FLIR LEPTON®

Software Interface Description Document (IDD)

Document Number: 110-0144-04 Rev 200

Table of Contents

List of Tables

List of Figures

1 Document Description

1.1 Revision History

Rev. #	Date	Comments
100	December 8, 2016	Official release for Lepton and Lepton with Radiometry configurations
200	April 4, 2017	Updates to include Lepton 3 (160x120) Release

1.2 Scope

This interface description document (IDD) defines software interface requirements and software commands available to a Host for Lepton, Lepton with Radiometry, and the Lepton 3 configurations. This version of the IDD includes AGC, SYS, VID, OEM and RAD module commands.

1.3 References

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

1.3.1 FLIR Systems Documents

500-0763-01-09	Lepton with Radiometry Datasheet
500-0726-01-09	Lepton 3 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

1.3.4 Factory Defaults vs. Software Defaults

For each API, there may be an associated default value for any particular attribute. However, depending on the camera configurations or version, the actual defaults may be different than the one's listed. You will find the actual defaults for a particular configuration under columns labelled as "Factory Default".

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.

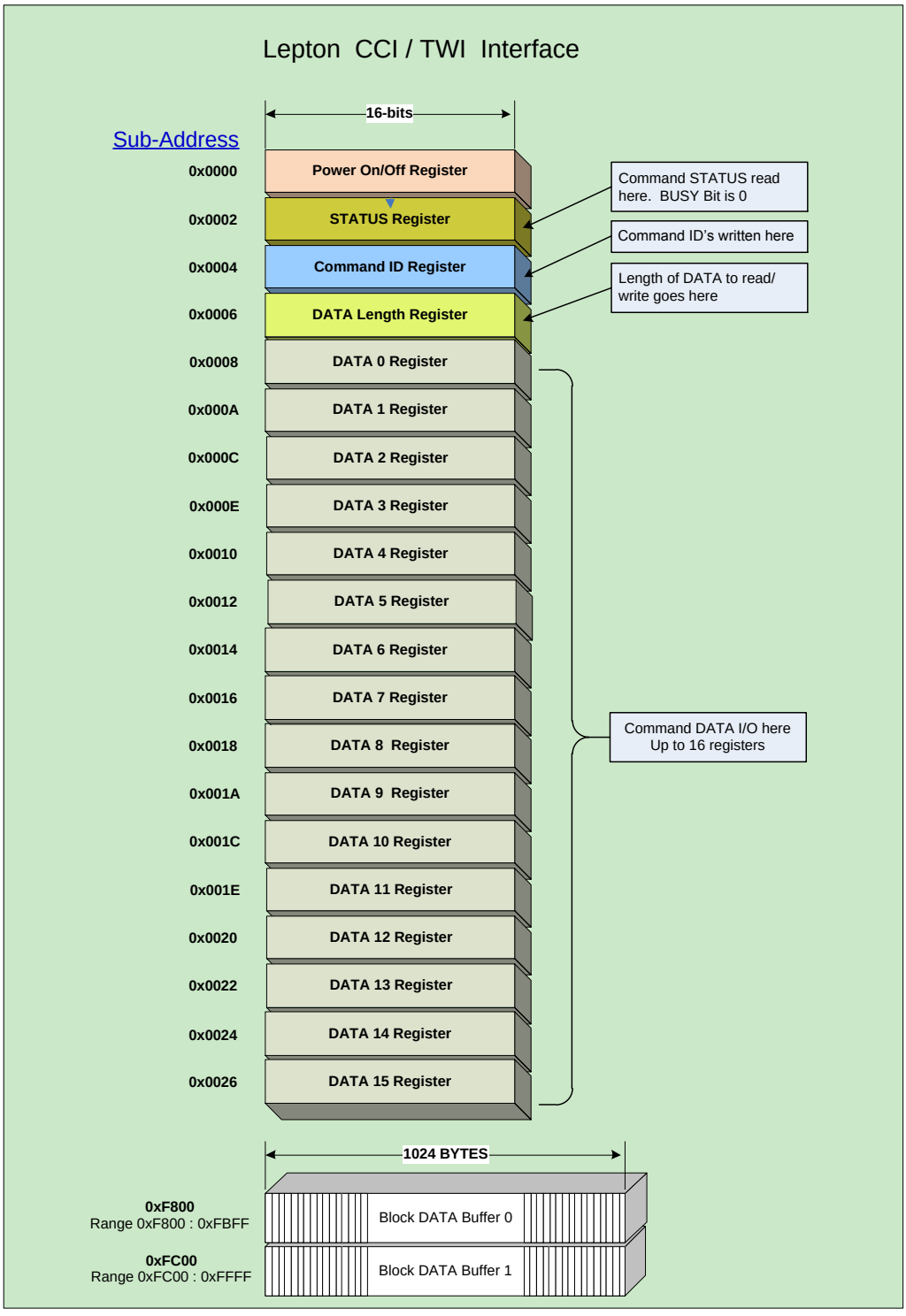

Figure 1 Lepton CCI/TWI Registers

Typical transmission requires the sequence of:

1. Polling the status register until camera is ready for a new command (BUSY bit clear).
2. Polling the SYS FFC Status command (see 4.4.29) to verify the system is ready to process a new command (System Ready returned).
3. Writing data to send to the camera if required into the DATA Registers or block Data buffer.
4. Writing the number of data words written (16-bit data words) to the Data Length Register.
5. Writing the desired command ID to the Command Register.
6. Polling the Status Register to determine when the command is completed (busy bit cleared).
7. Read the success code from the status register.
8. 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.

2.1.1 I2C Command Interface Start-Up

The I2C interface in the Lepton camera becomes available for Host access after the camera is brought out of reset or power cycling. It is imperative that the Host does not attempt to read or write this interface until it is available to avoid the camera from blocking further access attempts. The Host must wait a minimum of 950 milliseconds after releasing RESET_L.

2.1.1.1 Start-up command sequence

The Lepton camera requires power to be applied first, clocks applied, then RESET_L de-asserted. After this, the Host must wait a minimum of 950 milliseconds before attempting to access the I2C interface. When accessing the I2C interface, the following sequence is recommended:

a. Wait 950 milliseconds minimum after power on, clocks applied, and RESET_L de-asserted
   - If the camera has an attached shutter, the minimum wait period should be extended to 5 seconds to allow the camera's automatic execution of a flat-field correction (Auto FFC mode).

b. Read the STATUS register (register address 0x0002) bit 2
   - If Bit 2 is 0, then the camera has not booted yet, extend the wait period.
   - If Bit 2 is 1, then the camera has booted, I2C interface is available

c. Read the STATUS register (register address 0x0002) bit 0
   - If Bit 0 is 1, then the interface is busy, poll until Bit 0 becomes 0
   - If Bit 0 is 0, then the interface is ready for receiving commands.

d. Poll the SYS FFC Status command (see 4.4.29), to verify the system is ready to process a new command (System Ready returned).

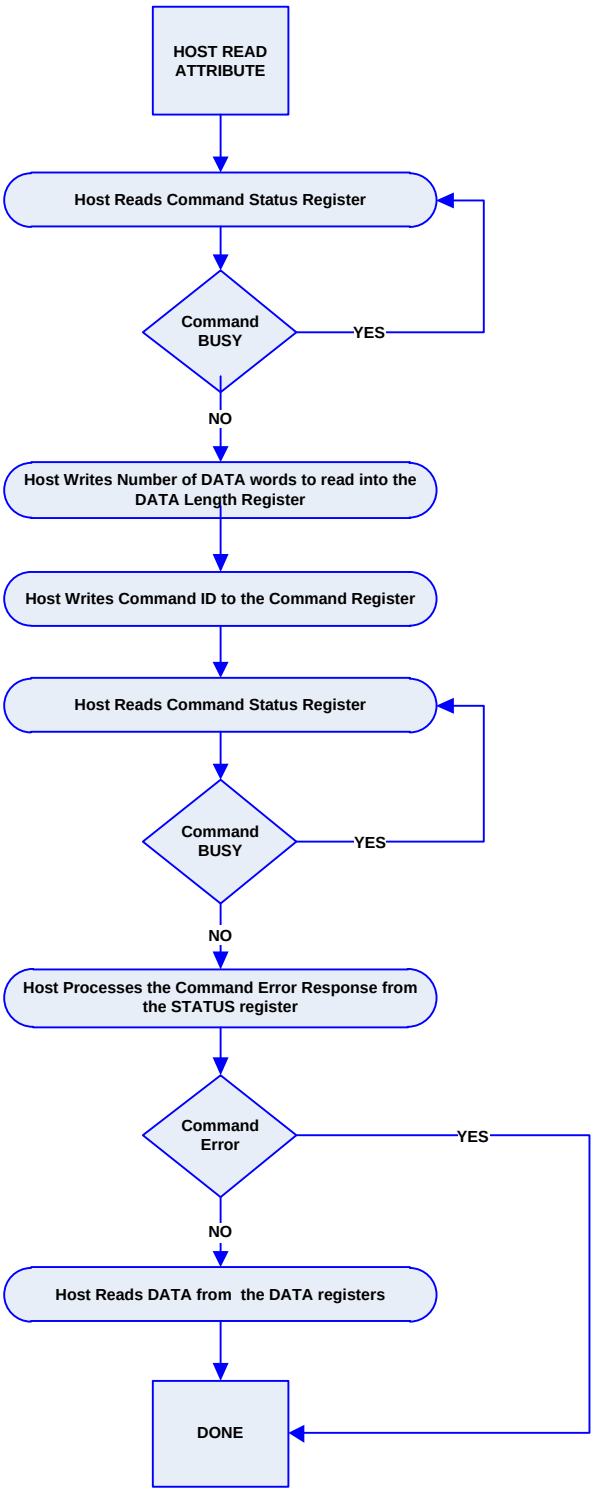

Figure 2 Lepton CCI/TWI Get or Read Attribute Sequence

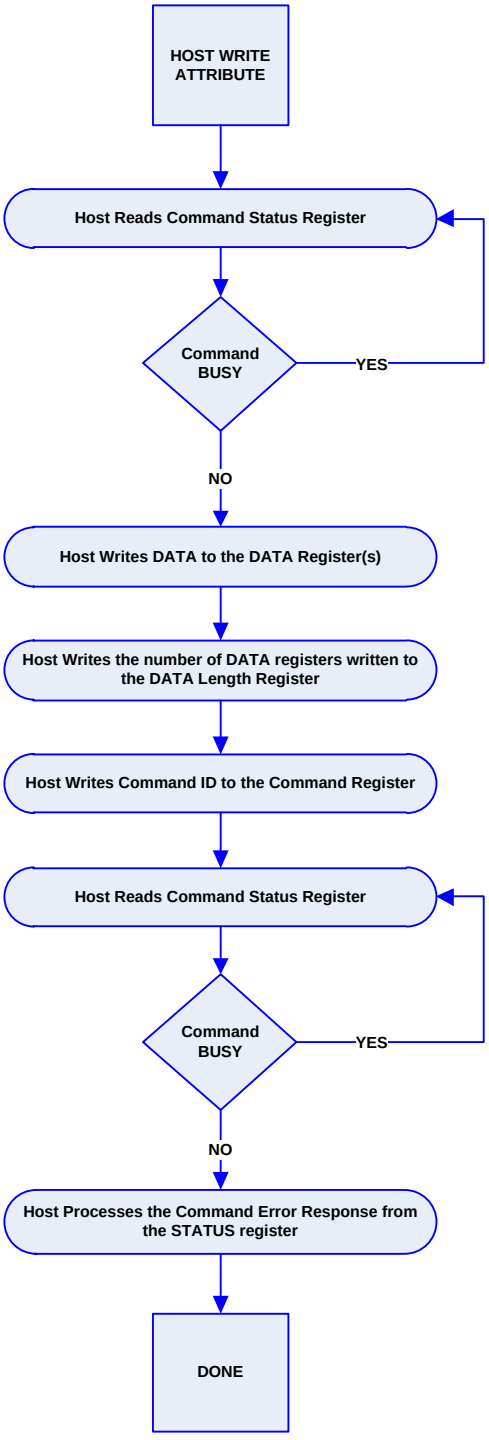

Figure 3 Lepton CCI/TWI Set or Write Sequence

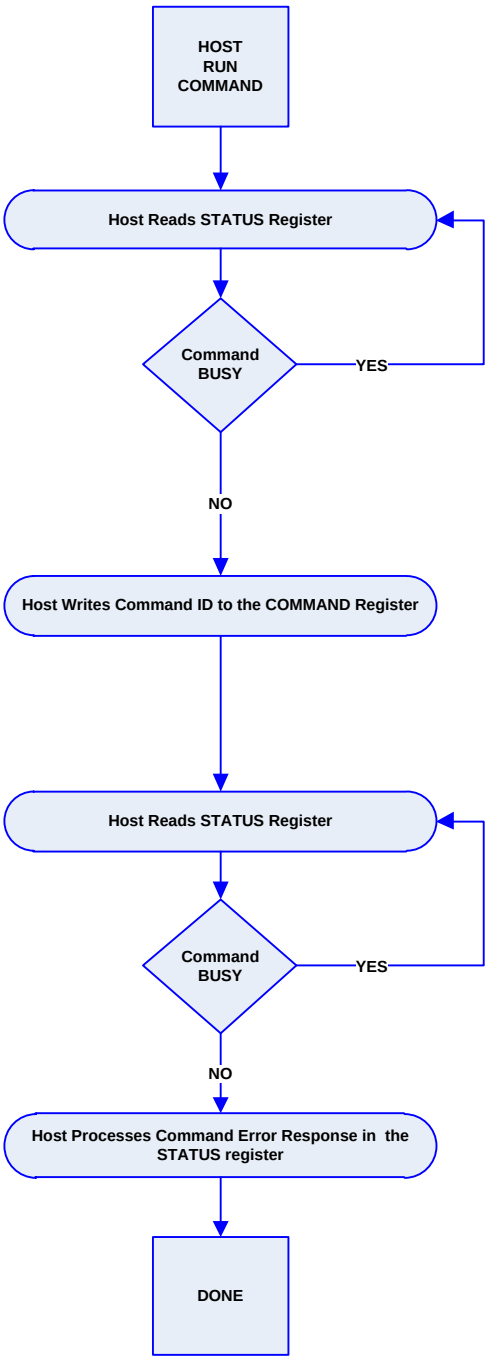

Figure 4 Lepton CCI/TWI Run Command Sequence

2.1.2 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.

