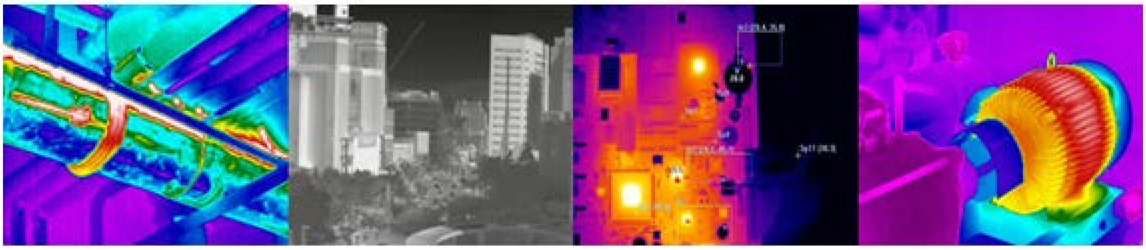
Cox Thermal Camera Analyzing Software SDK

Manual



Dear User

Editorial deadline: August 2011

Document number: 00000000-000000-00000-000

COX Co. Ltd

#810, Hanwha Biz Metro 1-cha,
, 242, Digital-ro, Guro-gu, Seoul,
KOREA

Phone: +82-2-857-3888
Fax: +82-70-7614-3871

Note:

No part of this User Manual may be reproduced in any form (print, photocopy, microfilm or any other process), nor may it be processed, duplicated or distributed using electronic systems without the written permission of COX.
This document was produced with appropriate care. No liability will be accepted in the case of damages arising from non-observance of the information contained herein.
We reserve the right to modify this document following technical advancements.

Contents

1. Revision History

Version	Date	Notes
A.0	Aug. 2011	Initial Release
A.1	Sep. 2011	Added functions
A.2	Nov. 2011	Update file stream functions
A.3	Dec. 2011	Update Isotherm function
A.4	Mar. 2012	Added alarm duration
A.5	Jun. 2012	Added spot streaming
A.6	Jul. 2012	Update file stream functions
A.7	Oct. 2014	Apply CX640

2. Introduction

CoxCamDll.Dll contains all the necessary functions for convenient image transfer from Cox CTC camera.

CoxamDll.Dll can be effectively integrated into applications created using Microsoft VISUAL C++.

The software package comprises the following files:

- ThermalCamDll.Dll
- ThermalCamDll.Lib
- ThermalCamDll.h
- colormap.bin
- TempMapTable_L.bin
- TempMapTable_M.bin

3. List of export functions

Functions for camera connection:

```
short stdcall OpenConnect(HANDLE *pHandle, UINT *pTimerID, LPCTSTR strDestination,
                          LPCTSTR strServiceName, int nProtocol, int nType);
short   stdcall CloseConnect(HANDLE *handle, UINT timerID);
short   stdcall SendCameraMessage(HANDLE handle, UINT *pTimerID, IRF_MESSAGE_TYPE_T
                                  type, unsigned short PMSGTYPE, unsigned short RCODE);
short   stdcall SendCameraMessage(HANDLE handle, UINT *pTimerID, IRF_MESSAGE_TYPE_T
                                  type, unsigned short PMSGTYPE, unsigned short  RCODE,
                                  DWORD RCODE2, DWORD RCODE3, DWORD RCODE4);
short   stdcall SendCameraMessage(HANDLE handle, UINT *pTimerID, IRF_SET_USER_PALETTE
                                  struct_palette);
short  stdcall GetIRImages(HANDLE handle, UINT *pTimerID, IRF_IR_CAM_DATA_T* cam_data);
```

Functions for temperature map table and palette table:

```
short  stdcall GetImageLUT(unsigned char *palette, IRF_PALETTE_TYPE_T paletteType,
                           bool bInvert);
short   stdcall GetTempMapTable(float* tempLUT, IRF_DYNAMIC_RANGE_T tempMode);
```

Functions to convert gray image to IR image:

```
short   stdcall GetImage(unsigned char *image, HANDLE ir_image, long image_size,
                         float *tempLUT, float *level, float *span, IRF_AUTO_RANGE_METHOD_T
                         *method);
short   stdcall GetCorrectedImage(unsigned char *image, HANDLE ir_image, long image_size,
                         float *tempLUT , IRF_TEMP_CORRECTION_PAR_T corrPara, float  *level,
                         float *span, IRF_AUTO_RANGE_METHOD_T*method);
short  stdcall GetGrayToPaletteImage(unsigned char *from_image, void* to_image,
                        unsigned short width, unsigned short height, unsigned char *palette,
                        int BitsPixel, bool bMirror);
```

Function for IR histogram:

```
short  _stdcall GetIRHistogram(unsigned int *hist, unsigned short *ir_image, long image_size);
```

Function for dynamic temperature range:

```
short  _stdcall GetTempRangeValue(IRF_DYNAMIC_RANGE_T tempMode, short *min, short *max);
```

Functions to convert temperature type:

```
Float_ stdcall ConvertFahToCels(float temp);
float_ stdcall ConvertCelsToFah(float temp);
float_ stdcall ConvertKelvToCels(float temp);
float_ stdcall ConvertKelvToFah(float temp);
float_  stdcall ConvertCelToKel(float temp);
float_ stdcall ConvertFahToKel(float temp);
```

Functions to get temperature:

```
Float__stdcall GetPointTemp(HANDLE ir_image, IRF_IMAGE_INFO_T image_info,
                            float *tempLUT, IRF_TEMP_CORRECTION_PAR_T corrPara, POINT pt);
float__stdcall GetNeighborPointTemp(HANDLE ir_image, IRF_IMAGE_INFO_T image_info,
                            float *tempLUT, IRF_TEMP_CORRECTION_PAR_T corrPara, POINT pt);
short__stdcall GetROITemp(HANDLE ir_image, IRF_IMAGE_INFO_T image_info,
                            float *tempLUT, IRF_TEMP_CORRECTION_PAR_T corrPara,
                            RECT roi, IRF_NUMERIC_INFO_T *numInfo, POINT *min_pt,
                            POINT *max_pt);
short__stdcall GetRawToTemp(HANDLE ir_image, IRF_IMAGE_INFO_T image_info,
                            float *tempLUT, IRF_TEMP_CORRECTION_PAR_T corrPara,
                            float* tempImage);
float__tdcall GetCorrectedTemp(float *tempLUT, IRF_TEMP_CORRECTION_PAR_T corrPara,
                            unsigned short engineOut);
float__stdcall GetIRdataToTemp(unsigned short ir_data, float *tempLUT,
                            IRF_TEMP_CORRECTION_PAR_T corrPara);
```

