shutter control to `shutterModeObj`

C SDK Interface:

```
LEP_RESULT LEP_GetFfcShutterModeObj( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                     LEP_SYS_FFC_SHUTTER_MODE_OBJ_T_PTR shutterModeObjPtr )

LEP_RESULT LEP_SetFfcShutterModeObj( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                     LEP_SYS_FFC_SHUTTER_MODE_OBJ_T shutterModeObj )
```

```
/* SYS FFC Shutter Mode Structure
*/
typedef struct LEP_SYS_FFC_SHUTTER_MODE_OBJ_T_TAG
{
   LEP_SYS_FFC_SHUTTER_MODE_E shutterMode;    /* defines current mode */
   LEP_SYS_SHUTTER_TEMP_LOCKOUT_STATE_E   tempLockoutState;
   LEP_SYS_ENABLE_E videoFreezeDuringFFC;
   LEP_SYS_ENABLE_E ffcDesired;               /* status of FFC desired */
   LEP_UINT32 elapsedTimeSinceLastFfc;        /* in milliseconds x1 */
   LEP_UINT32 desiredFfcPeriod;               /* in milliseconds x1 */
   LEP_BOOL   explicitCmdToOpen;              /* true or false */
   LEP_UINT16 desiredFfcTempDelta;            /* in Kelvin x100  */
   LEP_UINT16 imminentDelay;                  /* in frame counts x1 */

}LEP_SYS_FFC_SHUTTER_MODE_OBJ_T, *LEP_SYS_FFC_SHUTTER_MODE_OBJ_T_PTR;


typedef enum LEP_SYS_FFC_SHUTTER_MODE_E_TAG
{
   LEP_SYS_FFC_SHUTTER_MODE_MANUAL = 0,
   LEP_SYS_FFC_SHUTTER_MODE_AUTO,
   LEP_SYS_FFC_SHUTTER_MODE_EXTERNAL,

   LEP_SYS_FFC_SHUTTER_MODE_END
```

```
}LEP_SYS_FFC_SHUTTER_MODE_E, *LEP_SYS_FFC_SHUTTER_MODE_E_PTR;


typedef enum LEP SYS SHUTTER TEMP LOCKOUT STATE E TAG
{
   LEP_SYS_SHUTTER_LOCKOUT_INACTIVE = 0,        /* not locked out */
   LEP_SYS_SHUTTER_LOCKOUT_HIGH,                /* lockout due to high temp */
   LEP_SYS_SHUTTER_LOCKOUT_LOW,                 /* lockout due to low temp */

}LEP SYS SHUTTER TEMP LOCKOUT STATE E,*LEP SYS SHUTTER TEMP LOCKOUT STATE E PTR;


typedef enum LEP SYS ENABLE E TAG
{
   LEP SYS DISABLE = 0,
   LEP SYS ENABLE,

   LEP_END_SYS_ENABLE

}LEP SYS ENABLE E, *LEP SYS ENABLE E PTR;
```

Dimension	Minimum Value	Maximum Value	Default Value
shutterMode	LEP_SYS_FFC_SHUTTER_MODE_MANUAL	LEP_SYS_FFC_SHUTTER_MODE_EXTERNAL	LEP_SYS_FFC_SHUTTER_MODE_EXTERNAL
shutterLockout tempLockoutState	LEP_SYS_SHUTTER_LOCKOUT_INACTIVE	LEP_SYS_SHUTTER_LOCKOUT_LOW	LEP_SYS_SHUTTER_LOCKOUT_INACTIVE
videoFreezeDuringFFC	LEP_SYS_DISABLE	LEP_SYS_ENABLE	LEP_SYS_ENABLE
ffcDesired	LEP_SYS_DISABLE	LEP_SYS_ENABLE	LEP_SYS_DISABLE

Dimension	Minimum Value	Maximum Value	Default Value	Units	Scale
desiredFfcPeriod	0	4294967295	300000	milliseconds	1
desiredFfcTempDelta	0	65535	300	Kelvin	100
imminentDelay	0	65535	52	frames	1
closePeriodInFrames	0	65535	4	frames	1
openPeriodInFrames	0	65535	1	frames	1
explicitCmdToOpen	0 (false)	1 (true)	0 (false)	N/A	1
elapsedTimeSinceLastFfc	0	4294967295	0	milliseconds	1

4.5.16 SYS Run FFC Normalization

This command executes the camera's Flat-Field Correction (FFC) normalization. This command executes synchronously. Internally this command polls the camera status to determine when this command completes (see LEP_GetSysFFCStatus()), and only returns when completed.