2.1.2.1 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

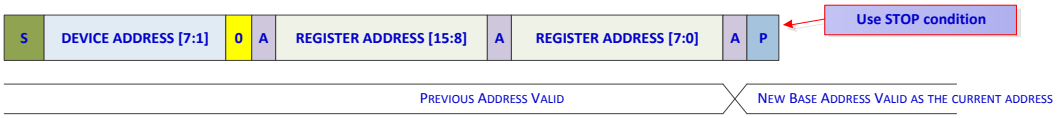

Figure 6 CCI/TWI Setting the camera's CCI/TWI current address

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

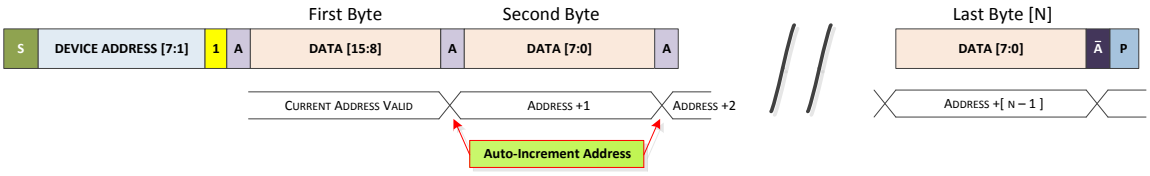

Figure 7 CCI/TWI Reading sequentially from the camera's CCI/TWI current address

2.1.2.2 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

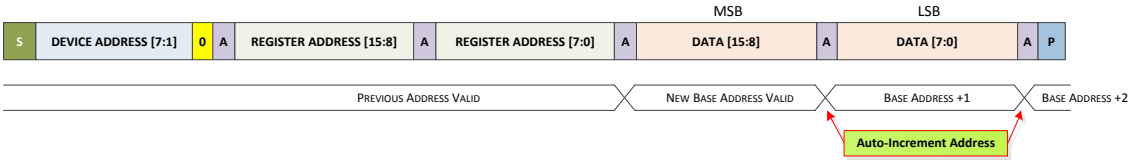

Figure 8 CCI/TWI Single WRITE to random location writes 16-bit DATA

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

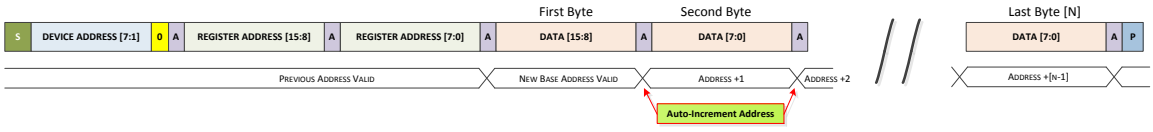

Figure 9 CCI/TWI Writing sequentially

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

2.1.3.1 Protection Bits

Certain commands require the setting of an associated protection bit such as RAD and OEM commands, because if inadvertently or incorrectly called, may compromise the camera operation.The camera uses these protection bits to verify that the camera is in the proper mode to accept these commands. If the camera is not in the correct mode, the command will not execute and the return code will indicate an invalid command. When executing the OEM and RAD interfaces, it is required that the OEM Bit (bit 14) is also set in the command register.

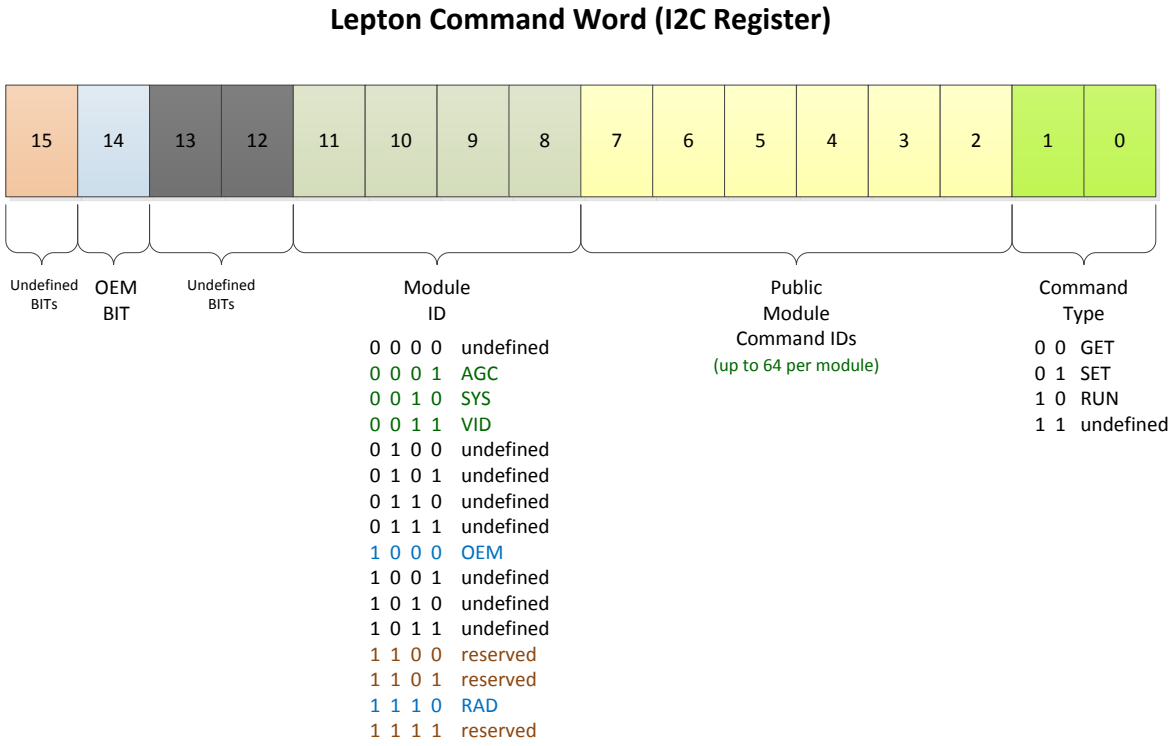

Figure 10 Lepton Command Word Format

2.1.3.2 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 five 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
- OEM – camera configuration for OEM customers
- RAD – Radiometry

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

2.1.3.3 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.

2.1.3.4 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

2.1.4 CCI/TWI Power ON Register

The Power ON register islocated at Register Address 0x0000, and is used to turn the camera ON (only if the camera was previously turned OFF by software command). To turn the camera ON, write a ZERO (0x0000) to this register. See 4.6.2.

2.1.5 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.

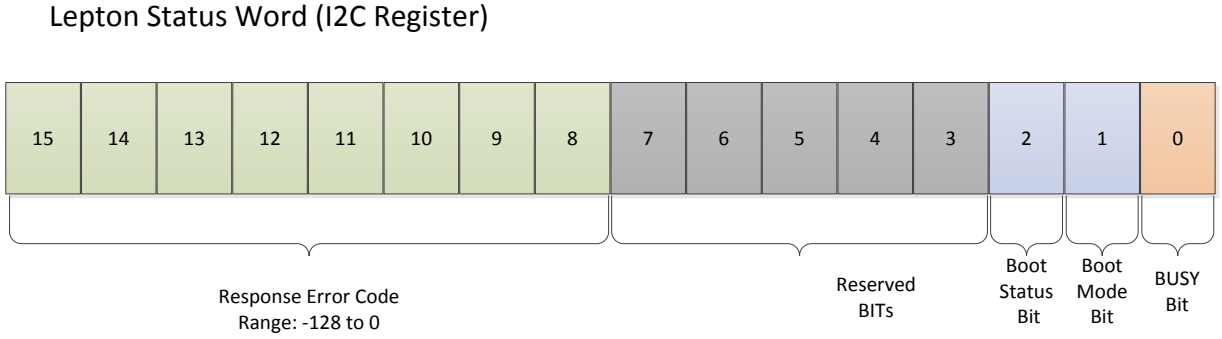

Figure 11 CCI/TWI Status Register Definition

2.1.5.1 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.

2.1.5.2 Boot Mode Bit (Bit 1)

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

2.1.6 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).

2.1.7 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.

2.1.8 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 15:8 contain the MSB and bits 7:0 contain the LSB.

2.1.8.1 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.

2.1.8.2 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
-----------------------------
00101010000010000100000
001000100000100001
-----------------------------
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
{
   LEP_OK                          = 0,     /* Camera ok */
   LEP_COMM_OK                     = LEP_OK, /* Camera comm ok (same as LEP_OK) */

   LEP_ERROR                       = -1,    /* Camera general error */
   LEP_NOT_READY                   = -2,    /* Camera not ready error */
   LEP_RANGE_ERROR                 = -3,    /* Camera range error */
   LEP_CHECKSUM_ERROR              = -4,    /* Camera checksum error */
   LEP_BAD_ARG_POINTER_ERROR       = -5,    /* Camera Bad argument  error */
   LEP_DATA_SIZE_ERROR             = -6,    /* Camera byte count error */
   LEP_UNDEFINED_FUNCTION_ERROR    = -7,    /* Camera undefined function error */
   LEP_FUNCTION_NOT_SUPPORTED      = -8,    /* Camera function not yet supported error */
   LEP_DATA_OUT_OF_RANGE_ERROR     = -9,    /* Camera input DATA is out of valid range error */
   LEP_COMMAND_NOT_ALLOWED         = -11,   /* Camera unable to execute command due to current camera
state */

   /* OTP access errors */
   LEP_OTP_WRITE_ERROR             = -15,   /*!< Camera OTP write error */
   LEP_OTP_READ_ERROR               = -16,   /* double bit error detected (uncorrectible) */

   LEP_OTP_NOT_PROGRAMMED_ERROR    = -18,   /* Flag read as non-zero */

   /* I2C Errors */
   LEP_ERROR_I2C_BUS_NOT_READY     = -20,   /* I2C Bus Error - Bus Not Avaialble */
   LEP_ERROR_I2C_BUFFER_OVERFLOW   = -22,   /* I2C Bus Error - Buffer Overflow */
   LEP_ERROR_I2C_ARBITRATION_LOST  = -23,   /* I2C Bus Error - Bus Arbitration Lost */
   LEP_ERROR_I2C_BUS_ERROR         = -24,   /* I2C Bus Error - General Bus Error */
   LEP_ERROR_I2C_NACK_RECEIVED     = -25,   /* I2C Bus Error - NACK Received */
   LEP_ERROR_I2C_FAIL              = -26,   /* I2C Bus Error - General Failure */

   /* Processing Errors */
   LEP_DIV_ZERO_ERROR              = -80,   /* Attempted div by zero */

   /* Comm Errors */
   LEP_COMM_PORT_NOT_OPEN          = -101,  /* Comm port not open */
   LEP_COMM_INVALID_PORT_ERROR     = -102,  /* Comm port no such port error */
   LEP_COMM_RANGE_ERROR            = -103,  /* Comm port range error */
   LEP_ERROR_CREATING_COMM         = -104,  /* Error creating comm */
   LEP_ERROR_STARTING_COMM         = -105,  /* Error starting comm */
   LEP_ERROR_CLOSING_COMM          = -106,  /* Error closing comm */
   LEP_COMM_CHECKSUM_ERROR         = -107,  /* Comm checksum error */
   LEP_COMM_NO_DEV                 = -108,  /* No comm device */
   LEP_TIMEOUT_ERROR               = -109,  /* Comm timeout error */
   LEP_COMM_ERROR_WRITING_COMM     = -110,  /* Error writing comm */
   LEP_COMM_ERROR_READING_COMM     = -111,  /* Error reading comm */
   LEP_COMM_COUNT_ERROR            = -112,  /* Comm byte count error */

   /* Other Errors */
   LEP_OPERATION_CANCELED          = -126,  /* Camera operation canceled */
   LEP_UNDEFINED_ERROR_CODE        = -127   /* Undefined error */

} LEP_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
0x800	OEM	OEM System Configuration
0xE00	RAD	Radiometry Module

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

1. Protection Bit –OEM.
2. A Module ID designating which camera subsystem to access (see Table 2),
3. A Command ID that specifies a unique element or command base, for that subsystem
4. 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.

4.3.2 OEM and RAD Module Command ID Generation

OEM and Radiometry (RAD) modules require a protection bit to be set before the camera will recognize it as a valid command. This ensures that these commands cannot be sent accidentally. The OEM/RAD protection bit is Bit-14 or referenced as 0x4000. This must be added to the Command ID generated as above. Thus, to power down the camera, the Command ID is synthesized as follows: Module ID + Command ID Base + Type + Protection Bit value = Command ID. For example the OEM module ID is 0x800, the Command Base ID for power down is 0x00, and we want to do a run command so the command type is 0x2, and the Protection Bit is 0x4000. So in this example, 0x0800 + 0x00 + 0x2 + 0x4000= 0x4802 and this is the LEP_RunOemPowerDown() Command ID.

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.