Functions for Image filter:

```
Short__stdcall ApplyImageFilter(unsigned char *image, unsigned short width, unsigned short height,
                            IRF_IMAGE_FILTER_T filter);
Short__stdcall ApplyColorImageFilter(void* image, unsigned short width, unsigned short height,
                            IRF_IMAGE_FILTER_T filter, int bitPixel);
Void__stdcall BilateralFilter(unsigned char *image, unsigned short width, unsigned short height,
                            float sigD, float sigR, int w);
Void__stdcall GetGaussianKernel(int *kernel, int *mult, int sz);
Short__stdcall FastGaussianBlur(BYTE *img, int iw, int ih, int *Gkernel, int *Gmult, int radius);
short__stdcall FastStackBlur(BYTE* img, int w, int h, int radius);
```

```
Short__stdcall BoxBlur(BYTE *src, int src_w, int src_h, int sz);
```

Functions for IR file stream:

```
Short__stdcall GetIRHeader(HANDLE handle, IRF_IR_FILE_HEADER_T* header,
                           unsigned long *curPos);
Short__stdcall PASCAL GetIRHeaders(HANDLE handle,
                         IRF_IR_FILE_HEADER_T* header, IRF_IR_DATA_HEADER_T* addedInfo,
                         unsigned long *curPos);
short__stdcall LoadIRImage(HANDLE *handle, char *FileName, unsigned long *totSize);
short__stdcall GetIRImageFromStream(HANDLE handle, unsigned short* ir_image, long image_size,
                           unsigned long totStreamSize, unsigned long *curPos, int* gap_time,
                           bool bLoop);.
Short__stdcall GetIRImageFromStream_n(HANDLE handle, unsigned short* ir_image,
                           long image_size, unsigned long totStreamSize, unsigned long *curPos,
                           int* gap_time, bool bLoop, bool new_ver);
short__stdcall GetIRImageFromStream_v2(HANDLE handle, unsigned short* ir_image,
                           long image_size, unsigned long totStreamSize, unsigned long *curPos,
                           int* gap_time,    int64 *curTime, bool bLoop, unsigned char ver);
short__stdcall GetRevIRImageFromStream(HANDLE handle, unsigned short* ir_image,
                           long image_size, unsigned long *curPos, int* gap_time);
short__stdcall GetRevIRImageFromStream_n(HANDLE handle, unsigned short* ir_image,
                           long image_size, unsigned long *curPos, int* gap_time, bool new_ver);
short__stdcall GetRevIRImageFromStream_v2(HANDLE handle, unsigned short* ir_image,
                           long image_size, unsigned long *curPos, int* gap_time, int64 *curTime,
                           unsigned char ver);
short__stdcall SaveIRImage(HANDLE *handle, char* filename, IRF_IR_FILE_HEADER_T *pHeader);
short__stdcall PASCAL SaveIRHeader(HANDLE *handle, char* filename,
                     IRF_IR_FILE_HEADER_T *pHeader, IRF_IR_DATA_HEADER_T *pAddedData);
short__stdcall SetIRImageToStream(HANDLE handle, unsigned short* ir_image, long image_size,
                           int millisecond, short *frameCnt);
short__stdcall SetIRImageToStream_v2(HANDLE handle, unsigned short* ir_image, long image_size,
                           int millisecond, short *frameCnt, unsigned char ver);
short__stdcall CloseIRStream(HANDLE handle);
```

Functions for color-bar and min/max position drawing:

Void__stdcall DrawColorBar(HWND hWnd, HDC hDC, unsigned char* palette, float level, float span,
IRF_TEMP_MODE_T tempUnit, bool bUpdateOnlyTickArea);

Void__stdcall DrawMinMaxPos(HDC hDC, POINT minP, POINT maxP, int size);

Function for error message display:

Short__stdcall GetError(short code, LPCTSTR msg);

4. Description of the functions

Connect Method

Connect with the camera using communication interface.

Syntax

```
ShortOpenConnect( HAND
   LE* pHandle, UINT*
   pTimerID, LPCTSTR
   strDestination,
   LPCTSTR strServiceName,
   int nProtocol,
   int nType);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
pHandle	Socket handle. [out]
pTimerID	Timer ID using sending alive packet to a camera. [out]
strDestination	Camera network address (IP Address) [in]
strServiceName	Camera network port(15001). [in]
nProtocol	Address family (AF_INET). [in]
nType	Socket type (SOCK_STREAM or SOCK_DGRAM). [in]

Disconnect Method

Disconnect the camera and exit the digital transfer mode. You should always disconnect the camera before shutting down your application.

Syntax

```
ShortCloseConnect(
    HANDLE*  handle,
   UINT timerID
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
Handle	Socket handle. [in]
pTimerID	Timer ID using sending alive packet to a camera. [in]

GetIRHeader Method

Get IR header information from the camera.

Syntax

```
shortGetIRHeader(
   HANDLE handle,
   IRF_IR_FILE_HEADER_T* header,
   Unsigned long* curPos
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
Handle	FILE handle. [in]
header	IR-file header structure. [out]
curPos	Current file stream position. [out]

GetIRHeaders Method

Get IR header information from the camera.

Syntax

```
shortGetIRHeader(
   HANDLE handle,
   IRF_IR_FILE_HEADER_T* header,
   IRF_IR_DATA_HEADER_T* addedInfo,
   Unsigned long* curPos
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
Handle	FILE handle. [in]
header	IR-file header structure. [out]
Header2	IR-data header structure. [out]
curPos	Current file stream position. [out]

SendCameraMessage Method

Transfer message to a camera with TCP/IP.

Syntax

```
shortSendCameraMessage(
   HANDLE handle,
   UINT* pTimerID,
   IRF_MESSAGE_TYPE_T type,
   unsigned short PMSGTYPE,
   unsigned short RCODE
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
Handle	Socket handle. [in]
pTimerID	Timer ID using to send alive packet to a camera. [in]
type	Sending message type (IRF_MESSAGE_TYPE_T). [in]
PMSGTYPE	Primary message type code. [in]
RCODE	Response code or fail code. [in]

TYPE	PMSGTYPE	RCODE	DESCRIPTION
_IRF_STREAM_ON	0	0	Request to start raw data transfer.
_IRF_STREAM_OFF	0	0	Request to stop raw data transfer.
_IRF_SPOT_STREAM_ON	0	0	Request to start spot data transfer.
_IRF_SPOT_STREAM_OFF	0	0	Request to stop spot data transfer.

SendMessageToCamera Method

Transfer message to a camera with TCP/IP.

Syntax

```
shortSendMessageToCamera(HA
   NDLE handle,
   UINT* pTimerID,
   IRF_MESSAGE_TYPE_T type,
   unsigned short PMSGTYPE,
   unsigned short RCODE,
   DWORD RCODE2,
   DWORD RCODE3,
   DWORD RCODE4
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
Handle	Socket handle. [in]
pTimerID	Timer ID using to send alive packet to a camera. [in]
type	Sending message type (IRF_MESSAGE_TYPE_T). [in]
PMSGTYPE	Primary message type code (Camera setting command code). [in]
RCODE	Response code or fail code. [in]
RCODE2	Response code.[in]
RCODE3	Response code.[in]
RCODE4	Response code.[in]