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x40**
With Run **0x42**

SDK Data Length: Run **0** size a run command argument

C-SDK Commands	Description
`LEP_RunSysFFCNormalization()`	Executes the FFC command.

C SDK Interface:

```
LEP_RESULT LEP_RunSysFFCNormalization(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr);
```

4.5.17 SYS FFC Status

This command returns the Flat-Field Correction normalization (FFC) status.

Dimension	Minimum Value	Maximum Value	Default Value
ffcStatusPtr	LEP_SYS_STATUS_WRITE_ERROR	LEP_SYS_FRAME_AVERAGE_COLLECTING_FRAMES	LEP_SYS_STATUS_READY

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x44**
With Get **0x44**

SDK Data Length: Get **2** size of an Enum on a 32-bit machine

C-SDK Commands	Description
LEP_GetSysFFCStatus()	Returns the current status of the FFC operation in **ffcStatusPtr**

C SDK Interface:

```
LEP_RESULT LEP_GetSysFFCStatus( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                LEP_SYS_STATUS_E_PTR ffcStatusPtr ))


typedef enum
{
   LEP_SYS_STATUS_WRITE_ERROR = -2,//
   LEP_SYS_STATUS_ERROR = -1,
   LEP_SYS_STATUS_READY = 0,
   LEP_SYS_STATUS_BUSY,
   LEP_SYS_FRAME_AVERAGE_COLLECTING_FRAMES,
   LEP_SYS_STATUS_END

} LEP_SYS_STATUS_E, *LEP_SYS_STATUS_E_PTR;
```

4.5.18 SYS AUX Temperature Celsius – *helper function*

This is a SDK command that returns the Lepton Camera's AUX Temperature in degrees Celsius. This function has no command ID since it is a helper function and uses the function **`LEP_GetSysAuxTemperatureKelvin()`** to get the current temperature in Kelvin before converting to degrees Celsius.

Minimum Value	Maximum Value	Units	Scale factor
-	-	Degrees Celsius	N/A (float value)

C-SDK Commands	Description
`LEP_GetSysAuxTemperatureCelcius()`	Returns the Lepton Camera's AUX Temperature in degrees Celsius

C SDK Interface:

```
LEP_RESULT LEP_GetSysAuxTemperatureCelcius( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                            LEP_SYS_AUX_TEMPERATURE_CELCIUS_T_PTR auxTemperaturePtr )

typedef LEP_FLOAT32 LEP_SYS_AUX_TEMPERATURE_CELCIUS_T, *LEP_SYS_AUX_TEMPERATURE_CELCIUS_T_PTR;
```

4.5.19 SYS FPA Temperature Celsius – *helper function*

This is a SDK command that returns the Lepton Camera's F{A Temperature in degrees Celsius. This function has no command ID since it is a helper function and uses the function **`LEP_GetSysFpaTemperatureKelvin()`** to get the current temperature in Kelvin before converting to degrees Celsius.

Minimum Value	Maximum Value	Units	Scale factor
-	-	Degrees Celsius	N/A (float value)

C-SDK Commands	Description
`LEP_GetSysFpaTemperatureCelcius()`	Returns the Lepton Camera's FPA Temperature in degrees Celsius

C SDK Interface:

```
LEP_RESULT LEP_GetSysFpaTemperatureCelcius(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                           LEP_SYS_FPA_TEMPERATURE_CELCIUS_T_PTR fpaTemperaturePtr)
```

```
typedef LEP_FLOAT32 LEP_SYS_FPA_TEMPERATURE_CELCIUS_T, *LEP_SYS_FPA_TEMPERATURE_CELCIUS_T_PTR;
```

4.6 SDK Module: VID 0x300

This module provides command and control of the video data. Selection of the video polarity (white-hot or black-hot), video output color look-up table, and access to reading the focus metric are available through this module.

4.6.1 VID Pseudo-Color Look-Up Table Select

This function allows selection of the video output pseudo-color LUT. This LUT applies to the video processed by camera post AGC application before output. Color LUTs do not apply to raw video output of any format. Requires using the video output format of 24-bit R,G,B. (See **Error! Reference source not found.**.), AGC enabled and scaled to 8-bit output (See **Error! Reference source not found.** and **Error! Reference source not found.**).