Note that the Focus metric (see 4.5.3) must be disabled when AGC is 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 **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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetAgcEnableState()`	Updates `agcEnableStatePtr` with the Camera's current AGC enable state.
All Lepton Configurations	`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 Policy Select

The camera supports 2 AGC policies to process incoming video data, histogram equalization (HEQ) and linear histogram stretch. This command sets and retrieves the AGC policy.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_AGC_LINEAR	LEP_AGC_HEQ	LEP_AGC_HEQ	N/A	N/A

SDK Module ID: AGC **0x0100**

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetAgcPolicy()`	Updates `agcPolicyPtr` with the Camera's current AGC policy.
All Lepton Configurations	`LEP_SetAgcPolicy()`	Sets Camera's current AGC policy to `agcPolicy`.

C SDK Interface:

```
LEP_RESULT LEP_GetAgcPolicy(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                            LEP_AGC_POLICY_E_PTR agcPolicyPtr)

LEP_RESULT LEP_SetAgcPolicy(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                            LEP_AGC_POLICY_E agcPolicy)

/* AGC Policy Enum
*/
typedef enum LEP_AGC_POLICY_TAG
{
    LEP_AGC_LINEAR=0,
    LEP_AGC_HEQ,
    LEP_END_AGC_POLICY

}LEP_AGC_POLICY_E, *LEP_AGC_POLICY_E_PTR;
```

4.4.3 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.

Lepton, Lepton with Radiometry

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

Lepton 3

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	159	159	pixels	1
end row	>= startRow	119	119	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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetAgcROI()`	Updates `agcROIPtr` with the Camera's current AGC ROI
All Lepton Configurations	`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;
```

4.4.4 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.

Lepton, Lepton with Radiometry

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	4,800	pixels	1

Lepton 3

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	19,200	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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`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;
```

4.4.5 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetAgcHeqDampingFactor()`	Updates `agcHeqDampingFactorPtr` with the Camera's current HEQ dampening factor
All Lepton Configurations	`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)
```

4.4.6 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.

Lepton, Lepton with Radiometry

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	4,800	4,800	pixels	1

Lepton 3

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	19,200	19,200	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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetAgcHeqClipLimitHigh()`	Updates `agcHeqClipLimitHighPtr` with the Camera's current HEQ level high value
All Lepton Configurations	`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)
```

4.4.7 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. This command is deprecated for Lepton 3 (replaced by Linear Percent).

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

Compatibility	C-SDK Commands	Description
Lepton, Lepton with Radiometry	`LEP_GetAgcHeqClipLimitLow()`	Updates `agcHeqClipLimitLowPtr` with the Camera's current HEQ level Low value
Lepton, Lepton with Radiometry	`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)
```

4.4.8 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

Compatibility	C-SDK Commands	Description
Lepton, Lepton with Radiometry	`LEP_GetAgcHeqEmptyCount()`	Updates `emptyCountPtr` with the Camera's current HEQ transfer function's bin empty count
All Lepton Configurations	`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)
```

4.4.9 AGC HEQ Output Scale Factor

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

Compatibility	C-SDK Commands	Description
Lepton, Lepton with Radiometry	`LEP_GetAgcHeqScaleFactor()`	Updates `scaleFactorPtr` with the Camera's current AGC HEQ Output Scale Factor
Lepton, Lepton with Radiometry	`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;
```

4.4.10 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 on an **enum** data type on a 32-bit machine
Set **2** size on an **enum** data type on a 32-bit machine

Compatibility	C-SDK Commands	Description
Lepton, Lepton with Radiometry	`LEP_GetAgcCalcEnableState()`	Updates `agcCalculationEnableStatePtr` with the Camera's current AGC Calculation enable state
Lepton, Lepton with Radiometry	`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;
```

4.4.11 AGC HEQ Linear Percent

This parameter controls the camera AGC HEQ algorithm's linear percent. The linear percent parameter fills holes in the histogram with pixels to avoid undesirable compression in 8-bit irradiance levels. Similar to the low clip limit, pixels are added to each full bin and to an additional number of empty bins following full bins. The linear percent parameter is more automatic than allowing the user to specify the low clip limit; the linear percent adjusts low clip limit based on the scene content and the desired percentage of total pixels in the histogram.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	100	20	percent	1

SDK Module ID: AGC **0x0100**

SDK Command ID: Base **0x4C**
With Get **0x4C**
With Set **0x4D**

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

Compatibility	C-SDK Commands	Description
Lepton 3	**`LEP_GetAgcHeqLinearPercent()`**	Updates **`agcHeqLinearPercentPtr`** with the Camera's current AGC HEQ Linear Percent value
Lepton 3	`LEP_SetAgcHeqLinearPercent()`	Sets Camera's current AGC HEQ Linear Percent to **`agcHeqLinearPercent`**

C SDK Interface:

```
LEP_RESULT LEP_GetAgcHeqLinearPercent( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_UINT16 *agcHeqLinearPercentPtr)

LEP_RESULT LEP_SetAgcHeqLinearPercent( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_UINT16 agcHeqLinearPercent)
```

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.

4.4.12 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`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);
```

4.4.13 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**

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**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;
```

4.4.14 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**

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**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;
```

4.4.15 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`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)
```

4.4.16 SYS AUX Temperature Kelvin

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

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**

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**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;
```

4.4.17 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**

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**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;
```

4.4.18 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
Set **2** size of an Enum on a 32-bit machine

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**LEP_ GetSysTelemetryEnableState()**	Returns the Lepton Camera's Telemetry Enable State
All Lepton Configurations	**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;
```

4.4.19 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
Set **2** size of an Enum on a 32-bit machine

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**LEP_ GetSysTelemetryLocation()**	Returns the location of Telemetry data as an enum
All Lepton Configurations	**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;
```

4.4.20 SYS Frame Average

This command executes the average frames command. Executing this command causes the camera to sum together a number of frames, divide the summed frame by the number of frames summed and generate a result frame containing the average of the summed frames.

For Lepton and Lepton with Radiometry, the number of frames to average is set by LEP_SYS_SetFramesToAverage(). For Lepton 3, the number of frames is currently fixed at 8.

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x20**
With Run **0x22**

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_RunFrameAverage()`	Executes the frame average command. The number of frames to average is set by separate command: LEP_SYS_SetFramesToAverage().

C SDK Interface:

```
LEP_RESULT LEP_RunFrameAverage(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr);
```

4.4.21 SYS Number of Frames to Average

This command Gets or Sets the number of frames to average when executing either a frame Average command (see 4.4.20) or a Flat-Field Correction (FFC) (see 4.6.12).

For Lepton and Lepton with Radiometry, the number of frames to average is set by LEP_SYS_SetFramesToAverage(). For Lepton 3, the number of frames is currently fixed at 8.

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**LEP_SYS_GetFramesToAverage()**	Gets the number of frames to average
Lepton, Lepton with Radiometry	**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;
```

4.4.22 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**`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;
```

4.4.23 SYS Camera Video Scene Statistics

This command returns the current scene statistics for the video frame defined by the SYS ROI (see section 4.4.24). 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 depends upon which camera, see below tables. When TLinear mode is enabled (available in the Lepton with Radiometry release), the camera output represents temperature values, and the scene statistics are reported in Kelvin x 100.

Lepton, Lepton with Radiometry

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	4,800	pixels	1

Lepton 3

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	19,200	pixels	1

SDK Module ID: SYS **0x0200**

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

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`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;
```

4.4.24 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 4.4.23). 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.

Lepton, Lepton with Radiometry

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

Lepton 3

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	159	159	pixels	1
end row	> start row	119	119	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_SYS_VIDEO_ROI_T` data type
Set **4** size of `LEP_SYS_VIDEO_ROI_T` data type

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetSysSceneRoi()`	Updates `sceneRoiPtr` with the Camera's current Scene ROI
All Lepton Configurations	`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;
```

4.4.25 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`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
```

4.4.26 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetShutterPosition()`	Updates `shutterPositionPtr` with the Camera's attached shutter current position.
All Lepton Configurations	`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;
```

4.4.27 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.

Dimension	Minimum Value	Maximum Value	Default Value	Lepton with Radiometry Factory Default
shutterMode	LEP_SYS_FFC_SHUTTER _MODE_MANUAL	LEP_SYS_FFC_SHUTTER _MODE_EXTERNAL	LEP_SYS_FFC_SHUTTER _MODE_EXTERNAL*	LEP_SYS_FFC_SHUTTER _MODE_AUTO
shutterLockout tempLockoutState	LEP_SYS_SHUTTER _LOCKOUT_INACTIVE	LEP_SYS_SHUTTER _LOCKOUT_LOW	LEP_SYS_SHUTTER _LOCKOUT_INACTIVE	LEP_SYS_SHUTTER _LOCKOUT_INACTIVE
videoFreezeDuringFFC	LEP_SYS_DISABLE	LEP_SYS_ENABLE	LEP_SYS_ENABLE	LEP_SYS_ENABLE
ffcDesired	LEP_SYS_DISABLE	LEP_SYS_ENABLE	LEP_SYS_DISABLE	LEP_SYS_DISABLE

* If the camera contains an internal shutter, then the default mode is `LEP_SYS_FFC_SHUTTER_MODE_AUTO`

Dimension	Minimum Value	Maximum Value	Software Value	Lepton with Radiometry Factory Default	Units	Scale
desiredFfcPeriod	0	4294967295	300000	180000	milliseconds	1
desiredFfcTempDelta	0	65535	300	150	Kelvin	100
imminentDelay	0	65535	52	52	frames	1
closePeriodInFrames	0	65535	4	4	frames	1
openPeriodInFrames	0	65535	1	1	frames	1
explicitCmdToOpen	0 (false)	1 (true)	0 (false)	0 (false)	N/A	1
elapsedTimeSinceLastFfc	0	4294967295	0	0	milliseconds	1

SDK Module ID: VID **0x0200**

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

SDK Data Length: Get **16** size of `LEP_SYS_FFC_SHUTTER_MODE_OBJ_T` data type
Set **16** size of `LEP_SYS_FFC_SHUTTER_MODE_OBJ_T` data type

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**LEP_GetFfcShutterModeObj()**	Updates **shutterModeObjPtr** with the Camera's current FFC mode and shutter control
All Lepton Configurations	**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;
```

4.4.28 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_RunSysFFCNormalization()`	Executes the FFC command.

C SDK Interface:

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

4.4.29 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**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;
```

4.4.30 SYS Gain Mode

This command sets the gain state of the camera. High gain mode provides higher responsivity and lower noise metrics for normal operation (default). Low gain mode provides lower responsivity and higher noise metrics, but with the benefit of increased intra-scene range necessary to view hotter scenes. Auto gain mode allows the camera to automatically switch the gain mode based on the temperature of the scene and thresholds configurable by the user in the "SYS Gain Mode Object" command. Auto gain mode can only be enabled when radiometry mode is enabled due to the temperature measurement dependency.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_SYS_GAIN_MODE_HIGH	LEP_SYS_GAIN_MODE_AUTO	LEP_SYS_GAIN_MODE_HIGH	N/A	N/A

SDK Module ID: SYS **0x0200**

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

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry	**LEP_GetSysGainMode()**	Returns the Lepton Camera's current Gain mode in **gainModePtr**
Lepton with Radiometry	**LEP_SetSysGainMode()**	Sets the Lepton Camera's current Gain mode to **gainMode**

C SDK Interface:

```
LEP_RESULT LEP_GetSysGainMode( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                               LEP_SYS_GAIN_MODE_E_PTR gainModePtr)

LEP_RESULT LEP_SetSysGainMode( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                               LEP_SYS_GAIN_MODE_E gainMode)
```

```
typedef enum LEP_SYS_GAIN_MODE_E_TAG
{
   LEP_SYS_GAIN_MODE_HIGH = 0,
   LEP_SYS_GAIN_MODE_LOW,
   LEP_SYS_GAIN_MODE_AUTO,

   LEP_SYS_END_GAIN_MODE,
} LEP_SYS_GAIN_MODE_E, *LEP_SYS_GAIN_MODE_E_PTR;;
```

4.4.31 SYS FFC States

This command gets the current FFC state in the camera.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_SYS_FFC_NEVER_COMMANDED	LEP_SYS_FFC_IN_PROCESS	LEP_SYS_FFC_NEVER_COMMANDED	N/A	N/A

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x4C**
With Get **0x4C**
With Set **0x4D**

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

Compatibility	C-SDK Commands	Description
Lepton 3, Lepton with Radiometry	`LEP_GetSysFFCStates()`	Returns the Lepton Camera's current FFC state `ffcStatePtr`

C SDK Interface:

```
LEP_RESULT LEP_GetSysFFCStates( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                LEP_SYS_FFC_STATES_E_PTR ffcStatePtr )
```

```
/* SYS FFC States Enum
   Captures the current camera FFC operation state
*/
typedef enum LEP_SYS_FFC_STATES_E_TAG
{
   LEP_SYS_FFC_NEVER_COMMANDED = 0,
   LEP_SYS_FFC_IMMINENT,
   LEP_SYS_FFC_IN_PROCESS,
   LEP_SYS_FFC_DONE,

   LEP_SYS_END_FFC_STATES

}LEP_SYS_FFC_STATES_E, *LEP_SYS_FFC_STATES_E_PTR;
```

4.4.32 SYS Gain Mode Object

This command gets or sets the gain mode object. Set the ROI, temperature thresholds (in Celsius for TLinear disabled mode and Kelvin for TLinear enabled mode), and percentages of the ROI that must meet the temperature threshold criteria for both high to low and low to high automatic gain mode switching.