PMSGTYPE	RCODE	RCODE2	RCODE3	RCODE4
CMD_AGC	0:OFF 1:HISTOGRAM 2:BRIGHTNESS	0	0	0
CMD_LEVEL	-20 ~ 120	0	0	0
CMD_SPAN	10 ~ 100	0	0	0
CMD_PALETTE	IRF_CAM_PALETTE_ TYPE_T	0	0	0
CMD_INVERT	0:OFF, 1:ON	0	0	0
CMD_MIRROR	0:OFF, 1:ON	0	0	0
CMD_FLIP	0:OFF, 1:ON	0	0	0
CMD_ZOOM	0:1X, 1:2X, 2:3X	0	0	0

CMD_NOISE_FILTER	0:OFF 1:SLIGHT SHARPTEN 2:STRONG SHARPTEN 3:MEDIAN 4:GAUSSIAN	0	0	0
CMD_COLORBAR	0:OFF, 1:ON	0	0	0
CMD_TEMP_VIEW	0:OFF, 1:ON	0	0	0
CMD_TEMP_INDICATOR	0:OFF, 1:ON	0	0	0
CMD_TEMP_TYPE	0:Celsius, 1:Fahrenheit	0	0	0
CMD_TRANSPARENCY	0:0%, 1:20%, 2:40%, 3:60%, 4:80%	0	0	0
CMD_CAM_ID	1 ~ 255	0	0	0
CMD_BAUDRATE	0:2400, 1:4800, 2:9600, 3:19200, 4:38400	0	0	0
CMD_ETHERNET	0:STATIC, 1:DHCP	IP ADDRESS 0xFFFFFF00 (255.255.255.0)	NETMASK 0xFFFFFF00 (255.255.255.0)	GATEWAY 0xFFFFFF00 (255.255.255.0)
CMD_ALARM1_FUNC	0:OFF, 1:CENTER, 2:MEAN, 3:MAX, 4:MIN, 5:ON	0	0	0
CMD_ALARM1_COND	0:ABOVE, 1:BELOW	0	0	0
CMD_ALARM1_VAL	-20 ~ 120	0	0	0
CMD_ALARM1_DUR	0 ~ 99	0	0	0
CMD_ALARM2_FUNC	0:OFF, 1:CENTER, 2:MEAN, 3:MAX, 4:MIN, 5:ON	0	0	0
CMD_ALARM2_COND	0:ABOVE, 1:BELOW	0	0	0
CMD_ALARM2_VAL	-20 ~ 120	0	0	0
CMD_ALARM2_DUR	0 ~ 99	0	0	0

CMD_TV_MODE	0:NTSC, 1:PAL	0	0	0
CMD_NUC	0:1MIN, 1:5MIN, 2:10MIN, 3:30MIN, 4:60MIN, 5:OFF 7:Manual	0	0	0

CMD_TEMP_MODE	0:(-20~120°C)
	1:(-20~650°C)
		0	0	0

CMD_NETWORK_FPS	0: 60 frames 1: 60/2=30 frames ~ 59: 60/60=1 frame	0	0	0
CMD_TEMP_CORRECT	0:Disable, 1:Enable	Ref1	0	0
CMD_SPOT_CONF	HIBYTE: spot index(1~10) LOBYTE: bit flag (enable:0x01, use local correction:0x02)	Ref1	LOWORD: x HIWORD: y	
CMD_ISOTHERM_CONF	HIBYTE: index(1~2) LOBYTE: bit flag (enable:0x01)	Ref2	Ref3	0

Ref1) Temperature correction or Spot configuration (32bits)

24~31	16~23	0~15
Emissivity.(0~100)	Atmospheric Transmission.(0~100)	Atmosphere Temp. (x10)

Ref2) Isotherm configuration (isotherm's color) (32 bits)

4th byte	3rd byte	2nd byte	1st byte
Reserved	Blue	Green	Red

Ref3) Isotherm configuration

HI WORD	LO WORD

GetIRImages Method

Get a sequence of IR raw data from the camera with TCP/IP.

Syntax

```
shortGetIRImages(
   HANDLE  handle,
   UINT* pTimerID,
   IRF_IR_CAM_DATA_T* cam_data
);
```

Parameters	**Description**
Return value	Return code. Use GetError to convert code to string.
handle	Socket handle. [in]
pTimerID	Timer ID using to send alive packet to a camera. [in]
cam_data	Sending message type. [out]

GetImageLUT Method

Get a palette LUT (lookup table) to change from IR data to image data. If palette is 8bit, length of LUT is 256.

Syntax

```
shortGetImageLUT( unsi
   gned char* palette,
   IRF_PALETTE_TYPE_T paletteType,
   bool bInvert
);
```

Parameters	**Description**
Return value	Return code. Use GetError to convert code to string.
palette	RGB color buffer. [out]
paletteType	Palette type (Grey, Rainbow, Medical and etc.,). [in]
bInvert	Color invert. [in]

GetTempMapTable Method

Get temperature map table to get temperature.

Syntax

```
shortGetTempMapTable(
   float* tempLUT,
   IRF_DYNAMIC_RANGE_T tempMode
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
tempLUT	16 bits temperature LUT. [out]
tempMode	Temperature mode (Low/Middle/High range). [in]

GetImage Method

Get an 8 bits gray image from 16 bits IR image. Before call this function, must be called GetTempMapTable( ).

<IRF_IR_CAM_DATA_T::ir_image> and <IRF_IR_CAM_DATA_T::save_data> must be initialized.

<IRF_IR_CAM_DATA_T::image_buffer_size> must be 0.

Syntax

```
shortGetImage( unsigne
   d char* image,
   HANDLE ir_image,
   long image_size,
   float* tempLUT,
   float* level,
   float* span,
   IRF_AUTO_RANGE_METHOD_T* method
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
image	8 bits buffer array with image pixels. [out]
ir_image	Structure of IRF_IR_CAM_DATA_T .(image_buffer_size must be 0) [in]

image_size	Image size (NTSC : 320*240, PAL : 384*288). [in]
tempLUT	Image LUT
level	The center value of temperature scale. [in]
span	The interval of temperature scale. [in]

GetCorrectedImage Method

Get an 8 bits gray image from 16 bits IR image. Before call this function, must be called GetTempMapTable( ).

<IRF_IR_CAM_DATA_T::ir_image> and <IRF_IR_CAM_DATA_T::save_data> must be initialized.

<IRF_IR_CAM_DATA_T::image_buffer_size> must be 0.