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_VID_GREYSCALE_LUT	LEP_VID_USER_LUT	LEP_VID_GREYSCALE_LUT	N/A	N/A

SDK Module ID: VID **0x0300**

SDK Command ID: Base **0x04**
With Get **0x04**
With Set **0x05**

SDK Data Length: Get **2** size on an **Enum** data type on a 32-bit machine
Set **2** size on an **Enum** data type on a 32-bit machine

C-SDK Commands	Description
LEP_GetVidPcolorLut()	Updates **vidPcolorLutPtr** with the Camera's current video pseudo-color LUT selection.
LEP_SetVidPcolorLut()	Sets Camera's current video pseudo-color LUT selection to **vidPcolorLut**

C SDK Interface:

```
LEP_RESULT LEP_GetVidPcolorLut(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                               LEP_PCOLOR_LUT_E_PTR vidPcolorLutPtr)


LEP_RESULT LEP_SetVidPcolorLut(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                               LEP_PCOLOR_LUT_E vidPcolorLut)
/* Video Pseudo-Color LUT Enum
*/
 typedef enum LEP_PCOLOR_LUT_E_TAG
 {
   LEP_VID_WHEEL6_LUT=0,
   LEP_VID_FUSION_LUT,
   LEP_VID_RAINBOW_LUT,
   LEP_VID_GLOBOW_LUT,
   LEP_VID_SEPIA_LUT,
   LEP_VID_COLOR_LUT,
   LEP_VID_ICE_FIRE_LUT,
   LEP_VID_RAIN_LUT,
   LEP_VID_USER_LUT,
   LEP_VID_END_PCOLOR_LUT

 }LEP_PCOLOR_LUT_E, *LEP_PCOLOR_LUT_E_PTR;
```

4.6.2 VID User Pseudo-Color Look-Up Table Upload/Download

This function allows uploading (SET to the camera), and downloading (GET from the camera) a user-defined video output pseudo-color LUT. This LUT applies to the video processed by camera post AGC application before output. Does not apply to raw video output. The format of the pseudo-color LUT is 256 x 32-bits.

Parameter	Minimum Value	Maximum Value	Default Setting	Units	Scale factor
reserved	0	0	N/A	N/A	1
red	0	255	N/A	N/A	1
green	0	255	N/A	N/A	1
blue	0	255	N/A	N/A	1

SDK Module ID: VID **0x0300**

SDK Command ID: Base **0x08**
With Get **0x08**
With Set **0x09**

SDK Data Length: Get **512** size of `LEP_VID_LUT_BUFFER_T` data type
Set **512** size of `LEP_VID_LUT_BUFFER_T` data type

C-SDK Commands	Description
LEP_GetVidUserLut()	Updates **vidUserLutBufPtr** with the Camera's current user-defined video pseudo-color LUT data. Length of the LUT is 1024 bytes supporting a 256 x 32-bit LUT format and passed as value in **vidUserLutBufLen.**
LEP_SetVidUserLut()	Updates the Camera's current user-defined video pseudo-color LUT data with the contents of **vidUserLutBufPtr**. Length of the LUT is 1024 bytes supporting 256 x 32-bit LUT format and passed as value in **vidUserLutBufLen.**

C SDK Interface:

```
LEP_RESULT LEP_GetVidUserLut(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                             LEP_UINT8 *vidUserLutBufPtr, LEP_UINT16 vidUserLutBufLen)


LEP_RESULT LEP_SetVidUserLut(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                             LEP_UINT8 *vidUserLutBufPtr, LEP_UINT16 vidUserLutBufLen)
/* User-Defined color look-up table (LUT)
*/
typedef struct LEP_VID_LUT_PIXEL_T_TAG
{
   LEP_UINT8 reserved;
   LEP_UINT8 red;
   LEP_UINT8 green;
   LEP_UINT8 blue;
} LEP_VID_LUT_PIXEL_T, *LEP_VID_LUT_PIXEL_T_PTR;

typedef struct LEP_VID_LUT_BUFFER_T_TAG
{
   LEP_VID_LUT_PIXEL_T bin[256];
} LEP_VID_LUT_BUFFER_T, *LEP_VID_LUT_BUFFER_T_PTR;
```