Field	Minimum Value	Maximum Value	Default Setting	Lepton with Radiometry Factory Default	Units	Scale factor
sysGainModeROI	0,0,0,0	59,59,79,79	0,0,59,79	0,0,59,79	Pixels	1
sys_P_high_to_low	0	100	20	25	Percent	1
sys_P_low_to_high	0	100	95	90	Percent	1
sys_C_high_to_low	0	600	110	115	Celsius	1
sys_C_low_to_high	0	600	90	85	Celsius	1
sys_T_high_to_low	0	900	383	388	Kelvin	1
sys_T_low_to_high	0	900	363	358	Kelvin	1
sysGainRoiPopulation	0	4800	4800	4800	Pixels	1
sysGainModeTempEnabled	0	1	0	0	Boolean	1
sysGainModeFluxThresholdLow	0	16383	8765	Calibrated per camera	Counts	1
sysGainModeFluxThresholdHigh	0	16383	9876	Calibrated per camera	Counts	1

SDK Module ID: SYS **0x0200**

SDK Command ID: Base **0x50**
With Get **0x50**
With Set **0x51**

SDK Data Length: Get **14** size of `LEP_SYS_GAIN_MODE_OBJ_T` data type
Set **14** size of `LEP_SYS_GAIN_MODE_OBJ_T` data type

Compatibility	C-SDK Commands	Description
Lepton with Radiometry	`LEP_GetSysGainModeObj()`	Returns the Lepton Camera's current Gain mode in `gainModeObjPtr`
Lepton with Radiometry	`LEP_SetSysGainModeObj()`	Sets the Lepton Camera's current Gain mode to `gainModeObj`

C SDK Interface:

```
LEP_RESULT LEP_GetSysGainModeObj( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                  LEP_SYS_GAIN_MODE_OBJ_T_PTR gainModeObjPtr )

LEP_RESULT LEP_SetSysGainModeObj( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                  LEP_SYS_GAIN_MODE_OBJ_T gainModeObj )
```

```
/* Gain Mode Object
*/
typedef struct LEP_SYS_GAIN_MODE_OBJ_T_TAG
{
   FLR_SYS_GAIN_MODE_ROI_T          sysGainModeROI;         /* Specified ROI to use for Gain Mode switching */
   FLR_SYS_GAIN_MODE_THRESHOLDS_T   sysGainModeThresholds;  /* Set of threshold triggers */
   FLR_UINT16                       sysGainRoiPopulation;   /* Population size in pixels within the ROI */
   FLR_UINT16                       sysGainModeTempEnabled; /* True if T-Linear is implemented */
   FLR_UINT16                       sysGainModeFluxThresholdLowToHigh;  /* calculated from desired temp */
   FLR_UINT16                       sysGainModeFluxThresholdHighToLow;  /* calculated from desired temp */

}LEP SYS GAIN MODE OBJ T, *LEP SYS GAIN MODE OBJ T PTR;
```

```
/* System Gain Mode ROI Structure
*/
typedef struct LEP_SYS_GAIN_MODE_ROI_T_TAG
{
   LEP_UINT16 startCol;
   LEP_UINT16 startRow;
   LEP_UINT16 endCol;
   LEP UINT16 endRow;

}LEP_SYS_GAIN_MODE_ROI_T, *LEP_SYS_GAIN_MODE_ROI_T_PTR;
```

```
/* Gain Mode Support
*/
typedef struct LEP_SYS_GAIN_MODE_THRESHOLDS_T_TAG
{
   LEP SYS THRESHOLD T sys P high to low;    /* Range: [0 - 100], percent   */
   LEP_SYS_THRESHOLD_T sys_P_low_to_high;    /* Range: [0 - 100], percent   */

   LEP_SYS_THRESHOLD_T sys_C_high_to_low;    /* Range: [0 - 600], degrees C   */
   LEP_SYS_THRESHOLD_T sys_C_low_to_high;    /* Range: [0 - 600], degrees C   */

   LEP_SYS_THRESHOLD_T sys_T_high_to_low;    /* Range: [0 - 900], Kelvin   */
   LEP SYS THRESHOLD T sys T low to high;    /* Range: [0 - 900], Kelvin   */

}LEP_SYS_GAIN_MODE_THRESHOLDS_T, *LEP_SYS_GAIN_MODE_THRESHOLDS_T_PTR;
```

4.4.33 SYS Average Frames – *Aggregate Command*

This is an SDK aggregate command that executes the frame average command using a parameter to specify the number of frames to average.

Executing this command causes the camera to sum together a number of frames, divide the summed frame by the number of frames summed and generate a result frame containing the average of the summed frames. The number of frames to average is specified by parameter to this function.

Compatibility	C-SDK Commands	Description
Lepton	**LEP_RunSysAverageFrames()**	Executes the frame average command using the number of frames to average is specified by parameter to this function. ***Aggregate command.***

C SDK Interface:

```
LEP_RESULT LEP_RunSysAverageFrames (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;
```

4.4.34 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)

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**`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;
```

4.4.35 SYS FPA Temperature Celsius – *helper function*

This is a SDK command that returns the Lepton Camera's FPA 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)

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**`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;
```

4.5 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.

4.5.1 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 4.6.7), AGC enabled and scaled to 8-bit output (See 4.4.1).

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetVidPcolorLut()`	Updates `vidPcolorLutPtr` with the Camera's current video pseudo-color LUT selection.
All Lepton Configurations	`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;
```

4.5.2 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`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`**.
All Lepton Configurations	`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;
```

4.5.3 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 4.5.6. When disabled, the camera does not execute the focus metric calculation.

Note that AGC (See 4.4.1) must be disabled when the focus metric is enabled.

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetVidFocusCalcEnableState()`	Updates `vidEnableFocusCalcStatePtr` with the Camera's current video focus calculation enable state.
All Lepton Configurations	`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;
```

4.5.4 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.

Lepton, Lepton with Radiometry

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

Lepton 3

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	158	158	pixels	1
end row	> start row+1	118	118	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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetVidFocusROI()`	Updates `vidFocusROIPtr` with the Camera's current video focus ROI
All Lepton Configurations	`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;
```

4.5.5 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetVidFocusMetricThreshold()`	Updates `vidFocusMetricThresholdPtr` with the Camera's current video focus metric threshold.
All Lepton Configurations	`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;
```

4.5.6 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**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;
```

4.5.7 VID Video Freeze Enable State

This function allows the current frame to be repeated in lieu of a live video stream. When enabled, live video is halted from the camera. 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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetVidFreezeEnableState()`	Updates `vidFreezeEnableStatePtr` with the Camera's current Video Freeze enable state
All Lepton Configurations	`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 ;
```

4.5.8 VID Video Output Format

This function provides the method to specify or retrieve the current video output format. In the current revision, only RGB888 and RAW14 are valid formats.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_VID_VIDEO_OUTPUT_FORMAT_RAW8	LEP_VID_VIDEO_OUTPUT_FORMAT_RAW8_6	LEP_VID_VIDEO_OUTPUT_FORMAT_RAW14	N/A	N/A

SDK Module ID: VID **0x0300**

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

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry, Lepton 3	`LEP_GetVidVideoOutputFormat()`	Updates `vidVideoOutputFormatPtr` with the Camera's current video ouput format
Lepton with Radiometry, Lepton 3	`LEP_SetVidVideoOutputFormat()`	Updates the Camera's Camera's current video ouput format with the contents of `vidVideoOutputFormat`.

C SDK Interface:

```
LEP_RESULT LEP_GetVidVideoOutputFormat( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                        LEP_VID_VIDEO_OUTPUT_FORMAT_E_PTR vidVideoOutputFormatPtr )

LEP_RESULT LEP_SetVidVideoOutputFormat( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                        LEP_VID_VIDEO_OUTPUT_FORMAT_E vidVideoOutputFormat )
```

```
/* Video Output Format
*/
typedef struct LEP_VID_VIDEO_OUTPUT_FORMAT_TAG
{
   LEP_VID_VIDEO_OUTPUT_FORMAT_RAW8 = 0,          // To be supported in later release
   LEP_VID_VIDEO_OUTPUT_FORMAT_RAW10,             // To be supported in later release
   LEP_VID_VIDEO_OUTPUT_FORMAT_RAW12,             // To be supported in later release
   LEP_VID_VIDEO_OUTPUT_FORMAT_RGB888,            // To be supported in later release
   LEP_VID_VIDEO_OUTPUT_FORMAT_RGB666,            // To be supported in later release
   LEP_VID_VIDEO_OUTPUT_FORMAT_RGB565,            // To be supported in later release
   LEP_VID_VIDEO_OUTPUT_FORMAT_YUV422_8BIT,       // To be supported in later release
```

```
    LEP_VID_VIDEO_OUTPUT_FORMAT_RAW14,              // SUPPORTED in this release
    LEP_VID_VIDEO_OUTPUT_FORMAT_YUV422_10BIT,       // To be supported in later release
    LEP_VID_VIDEO_OUTPUT_FORMAT_USER_DEFINED,       // To be supported in later release
    LEP_VID_VIDEO_OUTPUT_FORMAT_RAW8_2,             // To be supported in later release
    LEP_VID_VIDEO_OUTPUT_FORMAT_RAW8_3,             // To be supported in later release
    LEP_VID_VIDEO_OUTPUT_FORMAT_RAW8_4,             // To be supported in later release
    LEP_VID_VIDEO_OUTPUT_FORMAT_RAW8_5,             // To be supported in later release
    LEP_VID_VIDEO_OUTPUT_FORMAT_RAW8_6,             // To be supported in later release
    LEP_END_VID_VIDEO_OUTPUT_FORMAT

}LEP_VID_VIDEO_OUTPUT_FORMAT_E, *LEP_VID_VIDEO_OUTPUT_FORMAT_E_PTR;
```


4.5.9 VID Low Gain Pseudo-Color Look-Up Table Select

This function allows selection of the video output pseudo-color LUT to be used when the camera is in Low Gain Mode. 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 4.6.7), AGC enabled and scaled to 8-bit output (See 4.4.1).

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 **0x34**
With Get **0x34**
With Set **0x35**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry	`LEP_GetVidLowGainPcolorLut()`	Updates `vidPcolorLutPtr` with the Camera's current video pseudo-color LUT selection.
Lepton with Radiometry	`LEP_SetLowGainVidPcolorLut()`	Sets Camera's current video pseudo-color LUT selection to `vidPcolorLut`

C SDK Interface:

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

LEP_RESULT LEP_SetVidLowGainPcolorLut(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 SDK Module: OEM 0x800

This module provides additional camera configuration, control, information and status of the camera system. This includes more specific version information about the camera. This module also provides OEMs with filter controls, power controls, and video output configuration and control.

4.6.1 Setting the OEM Protection Bit

Issuing OEM commands requires setting the OEM Bit (Bit 14, value = 0x4000) in the command word in addition to setting the rest of the command word bits – see Figure 10. If this bit is not set to 1 for each OEM command issued, the camera will return an error code. If this bit is set when issuing any other commands that are not OEM or RAD commands, then the camera will also return an error code. See section 4.3.2 for a description of how an OEM module or RAD module command is synthesized.

4.6.2 OEM Power On

This function sends the Power On command to the camera to turn the Camera ON once the camera was shutdown using the LEP_RunOemPowerDown() command (see 4.6.3). The power ON command is executed by the SDK (not the camera) by writing a ZERO (0x0000) to the Camera I2C Device ID (see 2.1.3). This will turn the camera ON if the power down and reset pins are NOT asserted.

Note that this command is not fully supported. It works the first time, after that, a power cycle is required.

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **N/A**

SDK Data Length: Run **N/A** size a run command argument

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_RunOemPowerOn()`	Issues a Camera Power Down command to the Camera

C SDK Interface:

```
LEP_RESULT LEP_RunOemPowerOn(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr);
```

4.6.3 OEM Power Down

This function sends the Power Down command to the camera to effectively shut the Camera off. The camera will respond with LEP_OK if command received correctly and then place the Camera into a power down or OFF mode after a small delay. Power Down is identical to the Camera s mode; both place the Camera into the OFF state.

To turn the Camera back ON using software, the Host must perform the following sequence:

- Let the ASIC hold the DATA line (SDA) low
- Issue a single clock pulse. This Is required for the ASIC to de-assert the DATA line.
- Call **LEP_RunOemPowerOn()** (see 4.6.2). This function will write a ZERO (0x0000) to the Camera Device ID to turn the Camera ON.

This will bring the Camera out of Power Down.

SDK Module ID: OEM **0x0800**

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

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_RunOemPowerDown()`	Issues a Camera Power Down command to the Camera

C SDK Interface:

```
LEP_RESULT LEP_RunOemPowerDown(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr);
```

4.6.4 OEM FLIR Systems Part Number

This function returns FLIR Systems' Camera Part Number. The Camera Part Number is a 32-byte string identifier unique to a specific configuration of the Camera module.

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 **`oemPartNumberPtr`**

SDK Module ID: OEM **0x0800**

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

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**`LEP_GetOemFlirPartNumber()`**	Updates **`oemPartNumberPtr`** with the Camera's FLIR Systems Part Number.

C SDK Interface:

```
LEP_RESULT LEP_GetOemFlirPartNumber( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                     LEP_OEM_PART_NUMBER_T_PTR oemPartNumberPtr )
```

```
/* Part Number: A (32 byte string) identifier unique to a
** specific configuration of module; essentially a module
** Configuration ID.
*/
typedef LEP_CHAR8 *LEP_OEM_PART_NUMBER_T, *LEP_OEM_PART_NUMBER_T_PTR;
```

4.6.5 OEM Camera Software Revision

This function returns the Camera's software revision for both software processors in the Camera. The Camera's Software revision is composed of 3 fields: a major version, minor version, and a build number for each processor. Each of the 3 fields is 8-bits.

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x20**
With Get **0x20**

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetOemSoftwareVersion()`	Updates `oemSoftwareVersionPtr` with the Camera's Software Revision.

C SDK Interface:

```
LEP_RESULT LEP_GetOemSoftwareVersion( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                      LEP_OEM_SW_VERSION_T *oemSoftwareVersionPtr )
```

```
/* Software Version ID: A (24 bit depth) identifier for
** the software version stored in OTP.
*/
typedef struct LEP_OEM_SW_VERSION_TAG
{
   LEP_UINT8   gpp_major;
   LEP_UINT8   gpp minor;
   LEP_UINT8   gpp build;
   LEP_UINT8   dsp_major;
   LEP_UINT8   dsp_minor;
   LEP_UINT8   dsp_build;
   LEP_UINT16  reserved;