Syntax

```
shortGetCorrectedImage(
   unsigned char* image,
   HANDLE  ir_image,
   long image_size,
   float* tempLUT,
   IRF_TEMP_CORRECTION_PAR_T corrPara,
   float* level,
   float* span,
   IRF_AUTO_RANGE_METHOD_T* method
);
```

Parameters	**Description**
Return value	Return code. Use GetError to convert code to string.
image	8 bits buffer array with image pixels. [out]
ir_image	Structure of IRF_IR_CAM_DATA_T .(image_buffer_size must be 0) [in]
image_size	Image size. [in]
tempLUT	Image LUT
corrPara	Temperature correction structure (emissivity, atmospheric temperature, and atmospheric transmission). [in]
level	The center value of temperature scale. [in]
span	The interval of temperature scale. [in]

GetGrayToPaletteImage Method

Change from 8 bits gray image to palette image. Buffer size of palette image is detected by device display bits.

Syntax

```
shortGetGrayToPaletteImage(
     unsigned char* from_image,
    void* to_image,
    unsigned short width,
    unsigned short height,
    unsigned char* palette,
    int BitsPixel,
    bool bMirror,
    bool bFlip
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
from_image	8 bits gray image buffer. [in]
to_image	Color image buffer with RGB color palette. [out]
width	Image width. [in]
height	Image height. [in]
palette	RGB color palette buffer. [in]
BitsPixel	Number of device display bits (16/24/32 bits). [in]
bMirror	Image mirror. (true: on, false: off) [in]
bFlip	Image flip.(true: on, false: off)[in]

GetIRHistogram Method

Calculate histogram from IR image buffer.

Syntax

```
shortGetIRHistogram( unsig
    ned int* hist, unsigned
    short* ir_image, long
    image_size
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
hist	Histogram buffer. [out]

ir_image	IR image buffer with IR Raw values. [in]
Image_size	Image size (NTSC : 320*240, PAL : 384*288). [in]

GetTempRangeValue Method

Get minimum and maximum temperature for dynamic temperature range. (Low/Middle/High range).

Syntax

```
shortGetTempRangeValue(IRF_DYNAMIC
  _RANGE_T* tempMode, short* min,
  short* max
);
```

Parameters	**Description**
Return value	Return code. Use GetError to convert code to string.
tempMode	Dynamic temperature range (Low/Middle/High). [in]
min	Minimum temperature of selected dynamic range. [out]
max	Maximum temperature of selected dynamic range. [out]

ConvertFahToCels Method

Convert from Fahrenheit to Celsius.

Syntax

```
floatConvertFahToCels(float
  temp
);
```

Parameters	**Description**
Return value	Temperature in Celsius.
temp	Temperature in Fahrenheit. [in]

ConvertCelsToFah Method

Convert from Celsius to Fahrenheit.

Syntax

```
float ConvertCelsToFah(
   float temp
);
```

Parameters	Description
Return value	Temperature in Fahrenheit.
temp	Temperature in Celsius. [in]

ConvertKelvToCels Method

Convert from Kelvin to Celsius.

Syntax

```
floatConvertKelvToCels(floa
   t temp
);
```

Parameters	Description
Return value	Temperature in Celsius.
temp	Temperature in Kelvin. [in]

ConvertKelvToFah Method

Convert from Kelvin to Fahrenheit.

Syntax

```
floatConvertKelvToFah(float
   temp
);
```

Parameters	Description
Return value	Temperature in Fahrenheit.
temp	Temperature in Kelvin. [in]

ConvertCelToKel Method

Convert from Celsius to Kelvin.

Syntax

```
float ConvertCelToKel(
    float temp
);
```

Parameters	Description
Return value	Temperature in Kelvin.
temp	Temperature in Celsius. [in]

ConvertFahToKel Method

Convert from Fahrenheit to Kelvin.

Syntax

```
floatConvertFahToKel(flo
    at temp
);
```

Parameters	Description
Return value	Temperature in Kelvin.
temp	Temperature in Fahrenheit. [in]

GetPointTemp Method

Get a temperature of a pixel from IR raw image.

<IRF_IR_CAM_DATA_T::ir_image> and <IRF_IR_CAM_DATA_T::save_data> must be initialized.

<IRF_IR_CAM_DATA_T::image_buffer_size> must be 0.

Syntax

```
floatGetPointTemp(H
    ANDLE ir_image,
    IRF_IMAGE_INFO_T image_info,
    float* tempLUT,
    IRF_CORRECTION_PAR_T corrPara,
    POINT pt
);
```

Parameters	Description
Return value	Temperature of a point.
ir_image	Structure of IRF_IR_CAM_DATA_T . (image_buffer_size must be 0) [in]

image_info	Image information structure (bit, width and height). [in]
tempLUT	Temperature LUT. [in]
corrPara	Temperature correction structure (emissivity, atmospheric temperature, and atmospheric transmission). [in]
pt	Position structure (x, y). [in]

GetPointTemp Method

Get a temperature of a pixel from IR raw image.

<IRF_IR_CAM_DATA_T::ir_image> and <IRF_IR_CAM_DATA_T::save_data> must be initialized.

<IRF_IR_CAM_DATA_T::image_buffer_size> must be 0.

Syntax

```
Float GetIRdataToTemp

( HANDLE ir_data,
   float* tempLUT,
   IRF_CORRECTION_PAR_T corrPara,
);
```

Parameters	**Description**
Return value	Temperature of a point.
ir_data	Structure of IRF_IR_CAM_DATA_T . (image_buffer_size must be 0) [in]
tempLUT	Temperature LUT. [in]
corrPara	Temperature correction structure (emissivity, atmospheric temperature, and atmospheric transmission). [in]

GetNeighborPointTemp Method

Get an average temperature of neighbor pixels from IR raw image.

<IRF_IR_CAM_DATA_T::ir_image> and <IRF_IR_CAM_DATA_T::save_data> must be initialized.

<IRF_IR_CAM_DATA_T::image_buffer_size> must be 0.

Syntax

```
floatGetNeighborPointTemp

(HANDLE ir_image,
   IRF_IMAGE_INFO_T image_info,
   float* tempLUT,
   IRF_CORRECTION_PAR_T corrPara,
   POINT pt
);
```

Parameters	Description
Return value	Temperature of a point.
ir_image	Structure of IRF_IR_CAM_DATA_T . (image_buffer_size must be 0) [in]
image_info	Image information structure (bit, width and height). [in]
tempLUT	Temperature LUT. [in]
corrPara	Temperature correction structure (emissivity, atmospheric temperature, and atmospheric transmission). [in]
pt	Position structure (x, y). [in]

GetROITemp Method

Get temperature of minimum, maximum, average, and standard deviation in a ROI from IR image.

<IRF_IR_CAM_DATA_T::ir_image> and <IRF_IR_CAM_DATA_T::save_data> must be initialized.

<IRF_IR_CAM_DATA_T::image_buffer_size> must be 0.