4.6.3 VID Focus Calculation Enable State

The camera can calculate a video scene focus metric (also useful as a metric of contrast). This function specifies whether or not the camera is to make these calculations on the input video. When enabled, the camera will calculate the video scene focus metric on each frame processed and make the result available in the focus metric. See section 4.6.6. When disabled, the camera does not execute the focus metric calculation.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_VID_FOCUS_CALC_DISABLE	LEP_VID_FOCUS_CALC_ENABLE	LEP_VID_FOCUS_CALC_DISABLE	N/A	N/A

SDK Module ID: VID **0x0300**

SDK Command ID: Base **0x0C**
With Get **0x0C**
With Set **0x0D**

SDK Data Length: Get **2** size on an **Enum** data type on a 32-bit machine
Set **2** size on an **Enum** data type on a 32-bit machine

C-SDK Commands	Description
LEP_GetVidFocusCalcEnableState()	Updates **vidEnableFocusCalcStatePtr** with the Camera's current video focus calculation enable state.
LEP_SetVidFocusCalcEnableState()	Updates the Camera's current video focus calculation enable state with the contents of **vidFocusCalcEnableState**.

C SDK Interface:

```
LEP_RESULT LEP_GetVidFocusCalcEnableState(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                          LEP_VID_FOCUS_CALC_ENABLE_E_PTR
                                          vidEnableFocusCalcStatePtr)

LEP_RESULT LEP_SetVidFocusCalcEnableState(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                          LEP_VID_FOCUS_CALC_ENABLE_E
                                          vidFocusCalcEnableState)

/* Video Focus Metric Calculation Enable Enum
*/
typedef enum LEP_VID_ENABLE_TAG
{
   LEP VID FOCUS CALC DISABLE=0,
   LEP VID FOCUS CALC ENABLE,
   LEP_VID_END_FOCUS_CALC_ENABLE

}LEP_VID_FOCUS_CALC_ENABLE_E, *LEP_VID_FOCUS_CALC_ENABLE_E_PTR;
```

4.6.4 VID Focus ROI Select

The camera supports processing of pixels contained within a specified rectangular window or Region of Interest (ROI) to calculate a focus metric. This region is defined by 4 parameters: start column, start row, end column, and end row. The region is adjustable to a sub-window. Maximum extents must exclude a 1-pixel boundary from any edge.

Dimension	Minimum Value	Maximum Value	Default Value	Units	Scale factor
start column	1	< end column-1	1	pixels	1
start row	1	< end row-1	1	pixels	1
end column	> start column+1	78	78	pixels	1
end row	> start row+1	58	58	pixels	1

SDK Module ID: VID **0x0300**

SDK Command ID: Base **0x10**
With Get **0x10**
With Set **0x11**

SDK Data Length: Get **4** size of `LEP_VID_FOCUS_ROI_T` data type
Set **4** size of `LEP_VID_FOCUS_ROI_T` data type

C-SDK Commands	Description
`LEP_GetVidFocusROI()`	Updates **`vidFocusROIPtr`** with the Camera's current video focus ROI
`LEP_SetVidFocusROI()`	Sets Camera's current video focus ROI to **`vidFocusROI`**

C SDK Interface:

```
LEP_RESULT LEP_GetVidFocusROI(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                              LEP_VID_FOCUS_ROI_T_PTR vidFocusROIPtr)

LEP_RESULT LEP_SetVidFocusROI(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                              LEP_VID_FOCUS_ROI_T vidFocusROI)

/* VIDFOCUS ROI Structure
*/
 typedef struct LEP_VID_FOCUS_ROI_TAG
 {
     LEP_USHORT startCol;
     LEP_USHORT startRow;
     LEP_USHORT endCol;
     LEP_USHORT endRow;

 }LEP_VID_FOCUS_ROI_T, *LEP_VID_FOCUS_ROI_T_PTR;
```

4.6.5 VID Focus Metric Threshold

This function specifies the focus metric threshold. The focus metric evaluates image gradients and counts the number of gradient magnitudes that exceed the focus metric threshold. Therefore, larger values of the threshold imply the focus metric is counting gradients with larger magnitudes in effect filtering out small gradients in the image (pixel noise, for example). The Focus Metric uses the Tenengrad method which is an edge-based metric that measures the sum of the horizontal and vertical gradients using Sobel operators. The Focus Metric Threshold is applied to the sum of gradients. Gradients that exceed this threshold are then summed and counted and the Focus metric is computed from these.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	4294967295	30	N/A	N/A