}LEP OEM SW VERSION T, *LEP OEM SW VERSION T PTR;
```

4.6.6 OEM Video Output Enable

This function enables or disables the video output independent of output channel.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_VIDEO_OUTPUT_DISABLE	LEP_VIDEO_OUTPUT_ENABLE	LEP_VIDEO_OUTPUT_ENABLE	N/A	N/A

SDK Module ID: OEM **0x0800**

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetOemVideoOutputEnable()`	Updates `oemVideoOutputEnablePtr` with the Camera's current video output enable.
All Lepton Configurations	`LEP_SetOemVideoOutputEnable()`	Updates the Camera's current video output enable with the contents of `oemVideoOutputEnable`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemVideoOutputEnable(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_OEM_VIDEO_OUTPUT_ENABLE_E_PTR
                                        oemVideoOutputEnablePtr)

LEP_RESULT LEP_SetOemVideoOutputEnable(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_OEM_VIDEO_OUTPUT_ENABLE_E
                                        oemVideoOutputEnable)
```

```
/* Video Output Enable Enum
*/
typedef enum LEP_OEM_VIDEO_OUTPUT_ENABLE_TAG
{
   LEP_VIDEO_OUTPUT_DISABLE = 0,
   LEP_VIDEO_OUTPUT_ENABLE,
   LEP_END_VIDEO_OUTPUT_ENABLE

}LEP_OEM_VIDEO_OUTPUT_ENABLE_E, *LEP_OEM_VIDEO_OUTPUT_ENABLE_E_PTR;
```

4.6.7 OEM Video Output Format Select

This function provides the method to specify or retrieve the current video output format. In the current revision, only RGB888 and RAW14 are valid formats.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_VIDEO_OUTPUT_FORMAT_RAW8	LEP_VIDEO_OUTPUT_FORMAT_RAW8_6	LEP_VIDEO_OUTPUT_FORMAT_RAW14	N/A	N/A

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x28**
With Get **0x28**
With Set **0x29**

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetOemVideoOutputFormat()`	Updates `oemVideoOutputFormatPtr` with the Camera's current video output format selection.
All Lepton Configurations	`LEP_SetOemVideoOutputFormat()`	Updates the Camera's current video output format with the contents of `oemVideoOutputFormat`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemVideoOutputFormat(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_OEM_VIDEO_OUTPUT_FORMAT_E_PTR
                                       oemVideoOutputFormatPtr)

LEP_RESULT LEP_SetOemVideoOutputFormat(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_OEM_VIDEO_OUTPUT_FORMAT_E
                                       oemVideoOutputFormat)
```

```
/* Video Output Format Selection
*/
typedef enum LEP_OEM_VIDEO_OUTPUT_FORMAT_TAG
{
   LEP_VIDEO_OUTPUT_FORMAT_RAW8 = 0,          // To be supported in later release
   LEP_VIDEO_OUTPUT_FORMAT_RAW10,             // To be supported in later release
   LEP_VIDEO_OUTPUT_FORMAT_RAW12,             // To be supported in later release
   LEP_VIDEO_OUTPUT_FORMAT_RGB888,            // SUPPORTED in this release
   LEP_VIDEO_OUTPUT_FORMAT_RGB666,            // To be supported in later release
   LEP_VIDEO_OUTPUT_FORMAT_RGB565,            // To be supported in later release
   LEP_VIDEO_OUTPUT_FORMAT_YUV422_8BIT,       // To be supported in later release
   LEP_VIDEO_OUTPUT_FORMAT_RAW14,             // SUPPORTED in this release
   LEP_VIDEO_OUTPUT_FORMAT_YUV422_10BIT,      // To be supported in later release
```

```
   LEP_VIDEO_OUTPUT_FORMAT_USER_DEFINED,      // To be supported in later release
   LEP_VIDEO_OUTPUT_FORMAT_RAW8_2,            // To be supported in later release
   LEP_VIDEO_OUTPUT_FORMAT_RAW8_3,            // To be supported in later release
   LEP_VIDEO_OUTPUT_FORMAT_RAW8_4,            // To be supported in later release
   LEP_VIDEO_OUTPUT_FORMAT_RAW8_5,            // To be supported in later release
   LEP_VIDEO_OUTPUT_FORMAT_RAW8_6,            // To be supported in later release
   LEP_END_VIDEO_OUTPUT_FORMAT

}LEP_OEM_VIDEO_OUTPUT_FORMAT_E, *LEP_OEM_VIDEO_OUTPUT_FORMAT_E_PTR;
```


4.6.8 OEM Video Output Source Select

This function specifies or retrieves the video output source. The output source allows selecting between processed video data, unprocessed video data, and a variety of ramp patterns.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_VIDEO_OUTPUT_SOURCE_RAW	LEP_VIDEO_OUTPUT_SOURCE_FRAME_4	LEP_VIDEO_OUTPUT_SOURCE_COOKED	N/A	N/A

SDK Module ID: OEM **0x0800**

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

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetOemVideoOutputSource()`	Updates `oemVideoOutputSourcePtr` with the Camera's current video output format selection.
All Lepton Configurations	`LEP_SetOemVideoOutputSource()`	Updates the Camera's current video output format with the contents of `oemVideoOutputSource`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemVideoOutputSource(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_OEM_VIDEO_OUTPUT_SOURCE_E_PTR
                                       oemVideoOutputSourcePtr)


LEP_RESULT LEP_SetOemVideoOutputSource(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_OEM_VIDEO_OUTPUT_SOURCE_E
                                       oemVideoOutputSource)
```

```
/* Video Output Source Selection
*/
typedef enum LEP OEM VIDEO OUTPUT SOURCE TAG
{
   LEP_VIDEO_OUTPUT_SOURCE_RAW = 0,       /* Before video processing. */
   LEP_VIDEO_OUTPUT_SOURCE_COOKED,        /* Post video processing - NORMAL MODE */
   LEP_VIDEO_OUTPUT_SOURCE_RAMP,          /* Software Ramp pattern - increase in X, Y */
   LEP_VIDEO_OUTPUT_SOURCE_CONSTANT,      /* Software Constant value pattern */
   LEP_VIDEO_OUTPUT_SOURCE_RAMP_H,        /* Software Ramp pattern - increase in X only */
   LEP VIDEO OUTPUT SOURCE RAMP V,        /* Software Ramp pattern - increase in Y only */
   LEP VIDEO OUTPUT SOURCE RAMP CUSTOM,   /* Software Ramp pattern - uses custom settings */
   /* Additions to support frame averaging, freeze frame, and data buffers
   */
   LEP_VIDEO_OUTPUT_SOURCE_FRAME_CAPTURE,  // Average, Capture frame
   LEP_VIDEO_OUTPUT_SOURCE_FRAME_FREEZE,   // Freeze-Frame Buffer
   /* RESERVED BUFFERS
```

```
   */
   LEP_VIDEO_OUTPUT_SOURCE_FRAME_0,        // Reserved DATA Buffer
   LEP_VIDEO_OUTPUT_SOURCE_FRAME_1,        // Reserved DATA Buffer
   LEP VIDEO OUTPUT SOURCE FRAME 2,        // Reserved DATA Buffer
   LEP VIDEO OUTPUT SOURCE FRAME 3,        // Reserved DATA Buffer
   LEP VIDEO OUTPUT SOURCE FRAME 4,        // Reserved DATA Buffer

   LEP_END_VIDEO_OUTPUT_SOURCE

}LEP_OEM_VIDEO_OUTPUT_SOURCE_E, *LEP_OEM_VIDEO_OUTPUT_SOURCE_E_PTR;
```

4.6.9 OEM Customer Part Number

This function gets the Customer Part Number. This part number is previously written into the Camera OTP during factory calibration.

The Customer Part Number is a 32-byte string identifier unique to a specific configuration of the Camera module.

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 **`oemPartNumberPtr`**

SDK Module ID: OEM **0x0800**

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

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetOemCustPartNumber()`	Updates **`oemPartNumberPtr`** with the Camera's current Customer Part Number.

C SDK Interface:

```
LEP_RESULT LEP_GetOemCustPartNumber( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                     LEP_OEM_PART_NUMBER_T_PTR oemPartNumberPtr )
```

```
/* Part Number: A (32 byte string) identifier unique to a
** Specific configuration of module; essentially a module
** Configuration ID.
*/
typedef LEP_CHAR8 *LEP_OEM_PART_NUMBER_T, *LEP_OEM_PART_NUMBER_T_PTR;
```

4.6.10 OEM Video Output Source Constant Value

This function gets and sets the video output source constant value. This value is used when the output source selection is LEP_VIDEO_OUTPUT_SOURCE_CONSTANT. The output video source bypasses all video pipeline processing and directly affects the output video. This command is typically used to set the camera output to a known constant value for downstream system tests and calibration.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	16383	undefined	Counts	1

SDK Module ID: OEM **0x0800**

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

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetOemVideoOutputSourceConstantValue()`	Updates `oemVideoOutputSourceConstPtr` with the Camera's current Video Output Source Constant.
All Lepton Configurations	`LEP_SetOemVideoOutputSourceConstantValue()`	Updates the Camera's video output constant with the contents of `oemVideoOutputSourceConstant`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemVideoOutputSourceConstant(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                               LEP_UINT16 *oemVideoOutputSourceConstPtr)

LEP_RESULT LEP_SetOemVideoOutputSourceConstant(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                               LEP_UINT16 oemVideoOutputSourceConstant)
```

4.6.11 OEM Run Camera Re-Boot

This function commands the Camera to re-boot. The Camera is first shutdown, and then restarts automatically.

SDK Module ID: OEM **0x0800**

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

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_RunOemReboot()`	Issues a run Camera Re-Boot command to the Camera.

C SDK Interface:

```
LEP_RESULT LEP_RunOemReboot(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr);
```

4.6.12 OEM FFC Normalization Target

The first two of these commands Get and Set the Flat-Field Correction (FFC) Normalization Target used by the third command to execute a Flat-Field Correction (FFC). The target value is factory set and should not be changed under normal circumstances. The Run command executes an FFC using currently active values for the FFC normalization target and number of frames to average (see 4.4.21). This command executes synchronously. Poll the OEM Status to determine when this command completes (see 4.6.13).

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	16383	8192	N/A	N/A

SDK Module ID: SYS **0x0800**

SDK Command ID: Base **0x44**
With Get **0x44**
With Set **0x45**
With Run **0x46**

SDK Data Length: Get **1** size of a `LEP_UINT16`
Set **1** size of a `LEP_UINT16`
Run **0** size of a Run command

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetOemFFCNormalizationTarget()`	Gets the normalization target
All Lepton Configurations	`LEP_SetOemFFCNormalizationTarget()`	Sets the normalization target
All Lepton Configurations	`LEP_RunOemFFC()`	Executes the FFC normalization using previously specified normalization target value

C SDK Interface:

```
LEP_RESULT LEP_GetOemFFCNormalizationTarget( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                             LEP_OEM_FFC_NORMALIZATION_TARGET_T_PTR ffcTargetPtr )

LEP_RESULT LEP_SetOemFFCNormalizationTarget( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                             LEP_OEM_FFC_NORMALIZATION_TARGET_T ffcTarget )

LEP_RESULT  LEP_RunOemFFC( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr )

typedef LEP_UINT16  LEP_OEM_FFC_NORMALIZATION_TARGET_T, *LEP_OEM_FFC_NORMALIZATION_TARGET_T_PTR;
```

4.6.13 OEM Status

This function obtains the current status of an OEM run operation. This function is used whenever an OEM command is issued that executes an operation like the run FFC. Typically, the host polls the status to determine when the command has completed. If the return value is negative, then the operation completed with an error. Positive values indicate an in-process state. Zero indicates the operation completed without error.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_OEM_STATUS_OTP_WRITE_ERROR	LEP_OEM_FRAME_AVERAGE_COLLECTING_FRAMES	LEP_OEM_STATUS_READY	N/A	N/A

SDK Module ID: OEM **0x0800**

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

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**LEP_GetOemCalStatus()**	Gets the Current OEM operation status.

C SDK Interface:

```
LEP_RESULT LEP_GetOemCalStatus( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                LEP_OEM_STATUS_E_PTR calStatusPtr )
```

```
typedef enum
{
   LEP_OEM_STATUS_OTP_WRITE_ERROR = -2,
   LEP_OEM_STATUS_ERROR = -1,
   LEP_OEM_STATUS_READY = 0,
   LEP_OEM_STATUS_BUSY,
   LEP_OEM_FRAME_AVERAGE_COLLECTING_FRAMES,
   LEP_OEM_STATUS_END

} LEP_OEM_STATUS_E, *LEP_OEM_STATUS_E_PTR;
```

4.6.14 OEM Frame Mean Intensity

This function obtains the current frame mean intensity value within the video Region of Interest defined by SYS ROI (see 4.4.24). Note that this ROI (and the resulting mean) is not the same as that used by AGC Histogram Statistics.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	16383	N/A	N/A	N/A

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x4C**
With Get **0x4C**

SDK Data Length: Get **1** size of a `LEP_UINT16`

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetOemFrameMean()`	Gets the current frame mean intensity value within the SYS ROI

C SDK Interface:

```
LEP_RESULT LEP_GetOemFrameMean( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                LEP_OEM_FRAME_AVERAGE_T_PTR frameAveragePtr )
```