Syntax

```
shortGetROITemp( H
   ANDLE ir_image,
   IRF_IMAGE_INFO_T image_info,
   float* tempLUT,
   IRF_CORRECTION_PAR_T corrPara,
   RECT roi,
   IRF_NUMERIC_INFO_T* numInfo,
   POINT* min_pt,
   POINT* max_pt
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
ir_image	Structure of IRF_IR_CAM_DATA_T . (image_buffer_size must be 0) [in]
image_info	Image information structure (bit, width and height). [in]
tempLUT	Temperature LUT. [in]
corrPara	Temperature correction structure (emissivity, atmospheric temperature, and atmospheric transmission). [in]
roi	Region of interest (x, y, width, and height). [in]
numInfo	Numeric information structure (min, max, average, and S.D.). [out]

min_pt	Minimum temperature position(x, y). [out]
max_pt	Maximum temperature position(x, y). [out]

GetRawToTemp Method

Get corrected temperature data from IR image.

<IRF_IR_CAM_DATA_T::ir_image> and <IRF_IR_CAM_DATA_T::save_data> must be initialized.

<IRF_IR_CAM_DATA_T::image_buffer_size> must be 0.

Syntax

```
shortGetRawToTemp(
   HANDLE ir_image,
   IRF_IMAGE_INFO_T image_info,
   float* tempLUT,
   IRF_CORRECTION_PAR_T corrPara,
   float* tempImage
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
ir_image	Structure of IRF_IR_CAM_DATA_T . (image_buffer_size must be 0) [in]
image_info	Image information structure (bit, width and height). [in]
tempLUT	Temperature LUT. [in]
corrPara	Temperature correction structure (emissivity, atmospheric temperature, and atmospheric transmission). [in]
tempImage	Corrected temperature data. [out]

GetCorrectedTemp Method

Get a corrected temperature of a pixel from correction parameter.

Syntax

```
floatGetCorrectedTemp(floa
   t* tempLUT,
   IRF_TEMP_CORRECTION_PAR_T corrPara,
   unsigned short engineOut,
);
```

Parameters	Description
Return value	Corrected temperature of a point.
tempLUT	Temperature LUT. [in]
corrPara	Temperature correction structure (emissivity, atmospheric temperature, and atmospheric transmission). [in]
engineOut	IR raw data. [in]

ApplyImageFilter Method

Apply image filter in 8 bits image.

Syntax

```
shortApplyImageFilter( unsigne
   d char* image, unsigned
   short width, unsigned short
   height,
   IRF_IMAGE_FILTER_T filter
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
image	8 bits image buffer. [in, out]
width	Image width. [in]
height	Image height. [in]
filter	Image filter (none, soften, and sharpening). [in]

ApplyColorImageFilter Method

Apply image filter in color image.

Syntax

```
shortApplyColorImageFilter(
   void* image,
   unsigned short width,
   unsigned short height,
   IRF_IMAGE_FILTER_T filter,
   int bitPixel
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
image	Color image buffer for bits each other (16/24/32). [in, out]
width	Image width. [in]
height	Image height. [in]
filter	Image filter (none, soften, and sharpening). [in]
bitPixel	Number of device display bits (16/24/32 bits). [in]

BilateralFilter Method

Apply bilateral filter in gray image.

Syntax

```
voidBilateralFilter( unsig
   ned char* image,
   unsigned short width,
   unsigned short height,
   float sigD,
   float sigR,
   int w
);
```

Parameters	Description
Return value	
image	8 bits gray image buffer. [in, out]
width	Image width. [in]
height	Image height. [in]
sigD	Spatial sigma. [in]
sigR	Edge sigma. [in]
w	Half-width of window. [in]

GetGaussianKernel Method

Get Gaussian kernel and make a map table to do quickly calculation before call FastGaussianBlur function.

Syntax

```
void GetGussianKernel(
    int* kernel,
    int* mult,
    int sz
);
```

Parameters	Description
Return value	
kernel	Gaussian kernel (buffer size : kernel size). [out]
mult	Map table (buffer size : kernel size*256). [out]
sz	Half-width of kernel (if kernel size is 5, sz is 2). [in]

FastGaussianBlur Method

Apply Fast Gaussian Blur filter in 8 bits gray image. Before call this function, must called GetGaussianKernel function.

Syntax

```
shortFastGaussianBlur(
    BYTE* img,
    int iw,
    int ih,
    int* Gkernel,
    int* Gmult,
    int radius
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
img	8 bits gray image buffer. [in, out]
iw	Image width. [in]
ih	Image height [in]
Gkernel	Gaussian kernel. (called GetGaussianKernel function) [in]
Gmult	Map table. (called GetGaussianKernel function) [in]
radius	Half-width of kernel (if kernel size is 5, radius is 2). [in]

FastStackBlur Method

Apply Fast Stack Blur filter in 8 bits gray image.

Syntax

```
shortFastStackBlur(
   BYTE* img,
   int w,
   int h,
   int radius
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
img	8 bits gray image buffer. [in, out]
iw	Image width. [in]
ih	Image height [in]
radius	Half-width of kernel (if kernel size is 5, radius is 2). [in]

BoxBlur Method

Apply Box Blur filter in 8 bits gray image.

Syntax

```
shortBoxBlur(
   BYTE*  img,
   int src_w,
   int src_h,
   int sz
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
img	8 bits gray image buffer. [in, out]
src_w	Image width. [in]
src_h	Image height [in]
sz	Half-width of kernel (if kernel size is 5, sz is 2). [in]

LoadIRImage Method

Read IR images from a file stream.

Syntax

```
voidLoadIRImage( HAN
   DLE* handle, char*
   FileName unsigned
   long* totSize,
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
handle	FILE stream pointer. [in, out]
FileName	IR file name (*.crd). [in]
totSize	Total size of IR file. [out]

GetIRImageFromStream Method

Get IR raw data from an IR file. Before call this function, must called LoadIRImage function.

Syntax

```
shortGetIRImageFromStream(
   HANDLE handle,
   unsigned short* ir_image
   long image_size,
   unsigned long totStreamSize,
   unsigned long* curPos,
   int* gap_time,
   bool bLoop
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
handle	FILE pointer handle. [in]
ir_image	16 bits IR image buffer. [out]
image_size	Image buffer size (NTSC : 320*240, PAL : 384*288) [in]
totStreamSize	Total stream size of an IR file. [in]
curPos	Current position of file stream. [in, out]
gap_time	Get time between frames from a file stream (millisecond). [out]
bLoop	Play loop (true : on, false : off) [in]

GetIRImageFromStream_n Method

Get IR raw data from an IR file. Before call this function, must called LoadIRImage function.

Syntax

```
shortGetIRImageFromStream_n(
   HANDLE handle,
   unsigned short* ir_image
   long image_size,
   unsigned long totStreamSize,
   unsigned long* curPos,
   int* gap_time,
   bool bLoop,
   bool new_ver
);
```

Parameters	**Description**
Return value	Return code. Use GetError to convert code to string.
handle	FILE pointer handle. [in]
ir_image	16 bits IR image buffer. [out]
image_size	Image buffer size (NTSC : 320*240, PAL : 384*288) [in]
totStreamSize	Total stream size of an IR file. [in]
curPos	Current position of file stream. [in, out]
gap_time	Get time between frames from a file stream (millisecond). [out]
bLoop	Play loop (true : on, false : off) [in]
new_ver	If file's version is less than 17, the new_ver is true.

GetIRImageFromStream_v2 Method

Get IR raw data from an IR file. Before call this function, must called LoadIRImage function.

Syntax