SDK Module ID: VID **0x0300**

SDK Command ID: Base **0x14**
With Get **0x14**
With Set **0x15**

SDK Data Length: Get **2** size of `LEP_VID_FOCUS_METRIC_THRESHOLD_T` data type
Set **2** size of `LEP_VID_FOCUS_METRIC_THRESHOLD_T` data type

C-SDK Commands	Description
`LEP_GetVidFocusMetricThreshold()`	Updates `vidFocusMetricThresholdPtr` with the Camera's current video focus metric threshold.
`LEP_SetVidFocusMetricThreshold()`	Updates the Camera's current video focus metric threshold with the contents of `vidFocusMetricThreshold`.

C SDK Interface:

```
LEP_RESULT LEP_GetVidFocusMetricThreshold(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                LEP_VID_FOCUS_METRIC_THRESHOLD_T_PTR vidFocusMetricThresholdPtr)

LEP_RESULT LEP_SetVidFocusMetricThreshold(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                LEP_VID_FOCUS_METRIC_THRESHOLD_T vidFocusMetricThreshold)


typedef LEP_UINT32 LEP_VID_FOCUS_METRIC_THRESHOLD_T, *LEP_VID_FOCUS_METRIC_THRESHOLD_T_PTR;
```

4.6.6 VID Focus Metric

This function returns the most recently calculated scene focus metric. The focus metric calculation counts image gradients that exceed the focus metric threshold. Larger values imply better scene focus due the presence of more large gradients. The focus metric is not defined if the video scene focus metric calculations are not enabled. . The focus metric uses the Tenengrad method, an edge-based metric that measures the sum of the horizontal and vertical gradients using Sobel operators. The focus metric threshold is applied to the sum of gradients. Gradients that exceed this threshold are then summed and counted and the Focus metric is computed from these.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	4294967295	N/A	none	1

SDK Module ID: VID **0x0300**

SDK Command ID: Base **0x18**
With Get **0x18**

SDK Data Length: Get **2** size of **LEP_VID_FOCUS_METRIC_T** data type

C-SDK Commands	Description
LEP_GetVidFocusMetric()	Updates **vidFocusMetricPtr** with the Camera's current video focus value. Not defined if focus calculation is not enabled.

C SDK Interface:

```
LEP_RESULT LEP_GetVidFocusMetric(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                 LEP_VID_FOCUS_METRIC_T_PTR vidFocusMetricPtr)


typedef LEP_UINT32 LEP_VID_FOCUS_METRIC_T, *LEP_VID_FOCUS_METRIC_T_PTR;
```

4.6.7 VID Video Freeze Enable State

This function allows the current frame to be repeated in lieu of a live video stream. When disabled, live video resumes.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_VID_FREEZE_DISABLE	LEP_VID_FREEZE_ENABLE	LEP_VID_FREEZE_DISABLE	N/A	N/A

SDK Module ID: VID **0x0300**

SDK Command ID: Base **0x24**
With Get **0x24**
With Set **0x25**

SDK Data Length: Get **2** size on an **Enum** data type on a 32-bit machine
Set **2** size on an **Enum** data type on a 32-bit machine

C-SDK Commands	Description
`LEP_GetVidFreezeEnableState()`	Updates `vidFreezeEnableStatePtr` with the Camera's current Video Freeze enable state
`LEP_SetVidFreezeEnableState()`	Updates the Camera's current Video Freeze enable state with the contents of `vidFreezeEnableState`.

C SDK Interface:

```
LEP_RESULT LEP_GetVidFreezeEnableState(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_VID_FREEZE_ENABLE_E_PTR vidFreezeEnableStatePtr)

LEP_RESULT LEP_SetVidFreezeEnableState(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_VID_FREEZE_ENABLE_E vidFreezeEnableState)
```

```
/* Video Freeze Output Enable Enum
*/
typedef enum LEP_VID_FREEZE_ENABLE_TAG
{
   LEP_VID_FREEZE_DISABLE = 0,
   LEP_VID_FREEZE_ENABLE,
   LEP_VID_END_FREEZE_ENABLE

}LEP_VID_FREEZE_ENABLE_E, *LEP_VID_FREEZE_ENABLE_E_PTR ;
```