```
typedef LEP_UINT16  LEP_OEM_FRAME_AVERAGE_T, *LEP_OEM_FRAME_AVERAGE_T_PTR;
```

4.6.15 OEM GPIO Mode Select

This function gets and sets the GPIO pins mode.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_OEM_GPIO_MODE_GPIO	LEP_OEM_GPIO_MODE_VSYNC	LEP_OEM_GPIO_MODE_GPIO	N/A	1

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x54**
With Get **0x54**
With Set **0x55**

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetOemGpioMode()`	Updates `gpioModePtr` with the Camera's current GPIO pins mode.
All Lepton Configurations	`LEP_SetOemGpioMode()`	Updates the Camera's GPIO pins mode with the contents of `gpioMode`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemGpioMode( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                               LEP_OEM_GPIO_MODE_E_PTR gpioModePtr )

LEP_RESULT LEP_SetOemGpioMode( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                               LEP_OEM_GPIO_MODE_E gpioMode )
```

```
typedef enum LEP_OEM_GPIO_MODE_E_TAG
{
   LEP_OEM_GPIO_MODE_GPIO = 0,
   LEP_OEM_GPIO_MODE_I2C_MASTER = 1,
   LEP_OEM_GPIO_MODE_SPI_MASTER_VLB_DATA = 2,
   LEP_OEM_GPIO_MODE_SPIO_MASTER_REG_DATA = 3,
   LEP_OEM_GPIO_MODE_SPI_SLAVE_VLB_DATA = 4,
   LEP_OEM_GPIO_MODE_VSYNC = 5,

   LEP_OEM_END_GPIO_MODE,
}LEP_OEM_GPIO_MODE_E, *LEP_OEM_GPIO_MODE_E_PTR;
```

4.6.16 OEM GPIO VSync Phase Delay

This function gets and sets the GPIO VSync phase delay. The Lepton Camera can issue a pulse on GPIO3 when there is an inter VSync. The output pulse may be issued in phase with the camera's internal VSync, or it may be issued earlier or later. This command controls this phase relationship. The delays are in line periods, approximately 0.5 milliseconds per period. The phase delay is limited to +/- 3 line periods.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_OEM_VSYNC_DELAY_MINUS_3	LEP_OEM_VSYNC_DELAY_PLUS_3	LEP_OEM_VSYNC_DELAY_NONE	N/A	N/A

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x58**
With Get **0x58**
With Set **0x59**

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetOemGpioVsyncPhaseDelay()`	Updates `numHsyncLinesPtr` with the Camera's current GPIO VSync phase delay.
All Lepton Configurations	`LEP_SetOemGpioVsyncPhaseDelay()`	Updates the Camera's GPIO VSync phase delay with the contents of `numHsyncLines`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemGpioVsyncPhaseDelay( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                          LEP_OEM_VSYNC_DELAY_E_PTR numHsyncLinesPtr )

LEP_RESULT LEP_SetOemGpioVsyncPhaseDelay( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                          LEP_OEM_VSYNC_DELAY_E numHsyncLines )
```

```
typedef enum LEP_OEM_VSYNC_DELAY_E_TAG
{
   LEP_OEM_VSYNC_DELAY_MINUS_3 = -3,
   LEP_OEM_VSYNC_DELAY_MINUS_2 = -2,
   LEP_OEM_VSYNC_DELAY_MINUS_1 = -1,
   LEP_OEM_VSYNC_DELAY_NONE = 0,
```

```
    LEP_OEM_VSYNC_DELAY_PLUS_1 = 1,
    LEP_OEM_VSYNC_DELAY_PLUS_2 = 2,
    LEP_OEM_VSYNC_DELAY_PLUS_3 = 3,

    LEP_END_OEM_VSYNC_DELAY
} LEP_OEM_VSYNC_DELAY_E, *LEP_OEM_VSYNC_DELAY_E_PTR;
```

4.6.17 OEM User Defaults

The camera supports the ability allow an OEM to save certain camera runtime states to OTP for persistent storage and automatic restore upon camera startup. The host can interrogate the camera to determine if the OEM default values were written to OTP or not using the LEP_GetOemUserDefaultsState API. The Host can also command the Camera to write the current camera values into OTP for automatic restore at camera startup using the LEP_RunOemUserDefaultsCopyToOtp API. The VPROG voltage must be set in order to write the user defaults to OTP successfully.

Coordinate	Minimum Value	Maximum Value	Default Value	Units	Scale factor
row	0	119	N/A	N/A	1
col	0	159	N/A	N/A	1

SDK Module ID: VID **0x800**

SDK Command ID: Base **0x5C**
With Get **0x5C**
With Run **0x5E**

SDK Data Length: Get **2** size of an Enum on a 32-bit machine
Run **0** size a run command argument

Compatibility	C-SDK Commands	Description
Lepton with Radiometry, Lepton 3	`LEP_GetOemUserDefaultsState()`	Updates `userParamsStatePtr` with the Camera's current OEM OTP user defaults written sate
Lepton with Radiometry, Lepton 3	`LEP_RunOemUserDefaultsCopyToOtp()`	Executes writing the OEM user defaults to OTP

C SDK Interface:

```
LEP_RESULT LEP_GetOemUserDefaultsState(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_OEM_USER_PARAMS_STATE_E_PTR userParamsStatePtr)

LEP_RESULT LEP_RunOemUserDefaultsCopyToOtp(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr)

/* OEM User Defaults State
*/
typedef enum LEP OEM USER PARAMS STATE E TAG
{
   LEP_OEM_USER_PARAMS_STATE_NOT_WRITTEN = 0,
```

```
    LEP_OEM_USER_PARAMS_STATE_WRITTEN,
    LEP_OEM_END_USER_PARAMS_STATE,
}LEP_OEM_USER_PARAMS_STATE_E, *LEP_OEM_USER_PARAMS_STATE_E_PTR;
```


4.6.18 OEM Restore User Defaults

This function will restore the OEM user defaults from OTP if OTP was previously written with these defaults. If user defaults were not previously written, an error code is returned.

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x60**
With Run **0x62**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry, Lepton 3	`LEP_RunOemUserDefaultsRestore()`	Restore the OEM user defaults from OTP if OTP was previously written with these defaults

C SDK Interface:

```
LEP_RESULT LEP_RunOemUserDefaultsRestore(LEP_CAMERA_PORT_DESC_T_PTR portDescPtr);
```

4.6.19 OEM Shutter Profile

This function gets and sets the shutter profile.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_OEM_GPIO_MODE_GPIO	LEP_OEM_GPIO_MODE_VSYNC	LEP_OEM_GPIO_MODE_GPIO	N/A	1

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x64**
With Get **0x64**
With Set **0x65**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry, Lepton 3	`LEP_GetOemShutterProfileObj()`	Updates `ShutterProfileObjPtr` with the Camera's current shutter profile
Lepton with Radiometry, Lepton 3	`LEP_SetOemShutterProfileObj()`	Updates the Camera's shutter profile with the contents of `ShutterProfileObj`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemShutterProfileObj( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                        LEP_OEM_SHUTTER_PROFILE_OBJ_T_PTR ShutterProfileObjPtr )

LEP_RESULT LEP_SetOemShutterProfileObj( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                        LEP_OEM_SHUTTER_PROFILE_OBJ_T ShutterProfileObj )

/* Shutter Profile Object
*/
typedef struct LEP_OEM_SHUTTER_PROFILE_OBJ_T_TAG
{
   LEP_UINT16 closePeriodInFrames;          /* in frame counts x1 */
   LEP_UINT16 openPeriodInFrames;           /* in frame counts x1 */

}LEP_OEM_SHUTTER_PROFILE_OBJ_T, *LEP_OEM_SHUTTER_PROFILE_OBJ_T_PTR;
```

4.6.20 OEM Thermal Shutdown Enable

This function enables or disables the camera thermal shutdown intended to protect the camera from heating beyond operational temperature range.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_OEM_DISABLE	LEP_OEM_ENABLE	LEP_OEM_ENABLE	N/A	N/A

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x68**
With Get **0x68**
With Set **0x69**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry, Lepton 3	`LEP_GetOemThermalShutdownEnable()`	Updates `ThermalShutdownEnableStatePtr` with the Camera's current thermal shutdown enable state.
Lepton with Radiometry, Lepton 3	`LEP_SetOemThermalShutdownEnable()`	Updates the Camera's current thermal shutdown enable state with the contents of `ThermalShutdownEnableState`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemThermalShutdownEnable( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                          LEP_OEM_THERMAL_SHUTDOWN_ENABLE_T_PTR ThermalShutdownEnableStatePtr )

LEP_RESULT LEP_SetOemThermalShutdownEnable( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                  LEP_OEM_THERMAL_SHUTDOWN_ENABLE_T ThermalShutdownEnableState )
```

```
/* Enable State Enum
*/
typedef enum LEP_OEM_STATE_E_TAG
{
   LEP_OEM_DISABLE = 0,
   LEP_OEM_ENABLE,
   LEP_OEM_END_STATE

}LEP_OEM_STATE_E,*LEP_OEM_STATE_E_PTR;

/* Thermal Shutdown structure
```

```
*/
typedef struct LEP_OEM_THERMAL_SHUTDOWN_ENABLE_T_TAG
{
   LEP OEM STATE E oemThermalShutdownEnable;

}LEP OEM THERMAL SHUTDOWN ENABLE T, *LEP OEM THERMAL SHUTDOWN ENABLE T PTR;
```

4.6.21 OEM Bad Pixel Replacement Control

This function enables or disables the camera's bad pixel replacement control.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_OEM_DISABLE	LEP_OEM_ENABLE	LEP_OEM_ENABLE	N/A	N/A

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x6C**
With Get **0x6C**
With Set **0x6D**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry, Lepton 3	`LEP_GetOemBadPixelReplaceControl()`	Updates `BadPixelReplaceControlPtr`with the Camera's current bad pixel replacement control enable state.
Lepton with Radiometry, Lepton 3	`LEP_SetOemBadPixelReplaceControl()`	Updates the Camera's current bad pixel replacement control enable state with the contents of `BadPixelReplaceControl`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemBadPixelReplaceControl( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                            LEP_OEM_BAD_PIXEL_REPLACE_CONTROL_T_PTR BadPixelReplaceControlPtr )

LEP_RESULT LEP_SetOemBadPixelReplaceControl( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                    LEP_OEM_BAD_PIXEL_REPLACE_CONTROL_T BadPixelReplaceControl )
```

```
/* Enable State Enum
*/
typedef enum LEP_OEM_STATE_E_TAG
{
   LEP_OEM_DISABLE = 0,
   LEP_OEM_ENABLE,
   LEP_OEM_END_STATE

}LEP OEM STATE E,*LEP OEM STATE E PTR;


/* Bad Pixel Replacement Control structure
*/
```

```
typedef struct LEP_OEM_BAD_PIXEL_REPLACE_CONTROL_T_TAG
{
   LEP_OEM_STATE_E oemBadPixelReplaceEnable;

}LEP OEM BAD PIXEL REPLACE CONTROL T, *LEP OEM BAD PIXEL REPLACE CONTROL T PTR;
```

4.6.22 OEM Temporal Filter Control

This function enables or disables the camera's temporal filter.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_OEM_DISABLE	LEP_OEM_ENABLE	LEP_OEM_ENABLE	N/A	N/A

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x70**
With Get **0x70**
With Set **0x71**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry, Lepton 3	`LEP_GetOemTemporalFilterControl()`	Updates `TemporalFilterControlPtr` the Camera's current temporal filter enable state.
Lepton with Radiometry, Lepton 3	`LEP_SetOemTemporalFilterControl()`	Updates the Camera's current temporal filter enable state with the contents of `TemporalFilterControl`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemTemporalFilterControl( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                LEP_OEM_TEMPORAL_FILTER_CONTROL_T_PTR TemporalFilterControlPtr )

LEP_RESULT LEP_SetOemTemporalFilterControl( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                    LEP_OEM_TEMPORAL_FILTER_CONTROL_T TemporalFilterControl )
/* Enable State Enum
*/
typedef enum LEP_OEM_STATE_E_TAG
{
   LEP_OEM_DISABLE = 0,
   LEP_OEM_ENABLE,
   LEP_OEM_END_STATE

}LEP_OEM_STATE_E,*LEP_OEM_STATE_E_PTR;

/* Temporal Filter Control structure
*/
typedef struct LEP_OEM_TEMPORAL_FILTER_CONTROL_T_TAG
{
   LEP_OEM_STATE_E oemTemporalFilterEnable;

}LEP_OEM_TEMPORAL_FILTER_CONTROL_T, *LEP_OEM_TEMPORAL_FILTER_CONTROL_T_PTR;
```

4.6.23 OEM Column Noise Filter (SCNR) Control

This function enables or disables the camera's column noise filter.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_OEM_DISABLE	LEP_OEM_ENABLE	LEP_OEM_ENABLE	N/A	N/A

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x74**
With Get **0x74**
With Set **0x75**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry, Lepton 3	`LEP_GetOemColumnNoiseEstimateControl()`	Updates `ColumnNoiseEstimateControlPtr` the Camera's current column noise filter enable state.
Lepton with Radiometry, Lepton 3	`LEP_SetOemColumnNoiseEstimateControl()`	Updates the Camera's current column noise filter enable state with the contents of `ColumnNoiseEstimateControl`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemColumnNoiseEstimateControl( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                    LEP_OEM_COLUMN_NOISE_ESTIMATE_CONTROL_T_PTR ColumnNoiseEstimateControlPtr )

LEP_RESULT LEP_SetOemColumnNoiseEstimateControl( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                          LEP_OEM_COLUMN_NOISE_ESTIMATE_CONTROL_T ColumnNoiseEstimateControl )
```

```
/* Enable State Enum
*/
typedef enum LEP_OEM_STATE_E_TAG
{
   LEP_OEM_DISABLE = 0,
   LEP_OEM_ENABLE,
   LEP_OEM_END_STATE

}LEP OEM STATE E,*LEP OEM STATE E PTR;


/* Column Noise Filter Control structure
*/
```

```
typedef struct LEP_OEM_COLUMN_NOISE_ESTIMATE_CONTROL_T_TAG
{
   LEP_OEM_STATE_E oemColumnNoiseEstimateEnable;

}LEP OEM COLUMN NOISE ESTIMATE CONTROL T, *LEP OEM COLUMN NOISE ESTIMATE CONTROL T PTR;
```

4.6.24 OEM Pixel Noise Filter (SPNR) Control

This function enables or disables the camera's pixel noise filter.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_OEM_DISABLE	LEP_OEM_ENABLE	LEP_OEM_ENABLE	N/A	N/A

SDK Module ID: OEM **0x0800**

SDK Command ID: Base **0x78**
With Get **0x78**
With Set **0x79**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry, Lepton 3	`LEP_GetOemPixelNoiseSettings()`	Updates `pixelNoiseSettingsPtr` the Camera's current pixel noise filter enable state.
Lepton with Radiometry, Lepton 3	`LEP_SetOemPixelNoiseSettings()`	Updates the Camera's current pixel noise filter enable state with the contents of `pixelNoiseSettings`.