```
shortGetIRImageFromStream_v2(
   HANDLE handle,
   unsigned short* ir_image
   long image_size,
   unsigned long totStreamSize,
   unsigned long* curPos,
   int* gap_time,
   __int64* curTime,
   bool bLoop,
   unsigned char ver
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
handle	FILE pointer handle. [in]
ir_image	16 bits IR image buffer. [out]
image_size	Image buffer size (NTSC : 320*240, PAL : 384*288) [in]
totStreamSize	Total stream size of an IR file. [in]
curPos	Current position of file stream. [in, out]
gap_time	Get time between frames from a file stream (millisecond). [out]
curTime	Get time when image is saved.[out]
bLoop	Play loop (true : on, false : off) [in]
ver	Set saved file's version.[in]

GetRevIRImageFromStream Method

Get a reverse sequence of IR raw data from an IR file. Before call this function, must called LoadIRImage function.

Syntax

```
shortGetRevIRImageFromStream(
    HANDLE handle,
    unsigned short* ir_image
    long image_size,
    unsigned long* curPos,
    int* gap_time
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
handle	FILE pointer handle. [in]
ir_image	16 bits IR image buffer. [out]
image_size	Image buffer size (NTSC : 320*240, PAL : 384*288) [in]
curPos	Current position of file stream. [in, out]

gap_time	Get time between frames from a file stream. [out]

GetRevIRImageFromStream_n Method

Get a reverse sequence of IR raw data from an IR file. Before call this function, must called LoadIRImage function.

Syntax

```
shortGetRevIRImageFromStream_n( HANDL
   E handle,
   unsigned short* ir_image
   long image_size,
   unsigned long* curPos,
   int* gap_time,
   bool new_ver
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
handle	FILE pointer handle. [in]
ir_image	16 bits IR image buffer. [out]
image_size	Image buffer size (NTSC : 320*240, PAL : 384*288) [in]
curPos	Current position of file stream. [in, out]
gap_time	Get time between frames from a file stream. [out]
new_ver	If file's version is less than 17, the new_ver is true.

GetRevIRImageFromStream_v2 Method

Get a reverse sequence of IR raw data from an IR file. Before call this function, must called LoadIRImage function.

Syntax

```
shortGetRevIRImageFromStream_v2( HAND
   LE handle,
   unsigned short* ir_image
   long image_size,
   unsigned long* curPos,
   int* gap_time,
   __int64* curTime,
   unsigned char ver
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
handle	FILE pointer handle. [in]
ir_image	16 bits IR image buffer. [out]
image_size	Image buffer size (NTSC : 320*240, PAL : 384*288) [in]
curPos	Current position of file stream. [in, out]
gap_time	Get time between frames from a file stream. [out]
curTime	Get time when image is saved.[out]
ver	Set saved file's version.[in]

SaveIRImage Method

Get a file stream handle and save IR file header.

Syntax

```
short   SaveIRImage
   (HANDLE*  handle,
   char* filename,
   IRF_IR_FILE_HEADER_T* pHeader
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
handle	FILE pointer handle. [out]
filename	File name. [in]
pHeader	IR header information. [in]

SaveIRHeader Method

Get a file stream handle and save IR file header.

Syntax

```
shortSaveIRImage(
   HANDLE* handle,
   char* filename,
   IRF_IR_FILE_HEADER_T* pHeader,
   IRF_IR_DATA_HEADER_T *pAddedData
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
handle	FILE pointer handle. [out]
filename	File name. [in]
pHeader	IR header information. [in]
pAddedData	IR data header information. [in]

SetIRImageToStream Method

Save IR raw data to a file with file pointer handle. Before call this function, must called SaveIRImage function.

Syntax

```
short SetIRImageToStream
    (HANDLE handle,
    unsigned short* ir_image,
    long image_size
    int millisecond
    short* frameCnt
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
handle	FILE pointer handle. [in]
ir_image	IR raw data buffer. [in]
image_size	Image size (width * height). [in]
millisecond	Time between frames (30frame => 33 millisecond). [in]
frameCnt	Saved frame number.(A file size < 9600 frames) [in]

SetIRImageToStream_v2 Method

Save IR raw data to a file with file pointer handle. Before call this function, must called SaveIRImage function.

Syntax

```
short SetIRImageToStream_v2
    (HANDLE handle,
    unsigned short* ir_image,
    long image_size
    int millisecond
    short* frameCnt,
    unsigned char ver
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
handle	FILE pointer handle. [in]
ir_image	IR raw data buffer. [in]
image_size	Image size (width * height). [in]
millisecond	Time between frames (30frames => 33 millisecond). [in]
frameCnt	Saved frame number.(A file size < 9600 frames) [in]
ver	File version <17:IRF_IR_FILE_HEADER_T, 17:IRF_IR_FILE_HEADER_T+IRF_IR_DATA_HEADER_T 20:added frame header

CloseIRStream Method

Close IR file stream.

Syntax

```
short SaveIRImage
    ( HANDLE handle
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
handle	FILE pointer handle. [in]

DrawColorBar Method

Draw color-bar.

Syntax

```
void DrawColorBar
   ( HWND hWnd,
   HDC hDC,
   unsigned char* palette,
   float level,
   float span,
   IRF_TEMP_MODE_T tempUnit,
   bool bUpdateOnlyTickArea
);
```

Parameters	Description
Return value	
hWnd	Window handle. [in]
hDC	Device context handle. [in]
Palette	Color palette buffer. [in]
Level	The center value of temperature scale. [in]
Span	The interval of temperature scale. [in]
tempUnit	Temperature units (Celsius/Fahrenheit/Kelvin). [in]
bUpdateOnlyTickArea	Redraw only tick area (true : on, false : off). [in]

DrawMinMaxPos Method

Draw minimum and maximum temperature position.

Syntax

```
void DrawMinMaxPos
   ( HDC hDC,
   POINT  minP,
   POINT maxP
   Int size
);
```

Parameters	Description
Return value	
hDC	Device context handle. [in]
minP	Minimum position. [in]
maxP	Maximum position. [in]
size	Draw size. [in]

GetError Method

Convert return code or error code to formatted string.