C SDK Interface:

```
LEP_RESULT LEP_GetOemPixelNoiseSettings( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                         LEP_OEM_PIXEL_NOISE_SETTINGS_T_PTR pixelNoiseSettingsPtr )

LEP_RESULT LEP_SetOemPixelNoiseSettings( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                         LEP_OEM_PIXEL_NOISE_SETTINGS_T pixelNoiseSettings )
```

```
/* Enable State Enum
*/
typedef enum LEP_OEM_STATE_E_TAG
{
   LEP_OEM_DISABLE = 0,
   LEP_OEM_ENABLE,
   LEP_OEM_END_STATE

}LEP_OEM_STATE_E,*LEP_OEM_STATE_E_PTR;

/* Pixel Noise Filter Control structure
*/
typedef struct LEP_OEM_PIXEL_NOISE_SETTINGS_T_TAG
{
   LEP_OEM_STATE_E oemPixelNoiseEstimateEnable;

}LEP_OEM_PIXEL_NOISE_SETTINGS_T, *LEP_OEM_PIXEL_NOISE_SETTINGS_T_PTR
```

4.6.25 OEM Run FFC Normalization Frames – *Aggregate Command*

This is an aggregate command that executes the FFC Normalization using a parameter to specify the FFC target value explicitly. This command does use the number of frames to average as specified by the SYS Number of frames to average (see 4.4.21).

Executing this command causes the camera to execute the FFC Normalization. The FFC target value is specified by parameter to this function. This command does not use the current target value in the Camera.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
0	16383	8192	N/A	N/A

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_RunOemFFCNormalization()`	Executes the FFC normalization using the FFC target value specified by parameter to this function. ***Aggregate command.***

C SDK Interface:

```
LEP_RESULT LEP_RunOemFFCNormalization( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_OEM_FFC_NORMALIZATION_TARGET_T ffcTarget )
```

```
typedef LEP_UINT16  LEP_OEM_FFC_NORMALIZATION_TARGET_T, *LEP_OEM_FFC_NORMALIZATION_TARGET_T_PTR;
```

4.7 SDK Module: RAD 0xE00

This module provides interfaces to the camera's radiometry features. Note that the Lepton and Lepton 3 releases includes radiometry features that support temperature stable output, but the Lepton with Radiometry release includes additional calibrations and radiometric features.

4.7.1 Setting the OEM Protection Bit

Issuing RAD commands requires setting the OEM Bit (Bit 14, value = 0x4000) in the command word in addition to setting the rest of the command word bits – see Figure 10**.** If this bit is not set to 1 for each RAD command issued, the camera will return an error code. If this bit is set when issuing any other commands that are not OEM or RAD commands, then the camera will also return an error code. See section 4.3.2 for a description of how an OEM module or RAD module command is synthesized.

4.7.2 RAD RBFO External Parameters

This function gets and sets the radiometry RBFO External parameters. The RBFO parameters define the equation for conversion between flux and temperature.

	Minimum Value	Maximum Value	Default Setting	Lepton with Radiometry Factory Default	Units	Scale factor
R	10000	1000000	395653	Calibrated per camera	Counts	1
B	1200000	1700000	1428000	Calibrated per camera	Kelvin	1000
F	500	3000	1000	Calibrated per camera	N/A	1000
O	-16384000	16383000	156000	Calibrated per camera	Counts	1000

SDK Module ID: RAD **0x0E00**

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

SDK Data Length: Get **8** size of a `LEP_RBFO_T` data type, 4 x 2 words each
Set **8** size of a `LEP_RBFO_T` data type, 4 x 2 words each

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetRadRBFOExternal0()`	Updates `radRBFOPtr` with the Camera's current RBFO External parameters.
All Lepton Configurations	`LEP_SetRadRBFOExternal0()`	Updates the Camera's RBFO External parameters with the contents of `radRBFOPtr`.

C SDK Interface:

```
LEP_RESULT LEP_GetRadRBFOExternal0( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                    LEP_RBFO_T_PTR radRBFOPtr )

LEP_RESULT LEP_SetRadRBFOExternal0( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
```

LEP_RBFO_T_PTR radRBFOPtr)

```
/* RBFO
*/
typedef struct LEP RBFO T TAG
{
   LEP_UINT32 RBFO_R;   // value is not scaled
   LEP_UINT32 RBFO_B;   // value is scaled by X  << n
   LEP_UINT32 RBFO_F;
   LEP_INT32  RBFO_O;

}LEP_RBFO_T, *LEP_RBFO_T_PTR;
```

4.7.3 RAD Radiometry Control Enable

This function enables or disables the Camera Radiometry Control (temperature stable output), or returns the state of Control.

	Minimum Value	Maximum Value	Default Setting	Lepton with Radiometry Factory Default	Units	Scale factor
radEnableState	LEP_RAD_DISABLE	LEP_RAD_ENABLE	LEP_RAD_DISABLE	LEP_RAD_ENABLE	N/A	N/A

SDK Module ID: RAD **0x0E00**

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

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 value

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetRadEnableState()`	Updates `radEnableStatePtr` with current state of the radiometry control.
All Lepton Configurations	`LEP_SetRadEnableState()`	Updates the Camera's with current state of the radiometry control with the contents of `radEnableState`.

C SDK Interface:

```
LEP_RESULT LEP_GetRadEnableState( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                  LEP_RAD_ENABLE_E_PTR radEnableStatePtr )

LEP_RESULT LEP_SetRadEnableState( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                  LEP_RAD_ENABLE_E radEnableState )
```

```
/* Radiometry Enable state
*/
typedef enum LEP_RAD_ENABLE_E_TAG
{
   LEP_RAD_DISABLE = 0,
   LEP_RAD_ENABLE,
   LEP_END_RAD_ENABLE

}LEP_RAD_ENABLE_E, *LEP_RAD_ENABLE_E_PTR;
```

4.7.4 RAD TShutter Mode

This function gets or sets the TShutter mode. The TShutter mode specifies how TShutter value is obtained at FFC.

- User: Use the TShutter value set with LEP_SetRadTShutter()
- Cal: Use TEqShutter from calibration
- Fixed: the shutter temperature is considered static, and therefore the spotmeter is not updated at FFC

	Minimum Value	Maximum Value	Default Setting	Units	Scale factor
radTShutterMode	FLR_RAD_TS_USER_MODE	FLR_RAD_TS_FIXED_MODE	FLR_RAD_TS_CAL_MODE	N/A	N/A

SDK Module ID: RAD **0x0E00**

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**LEP_GetRadTShutterMode()**	Updates **radTShutterModePtr** with current TShutter mode
All Lepton Configurations	**LEP_SetRadTShutterMode()**	Updates the Camera's current TShutter mode with the contents of **radTShutterMode**.

C SDK Interface:

```
LEP_RESULT LEP_GetRadTShutterMode( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                   LEP_RAD_TS_MODE_E_PTR radTShutterModePtr )

LEP_RESULT LEP_SetRadTShutterMode( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                   LEP_RAD_TS_MODE_E radTShutterMode )
```

```
/* TShutter Modes
*/
typedef enum FLR_RAD_TS_MODE_E_TAG
{
   FLR_RAD_TS_USER_MODE = 0,
   FLR_RAD_TS_CAL_MODE,
   FLR_RAD_TS_FIXED_MODE,
   FLR_RAD_TS_END_TS_MODE

}FLR_RAD_TS_MODE_E, *FLR_RAD_TS_MODE_E_PTR;
```

4.7.5 RAD TShutter Temperature

This function gets or sets the TShutter temperature. The TShutter temperature is used at FFC when the TShutter Mode is "User".

	Minimum Value	Maximum Value	Default Setting	Units	Scale factor
`radTShutter`	0	65535	30000	N/A	N/A

SDK Module ID: RAD **0x0E00**

SDK Command ID: Base **0x28**
With Get **0x28**
With Set **0x29**

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_GetRadTShutter()`	Updates `radTShutterPtr` with current TShutter temperature
All Lepton Configurations	`LEP_SetRadTShutter()`	Updates the Camera's current TShutter temperature with the contents of `radTShutter`.

C SDK Interface:

```
LEP_RESULT LEP_GetRadTShutter( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                               LEP_RAD_KELVIN_T_PTR radTShutterPtr )

LEP_RESULT LEP_SetRadTShutter( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                               LEP_RAD_KELVIN_T radTShutter )
```

```
/* TShutter value is 100xKelvin [16.0]
*/
typedef LEP_UINT16  LEP_RAD_KELVIN_T, *LEP_RAD_KELVIN_T_PTR;
```

4.7.6 RAD FFC Normalization

This command executes a Flat-Field Correction (FFC) and updates the Global Gain and Global Offset. The target value is factory set and should not be changed under normal circumstances. The Run command executes an FFC using currently active values for the FFC normalization target and number of frames to average (see 4.4.21). This command executes synchronously. Poll the RAD Run Status to determine when this command completes (see 4.7.7).

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

SDK Module ID: SYS **0x0E00**

SDK Command ID: Base **0x2C**
With Run **0x2E**

SDK Data Length: Run **0** size of a Run command

Compatibility	C-SDK Commands	Description
All Lepton Configurations	`LEP_RunRadFFC()`	Executes the FFC normalization using previously specified normalization target value and calculates the Global Gain and Global Offset

C SDK Interface:

```
LEP_RESULT LEP_RunRadFFC( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr )
```

4.7.7 RAD Run Status

This function obtains the current status of a RAD module run operation. This function is used whenever a RAD command is issued that executes an operation like the run FFC. Typically, the host polls the status to determine when the command has completed. If the return value is negative, then the operation completed with an error. Positive values indicate an in-process state.

Minimum Value	Maximum Value	Default Setting	Units	Scale factor
LEP_RAD_STATUS_ERROR	LEP_RAD_FRAME_AVERAGE_COLLECTING_FRAMES	LEP_RAD_STATUS_READY	N/A	N/A

SDK Module ID: OEM **0x0E00**

SDK Command ID: Base **0x30**
With Run **0x30**

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

Compatibility	C-SDK Commands	Description
All Lepton Configurations	**LEP_GetRadRunStatus()**	Gets the Current RAD run operation status. Updates **radStatusPtr** with current value of the run status

C SDK Interface:

```
LEP_RESULT LEP_GetRadRunStatus( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                LEP_RAD_STATUS_E_PTR radStatusPtr )
```

```
/* Run operation status
*/
typedef enum
{
   LEP_RAD_STATUS_ERROR = -1,
   LEP_RAD_STATUS_READY = 0,
   LEP_RAD_STATUS_BUSY,
   LEP_RAD_FRAME_AVERAGE_COLLECTING_FRAMES,

   LEP_RAD_STATUS_END

} LEP_RAD_STATUS_E, *LEP_RAD_STATUS_E_PTR;
```

4.7.8 RAD Flux Linear Parameters

These functions either get or set various scene parameters used in the T-Linear calculations.

	Minimum Value	Maximum Value	Default Setting	Lepton with Radiometry Factory Default	Units	Scale factor
sceneEmissivity	82	8192	8192	8192	Percent	8192*0.01
TBkgK	0	65535	30000	29515	Kelvin	100
tauWindow	82	8192	8192	8192	Percent	8192*0.01
TWindowK	0	65535	30000	29515	Kelvin	100
tauAtm	82	8192	8192	8192	Percent	8192*0.01
TAtmK	0	65535	30000	29515	Kelvin	100
reflWindow	0	8192–**tauWindow**	0	0	Percent	8192*0.01
TReflK	0	65535	30000	29515	Kelvin	100

SDK Module ID: RAD **0x0E00**

SDK Command ID: Base **0xBC**
With Get **0xBC**
With Set **0xBD**

SDK Data Length: Get **8** size of **LEP_RAD_FLUX_LINEAR_PARAMS_T** data type
Set **8** size of **LEP_RAD_FLUX_LINEAR_PARAMS_T** data type

Compatibility	C-SDK Commands	Description
Lepton with Radiometry	`LEP_GetRadFluxLinearParams()`	Updates `fluxParamsPtr` with the camera's Radiometry scene parameters used for T-Linear calculation.
Lepton with Radiometry	`LEP_SetRadFluxLinearParams()`	Updates the Camera's current Radiometry scene parameters with the contents of `fluxParams`.