Syntax

```
void GetError
   ( short code,
   LPCTSTR msg,
);
```

Parameters	Description
Return value	Return code. Use GetError to convert code to string.
code	Return code or error code. [in]
msg	Message. [out]

5. Description of the image headers

The image information structure

```
typedef struct
{
   unsigned short xSize;              /* horizontal size of infrared data */
   unsigned short ySize;              /* vertical size of infrared data */
} IRF_IMAGE_INFO_T;
```

The temperature correction structure

```
typedef struct
{
   float emissivity;                  /* 0.01 - 1.0 */
   float atmTemp;                     /* Atmospheric temperature in Celsius */
   float atmTrans;                    /* Atmospheric Transmission in Celsius */
} IRF_TEMP_CORRECTION_PAR_T;
```

Ref)

- *Calculated atmospheric transmission : A transmission value computed from the temperature, the relative humidity of the air, and the distance to the object*
- *Emissivity : The amount of radiation coming from an object compared to that of a blackbody.*

The IR file header structure

```
typedef struct
{
   BYTE ver;
   IRF_IMAGE_INFO_T image_info;
   IRF_TEMP_CORRECTION_PAR_T  temp_correction;
} IRF_IR_FILE_HEADER_T;

typedef struct
{
          BYTE dynamic_range;                       /* IRF_DYNAMIC_RANGE_T */
          int year;                                 /* File creation year */
          int month;
```

```
        int day;
        int hour;int
        minute; int
        second;
        int total_frame;                  /* Total frame number */
        IRF_SAVEDATA_T  save_data;                /* cam data in CAM_DATA*/
        BYTE reserved[460];
} IRF_IR_DATA_HEADER_T;
```

The AGC setting structure

```
/* AGC method */
typedef enum
{
   _IRF_MIN_MAX,                /* MinMax Algorithm */
   _IRF_BRIGHTNESS_RATE,        /* Brightness Rate (%) */
   _IRF_SD_RATE,                /* Standard Deviation Rate (%) */
   _IRF_AUTO_BRIGHT,            /* Auto Brightness */
   _IRF_ENHANCE_HIST
}IRF_AUTO_RANGE_INPUT_METHOD_T;


typedef enum
{
   _IRF_LINEAR,                 /* Linear method. (contrast + brightness) */
   _IRF_NON_LINEAR,             /* Non-Linear method. (Gamma function) */
   _IRF_TPE,                    /* Tail-less Plateau Equalization. */
   _IRF_APE,                    /* Adaptive Plateau Equalization. */
   _IRF_SAPE                    /* Self-adaptive plateau equalization. */
}IRF_AUTO_RANGE_OUTPUT_METHOD_T;


typedef enum
{
   _IRF_AUTO,
   _IRF_MANUAL
} IRF_AUTOMATIC_TYPE_T;
```

```
typedef struct
{
   IRF_AUTOMATIC_TYPE_T autoScale;                              /* Automatic scale. */
   IRF_AUTO_RANGE_INPUT_METHOD_T inputMethod;
   IRF_AUTO_RANGE_OUTPUT_METHOD_T  outputMethod;
   float B_Rate;                        /* Brightness rate (0 <= B_Rate <= 1.0). */
   float SD_Rate;                       /* SD rate (1.0 <= SD_Rate <= 6.0). */
   unsigned char intercept;             /* intercept of linear method (0 <= intercept <= 254). */
   float gamma;                         /* Gamma of non-linear method.   (0.1 <= gamma <= 25) */
   unsigned int plateau;                /* plateau value for tail-less plateau equalization. */
   float epsilon;               /* The epsilon that is threshold value is a scalar arbitrary set to a value
                                     between zero and one. (Adaptive Plateau Algorithm) */
   float psi;                   /* The psi is a scalar arbitrary set to a value between zero and one.
                                   (Adaptive Plateau Algorithm) */
   float prevPalteau;           /* previous plateau value for using Adaptive Plateau Algorithm. */
} IRF_AUTO_RANGE_METHOD_T;
```

The structure for transfer command to camera

```
typedef struct {
   unsigned short CMD;     // AGC, Level, Span, etc.
   unsigned short Value;   // on/off, color palette, osd transparency, baud rate, zoom, nuc
   DWORD Value2;           // ip address
   DWORD Value3;           // netmask
   DWORD Value4;           // gateway
   BYTE Reserved[16];
} IRF_SET_CAM_DATA_T;
```

The user palette transfer structure

```
typedef struct {
   BYTE Info[7];                  // Reserved
   BYTE Index;                    // Data Index (0:userPalette1, 1:userPalette2)
   BYTE Data[1024];               // RGBA (4bytes * 256 level)
   unsigned int pngLength;        // PNG File length
   BYTE pngData[8192];            // PNG File data;
} IRF_SET_USER_PALETTE;
```

The numeric information structure

```
typedef struct
{
   float min;                        /* Minimum temperature */
   float max;                        /* Maximum temperature */
   float avg;                        /* Average temperature */
   float std;                        /* Standard deviation temperature */
} IRF_NUMERIC_INFO_T;
```

The camera setting information structure

```
#define SAVEDATA_VERSION          0x11      /* Save structure version    */

typedef union strSAVEDATA
{
   struct
   {
         uint32_t    crc;            /* CRC check */
         uint8_t     ver;            /* Save structure version */
         uint8_t     id;             /* Camera ID(0~255) for RS-485 */
         uint8_t     sensor;         /* 0:320, 1:640 */
         uint8_t     tv;             /* NTSC/PAL */
         uint8_t     temp_mode;      /* Temperature mode (normal/high mode) */
         uint8_t     id;             /* ID */
         uint8_t     baudrate;       /* Baud rate for RS-485 */
         int16_t     level;          /* Level */
         uint16_t    span;           /* Span */
         uint8_t     agc;            /* Automatic gain control */
         uint8_t     invert;         /* Image invert */
         uint8_t     mirror;         /* Image mirror */
         uint8_t     flip;           /* Image flip */
         uint8_t     colorbar;       /* Color-bar display */
         uint8_t     showinfo;       /* Temperature information display */
         uint8_t     indicator;      /* Min/Max temperature position display */
         uint8_t     unit;           /* Temperature units */
         uint8_t     dhcp;           /* DHCP setting */
```

```
uint8_t     color;              /* Palette selection */
uint8_t     alpha;              /* OSD alpha */
uint8_t     zoom;               /* Zoom */
uint8_t     sharp;              /* Image filter : sharpness */
uint8_t     noise;              /* Image filter : noise reduction */
uint16_t    nuc;                /* NUC setting */
uint8_t     econt;              /* E-Contrast */
uint32_t    ipaddr;             /* IP address setting */
uint32_t    netmask;            /* Net address setting */
uint32_t    gateway;            /* Gateway setting */
uint32_t    dns;                /* Domain Name Server */
uint8_t     alarm1_func;        /* Function setting of alarm 1 */
uint8_t     alarm1_cond;        /* Condition setting of alarm 1 */
uint16_t    alarm1_value;       /* Value setting of alarm 1 */
uint8_t     alarm2_func;        /* Function setting of alarm 2 */
uint8_t     alarm2_cond;        /* Condition setting of alarm 2 */
uint16_t    alarm2_value;       /* Value setting of alarm 2 */
uint8_t     down_filter;
uint8_t     show_center;        /* Display center indicator */
uint8_t     show_spot;          /* Display spot indicator */
uint8_t     show_correction; /* Display temperature correction parameters */
uint8_t     show_isotherm;   /* Display isotherm */
uint8_t     alarm1_duration; /* Alarm1 duration (0 ~ 99 seconds) */
uint8_t     alarm2_duration; /* Alarm2 duration (0~ 99 seconds) */
  struct {
            uint8_t flag; //enable:0x01 exclude:0x02
            uint16_t x1;      //position (x2)
            uint16_t y1;
            uint16_t x2;
            uint16_t y2;
  } roi[2];
uint8_t     reserved1[48];

uint8_t     limit9;             /* 0:9Hz device, 1:60Hz device */
uint8_t     enable_high;        /* 0: only normal device, 1: normal and high mode device */
uint8_t     correction;
uint8_t     emissivity;         /* Emissivity */
uint8_t     transmission;       /* Transmission */
```

```
        int16_t      atmosphere;        /* Atmospheric temperature */

        struct {                        /* spot structure */
           uint8_t   enable;            /* spot selection */
           uint16_t x;                  /* spot x coordinate */
           uint16_t y;                  /* spot y coordinate */
           uint8_t   local;             /* use local object parameters */
           uint8_t   em;                /* spot's emissivity */
           uint8_t   tr;                /* spot's transmission */
           int16_t   at;                /* spot's atmospheric temperature */
           uint8_t   reserved[6];
        } spot[10];

        struct {                        /* Isotherm structure */
           uint8_t   enable;            /* Enable isotherm */
           uint32_t seed_color;         /* Isotherm color */
           int16_t   top;               /* Above temperature of isotherm */
           int16_t   bottom;            /* Below temperature of isotherm */
           uint8_t   reserved[3];       /* Reserved */
        } iso[3];

uint8_t reserved2[53];
uint8_t reserved3[128];
} IRF_SAVEDATA_T;
```