C SDK Interface:

```
LEP_RESULT LEP_GetRadFluxLinearParams (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                       LEP_RAD_FLUX_LINEAR_PARAMS_T_PTR fluxParamsPtr)
```

```
   LEP_RESULT LEP_SetRadFluxLinearParams (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                          LEP_RAD_FLUX_LINEAR_PARAMS_T fluxParams)


/* Radiometry Flux Linear Params
*/
typedef struct LEP_RAD_FLUX_LINEAR_PARAMS_T_TAG
{
   /* Type      Field name            format    comment */
   LEP_UINT16   sceneEmissivity; /*   3.13                                */
   LEP_UINT16   TBkgK;           /*   16.0      value in Kelvin 100x      */
   LEP_UINT16   tauWindow;       /*   3.13                                */
   LEP_UINT16   TWindowK;        /*   16.0      value in Kelvin 100x      */
   LEP_UINT16   tauAtm;          /*   3.13                                */
   LEP_UINT16   TAtmK;           /*   16.0      value in Kelvin 100x      */
   LEP_UINT16   reflWindow;      /*   3.13                                */
   LEP_UINT16   TReflK;          /*   16.0      value in Kelvin 100x      */

}LEP_RAD_FLUX_LINEAR_PARAMS_T, *LEP_RAD_FLUX_LINEAR_PARAMS_T_PTR;
```


4.7.9 RAD T-Linear Enable State

These functions either get or set the T-Linear output enable state. When enabled, the video output represents temperature in Kelvin with some scale factor defined by the T-linear Resolution parameter. T-Linear mode requires radiometry mode (temperature stable output) to also be enabled.

	Minimum Value	Maximum Value	Default Setting	Lepton with Radiometry Factory Default	Units	Scale factor
enableState	LEP_RAD_DISABLE	LEP_RAD_ENABLE	LEP_RAD_DISABLE	LEP_RAD_ENABLE	N/A	N/A

SDK Module ID: RAD **0x0E00**

SDK Command ID: Base **0xC0**
With Get **0xC0**
With Set **0xC1**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry	**LEP_GetRadTLinearEnableState()**	Updates **enableStatePtr** with the camera's T-Linear calculation enable state.
Lepton with Radiometry	**LEP_SetRadTLinearEnableState()**	Updates the Camera's current T-Linear calculation enable state with the contents of **enableState**.

C SDK Interface:

```
LEP_RESULT LEP_GetRadTLinearEnableState (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                         LEP_RAD_ENABLE_E_PTR enableStatePtr)

LEP_RESULT LEP_SetRadTLinearEnableState (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                         LEP_RAD_ENABLE_E enableState)
```

```
/* Radiometry Enable state
*/
typedef enum LEP_RAD_ENABLE_E_TAG
{
   LEP_RAD_DISABLE = 0,
   LEP_RAD_ENABLE,
   LEP_END_RAD_ENABLE

}LEP_RAD_ENABLE_E, *LEP_RAD_ENABLE_E_PTR;
```

4.7.10 RAD T-Linear Resolution

These functions either get or set the T-Linear output resolution, which defines the scale factor for the temperature measurements contained in the video output (per-pixel) with T-Linear mode enabled.

	Minimum Value	Maximum Value	Default Setting	Lepton with Radiometry Factory Default
`resolution`	LEP_RAD_RESOLUTION_0_1	LEP_RAD_RESOLUTION_0_01	LEP_RAD_RESOLUTION_0_1	LEP_RAD_RESOLUTION_0_01

Setting	Minimum Pixel Value	Maximum Pixel Value	Units	Scale factor
LEP_RAD_RESOLUTION_0_1	0	6553.5	Kelvin	10
LEP_RAD_RESOLUTION_0_01	0	655.35	Kelvin	100

SDK Module ID: RAD **0x0E00**

SDK Command ID: Base **0xC4**
With Get **0xC4**
With Set **0xC5**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry	`LEP_GetRadTLinearResolution()`	Updates `resolutionPtr` with the camera's T-Linear calculation resolution.
Lepton with Radiometry	`LEP_SetRadTLinearResolution()`	Updates the Camera's current T-Linear calculation resolution with the contents of `resolution`.

C SDK Interface:

```
LEP_RESULT LEP_GetRadTLinearResolution (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                        LEP_RAD_TLINEAR_RESOLUTION_E_PTR resolutionPtr)

LEP_RESULT LEP_SetRadTLinearResolution (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                        LEP_RAD_TLINEAR_RESOLUTION_E resolution)
```

```
/* Radiometry T-Linear Resolution
*/
typedef enum LEP_RAD_TLINEAR_RESOLUTION_E_TAG
{
   LEP_RAD_RESOLUTION_0_1 = 0,
   LEP_RAD_RESOLUTION_0_01,
   LEP_RAD_END_RESOLUTION

}LEP_RAD_TLINEAR_RESOLUTION_E, *LEP_RAD_TLINEAR_RESOLUTION_E_PTR;
```

4.7.11 RAD T-Linear Auto Resolution

These functions either get or set the T-Linear automatic resolution enable state. When enabled, T-Linear output resolution is chosen automatically based on scene statistics and gain mode.

	Minimum Value	Maximum Value	Default Setting	Units	Scale factor
enableState	LEP_RAD_DISABLE	LEP_RAD_ENABLE	LEP_RAD_DISABLE	N/A	N/A

SDK Module ID: RAD **0x0E00**

SDK Command ID: Base **0xC8**
With Get **0xC8**
With Set **0xC9**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry	**LEP_GetRadTLinearAutoResolution()**	Updates **enableStatePtr** with the camera's T-Linear automatic resolution feature enable state.
Lepton with Radiometry	**LEP_SetRadTLinearAutoResolution()**	Updates the Camera's current T-Linear automatic resolution feature with the contents of **enableState**.

C SDK Interface:

```
LEP_RESULT LEP_GetRadTLinearAutoResolution (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                            LEP_RAD_ENABLE_E_PTR enableStatePtr)

LEP_RESULT LEP_SetRadTLinearAutoResolution (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                            LEP_RAD_ENABLE_E enableState)
```

```
/* Radiometry Enable state
*/
typedef enum LEP_RAD_ENABLE_E_TAG
{
   LEP_RAD_DISABLE = 0,
   LEP_RAD_ENABLE,
   LEP_END_RAD_ENABLE

}LEP_RAD_ENABLE_E, *LEP_RAD_ENABLE_E_PTR;
```

4.7.12 RAD Spotmeter Region of Interest (ROI)

These functions either get or set a rectangular region of interest within the video frame extents which RAD operations can use to calculate temperature measurement minimum, maximum, and average.

Dimension	Minimum Value	Maximum Value	Default Value	Units	Scale factor
start column	0	< end column-1	39	pixels	1
start row	0	< end row-1	29	pixels	1
end column	> start column+1	79	40	pixels	1
end row	> start row+1	59	30	pixels	1

SDK Module ID: RAD **0x0E00**

SDK Command ID: Base **0xCC**
With Get **0xCC**
With Set **0xCD**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry	`LEP_GetRadSpotmeterRoi()`	Updates `spotmeterRoiPtr` with the camera's current spotmeter ROI.
Lepton with Radiometry	`LEP_SetRadSpotmeterRoi()`	Updates the Camera's current spotmeter ROI with the contents of `spotmeterRoi`.

C SDK Interface:

```
LEP_RESULT LEP_GetRadSpotmeterRoi (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                   LEP_RAD_ROI_T_PTR spotmeterRoiPtr)
LEP_RESULT LEP_SetRadSpotmeterRoi (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                   LEP_RAD_ROI_T spotmeterRoi)
```

```
/* Radiometry ROI
*/
typedef struct LEP_RAD_ROI_T_TAG
{
   LEP_UINT16   startRow;
   LEP_UINT16   startCol;
   LEP_UINT16   endRow;
   LEP_UINT16   endCol;

}LEP_RAD_ROI_T, *LEP_RAD_ROI_T_PTR;
```

4.7.13 RAD Spotmeter Value

These functions get the mean, min, and max temperature values for pixels within the spotmeter ROI.

Dimension	Minimum Value	Maximum Value	Default Value	Units	Scale factor
radSpotmeterValue	0	65535	N/A	Kelvin	100
radSpotmeterMaxValue	0	65535	N/A	Kelvin	100
radSpotmeterMinValue	0	65535	N/A	Kelvin	100
radSpotmeterPopulation	0	4800	N/A	Pixels	1

SDK Module ID: RAD **0x0E00**

SDK Command ID: Base **0xD0**
With Get **0xD0**

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

Compatibility	C-SDK Commands	Description
Lepton with Radiometry	`LEP_GetRadSpotmeterObjInKelvinX100()`	Updates `kelvinPtr` with the camera's current spotmeter values.

C SDK Interface:

```
LEP_RESULT LEP_GetRadSpotmeterObjInKelvinX100 (LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                               LEP_RAD_SPOTMETER_OBJ_KELVIN_T_PTR kelvinPtr)

/* Radiometry ROI
*/
typedef LEP UINT16 LEP RAD SPOTMETER KELVIN T, *LEP RAD SPOTMETER KELVIN T PTR;


typedef struct LEP_RAD_SPOTMETER_OBJ_KELVIN_T_TAG
{
   LEP RAD SPOTMETER KELVIN T   radSpotmeterValue;
   LEP_UINT16    radSpotmeterMaxValue;
   LEP_UINT16    radSpotmeterMinValue;
   LEP_UINT16    radSpotmeterPopulation;

}LEP_RAD_SPOTMETER_OBJ_KELVIN_T, *LEP_RAD_SPOTMETER_OBJ_KELVIN_T_PTR;
```

4.7.14 RAD Low Gain RBFO External Parameters

These functions either get or set the radiometry low gain RBFO External parameters. The RBFO parameters define the equation for conversion between flux and temperature.

	Minimum Value	Maximum Value	Default Setting	Lepton with Radiometry Factory Default	Units	Scale factor
R	10000	1000000	64155	Calibrated per camera	Counts	1
B	1200000	1700000	1428000	Calibrated per camera	Kelvin	1000
F	500	3000	1000	Calibrated per camera	N/A	1000
O	-16384000	16383000	728000	Calibrated per camera	Counts	1000

SDK Module ID: RAD **0x0E00**

SDK Command ID: Base **0xD8**
With Get **0xD8**
With Set **0xD9**

SDK Data Length: Get **8** size of a `LEP_RBFO_T` data type, 4 x 2 words each
Set **8** size of a `LEP_RBFO_T` data type, 4 x 2 words each

Compatibility	C-SDK Commands	Description
Lepton with Radiometry	`LEP_GetRadExternalRBFOLowGain()`	Updates `radRBFOPtr` with the Camera's current RBFO External parameters.
Lepton with Radiometry	`LEP_SetRadExternalRBFOLowGain()`	Updates the Camera's RBFO External parameters with the contents of `radRBFOPtr`.

C SDK Interface:

```
LEP_RESULT LEP_GetRadExternalRBFOLowGain ( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                           LEP_RBFO_T_PTR radRBFOPtr )
LEP_RESULT LEP_SetRadExternalRBFOLowGain ( LEP_CAMERA_PORT_DESC_T_PTR portDescPtr,
                                           LEP_RBFO_T_PTR radRBFOPtr )
```

```
/* RBFO
*/
typedef struct LEP_RBFO_T_TAG
{
   LEP_UINT32 RBFO_R;    // value is not scaled
   LEP_UINT32 RBFO_B;    // value is scaled by X  << n
   LEP_UINT32 RBFO_F;
   LEP_INT32  RBFO_O;

}LEP_RBFO_T, *LEP_RBFO_T_PTR;
```

FLIR LEPTON® Software IDD

If you have questions that are not covered in this manual, or need service, contact FLIR Commercial Systems Customer Support at 805.964.9797 for additional information prior to returning a camera.

This documentation and the requirements specified herein are subject to change without notice.

This equipment must be disposed of as electronic waste.

Contact your nearest FLIR Commercial Systems, Inc. representative for instructions on how to return the product to FLIR for proper disposal.

FCC Notice. This device is a subassembly designed for incorporation into other products in order to provide an infrared camera function. It is not an end-product fit fo1r consumer use. When incorporated into a host device, the end-product will generate, use, and radiate radio frequency energy that may cause radio interference. As such, the end-product incorporating this subassembly must be tested and approved under the rules of the Federal Communications Commission (FCC) before the end-product may be offered for sale or lease, advertised, imported, sold, or leased in the United States. The FCC regulations are designed to provide reasonable protection against interference to radio communications. See 47 C.F.R. §§ 2.803 and 15.1 et seq.

Industry Canada Notice. This device is a subassembly designed for incorporation into other products in order to provide an infrared camera function. It is not an end-product fit for consumer use. When incorporated into a host device, the end-product will generate, use, and radiate radio frequency energy that may cause radio interference. As such, the end-product incorporating this subassembly must be tested for compliance with the Interference-Causing Equipment Standard, Digital Apparatus, ICES-003, of Industry Canada before the product incorporating this device may be: manufactured or offered for sale or lease, imported, distributed, sold, or leased in Canada.

Avis d'Industrie Canada. Cet appareil est un sous-ensemble conçu pour être intégré à un autre produit afin de fournir une fonction de caméra infrarouge. Ce n'est pas un produit final destiné aux consommateurs. Une fois intégré à un dispositif hôte, le produit final va générer, utiliser et émettre de l'énergie radiofréquence qui pourrait provoquer de l'interférence radio. En tant que tel, le produit final intégrant ce sous-ensemble doit être testé pour en vérifier la conformité avec la Norme sur le matériel brouilleur pour les appareils numériques (NMB-003) d'Industrie Canada avant que le produit intégrant ce dispositif puisse être fabriqué, mis en vente ou en location, importé, distribué, vendu ou loué au Canada.

EU Notice. This device is a subassembly or component intended only for product evaluation, development or incorporation into other products in order to provide an infrared camera function. It is not a finished end-product fit for general consumer use. Persons handling this device must have appropriate electronics training and observe good engineering practice standards. As such, this product does not fall within the scope of the European Union (EU) directives regarding electromagnetic compatibility (EMC). Any end-product intended for general consumer use that incorporates this device must be tested in accordance and comply with all applicable EU EMC and other relevant directives.

FLIR® | The World's Sixth Sense™