The structure to process received data from TCP/IP

```
typedef enum
{
      _IRF_NONE = -1,
      _IRF_ACK,
      _IRF_NAK,
      _IRF_ALIVE,
      _IRF_STREAM_ON,              /* Request to start raw data transfer. */
      _IRF_STREAM_OFF,             /* Request to stop raw data transfer. */
      _IRF_STREAM_DATA,
      _IRF_BROADCAST,
      _IRF_REQ_CAM_DATA,           /* Request all camera setting value. */
```

```
    _IRF_CAM_DATA,              /* Received all camera setting value. */
    _IRF_SAVE_CAM_DATA,         /* Request to do save camera setting value. */
    _IRF_SET_CAM_DATA,          /* Set camera unit function setting. */
    _IRF_SET_USER_PALETTE,      /* User color palette update. (pc --> cam) */
    _IRF_REQ_SYS_INFO,          /* Request System Information. (pc --> cam) */
    _IRF_SYS_INFO,              /* Get System Information. (cam --> pc) */
    _IRF_SPOT_STREAM_ON,        /* Start spot streaming. (cam --> pc) */
    _IRF_SPOT_STREAM_OFF,       /* Stop spot streaming. (cam --> pc) */
    _IRF_SPOT_STREAM_DATA       /* Spot Streaming Data */
} IRF_MESSAGE_TYPE_T;



typedef struct
{
  unsigned short* ir_image;     /* 16bits raw image data */
  DWORD image_buffer_size;      /* Raw image size. */
  LPBYTE lpNextData;            /* This variable is remainder data make next raw image after
                                     make a raw image data from communication buffer. */
  DWORD dwSize;                 /* This variable is size of reminder data. */
  DWORD dwPosition;             /* This variable is current position in the reminder data. */
  IRF_MESSAGE_TYPE_T msg_type;
  IRF_SAVEDATA_T save_data;
  unsigned int fw_ver;          /* Firmware version in SYS_INFO */

  unsigned short PMSGTYPE;      // Primary Message Type Code
  unsigned short RCODE;         // Response Code
} IRF_IR_CAM_DATA_T;
```

The temperature mode enumeration

```
typedef enum
{
  _IRF_CELSIUS,         /* Celsius */
  _IRF_FAHRENHEIT,      /* Fahrenheit */
  _IRF_KELVIN           /* Kelvin */
} IRF_TEMP_MODE_T;
```

The dynamic range enumeration

```
typedef enum
{
   _IRF_LOW_RANGE,         /* -20 ~ 120 */
   _IRF_MIDDLE_RANGE,
   _IRF_HIGH_RANGE
} IRF_DYNAMIC_RANGE_T;
```

The image filter enumeration

```
typedef enum
{
   _IRF_FILTER_NONE,                  /* No filter */
   _IRF_MEDIAN,                       /* Median filter */
   _IRF_SOFTEN_SLIGHTLY,              /* Soften slightly */
   _IRF_SOFTEN_STRONG,                /* Soften strong */
   _IRF_SHARPENING_SLIGHTLY,          /* Sharpening slightly */
   _IRF_SHARPENING_STRONG,            /* Sharpening strong */
   _IRF_BOXBLUR,                      /* Box blur filter */
   _IRF_FAST_GAUSSIAN,                /* Fast Gaussian filter */
   _IRF_FAST_STACK_BLUR               /* Fast stack blur filter */ ,
   _IRF_BI_LATERAL                    /* Bi-Lateral filter */
} IRF_IMAGE_FILTER_T;
```

The palette type enumeration

```
typedef enum
{
   YELLOW_COLOR_MAP,
   RAINBOW_COLOR_MAP,
   RAIN900_COLOR_MAP,
   RAIN10_COLOR_MAP,
   MIDGREY_COLOR_MAP,
   MIDGREEN_COLOR_MAP,
   MEDICAL_COLOR_MAP,
   IRON10_COLOR_MAP,
   IRON_COLOR_MAP,
   GREYRED_COLOR_MAP,
   GREY10_COLOR_MAP,
   GREY_COLOR_MAP,
   GLOWBOW_COLOR_MAP
} IRF_PALETTE_TYPE_T;
```

The palette type of camera

```
typedef enum
{
   GREY,
   RAINBOW,
   IRON,
   GREYRED,
   GLOWBOW,
   YELLOW,
   MIDGREY,
   MIDGREEN
} IRF_CAM_PALETTE_TYPE_T;
```

6. Error Code

Status or error code	String
0	OK. No error
-1	Handle error.
-2	File open error.
-3	File close error.
-4	IR image read error.
-5	File stream buffer allocation error.
-6	End of IR image stream.
-7	Start of IR image stream.
-8	Writing error of IR image stream.
-9	Incorrect version of WS2_32.dll
-10	Connection error from a camera.
-11	Disconnected from a camera.
-12	Unknown packet ID.
-13	Message sending error.
-14	First frame position error.
-15	Window size error of image filter.
-16	Count error of image frame.
-17	Palette file open error
-100	Received NAK message from a camera.
-1000	Buffer allocation error.