Hamamatsu
Driver Circuit for Image Sensor
Control Library

Function Specifications

Revision History

Revision	Date	Description
1	09/07/2014	Initial Release
2	10/12/2014	Updated/clarified the following sections: 6.1.7, 6.1.8, 6.1.9, 6.1.10, 6.1.11, 6.1.12, 6.1.13, 6.1.14, 6.1.26, 6.1.27, 6.1.28, 6.1.29, 6.1.30, 6.1.31, 6.1.32 , 6.1.33, 6.1.34, 6.1.35 All changes are indicated with change bars.
3	12/03/2014	Added IGAASetGain and IGAAGetGain function calls Updated the description of IGAAOpen, IGAAClose, IGAAStart, IGAAStop
4	07/24/2017	Added IGAACaptureToFile function call
5	09/01/2017	Updated Table 1.

Table of Contents

Table of Tables

1 Overview

The Driver Circuit for Image Sensor Control Library (Hereafter abbreviated as "IGAA") is a library for controlling the Driver Circuit for Image Sensor. With this library, software for controlling the operations of the Driver Circuit can be easily developed.

2 Operating Environment

- **Supported OS**
  Microsoft® Windows XP® (Service Pack 3 or later)
  Microsoft® Windows 7® (Service Pack 1 or later)

- **CPU**
  In accordance with the recommended OS.

- **Memory**
  In accordance with the recommended OS.

3 Development Environment Construction

Please use the copy of IGAA.dll in the prescribed folder of the development environment.
A copy of IGAA.h, IGAAStatusCode.h and IGAA.lib will be required when the need arises.

4 Required Files

The library consists of the following files:

Library file:	IGAA.dll
Header file:	IGAA.h, IGAAStatusCode.h
Import library :	IGAA.lib
Driver :	cyusb3.sys
Driver information file:	cyusb3.inf

5 Function List

The following functions are available within IGAA.dll:

1	IGAAInitialize	Initializes the library
2	IGAAUninitialize	Unloads the library resources and closes the device driver
3	IGAAOpen	Opens the device
4	IGAAClose	Closes the device
5	IGAAGetDeviceState	Retrieves the device state type
6	IGAAGetImageSize	Retrieves the width and height of the one frame data
7	IGAASetPixelMode	Sets the Pixel Mode
8	IGAAGetPixelMode	Retrieves the Pixel Mode
9	IGAASetInterfaceSpeed	Sets the Interface Speed
10	IGAAGetInterfaceSpeed	Retrieves the Interface Speed
11	IGAASetInterfaceStyle	Sets the Interface Style
12	IGAAGetInterfaceStyle	Retrieves the Interface Style
13	IGAASetMuxControl	Sets the Mux Control Polarity
14	IGAAGetMuxControl	Retrieves the Mux Control Polarity
15	IGAAGetCaptureBytes	Retrieves the number of bytes of one captured image
16	IGAAGetTotalCaptureBytes	Retrieves the total number of bytes received in capturing images
17	IGAACapture	Return one image received in capturing images from the device, if capturing started
18	IGAACaptureBuffer	Return data received in capturing image from the device
19	IGAAStart	Starts to capture images from the device
20	IGAAStop	Stops capturing the image
21	IGAAWait	Waits till the image is captured
22	IGAASetPixelClockDivider	Sets the Pixel clock divider value
23	IGAAGetPixelClockDivider	Retrieves the Pixel clock divider value
24	IGAASetExposureTime	Sets the exposure time
25	IGAAGetExposureTime	Retrieves the exposure time
26	IGAASetDigitalPotentiometer1	Sets the Digital Potentiometer 1 value
27	IGAAGetDigitalPotentiometer1	Retrieves the Digital Potentiometer 1 value
28	IGAASetDigitalPotentiometer2	Sets the Digital Potentiometer 2 value
29	IGAAGetDigitalPotentiometer2	Retrieves the Digital Potentiometer 2 value
30	IGAAGetAtoDChannel0	Retrieves the Analog to Digital channel 0 converter value
31	IGAAGetAtoDChannel1	Retrieves the Analog to Digital channel 1 converter value
32	IGAAGetAtoDChannel2	Retrieves the Analog to Digital channel 2 converter value
33	IGAASetDtoAConverter	Sets the Digital to Analog converter value
34	IGAAGetDtoAConverter	Retrieves the Digital to Analog converter value
35	IGAASetPatternGenerator	Sets the Pattern Generator Mode
36	IGAAGetPatternGenerator	Retrieves the Pattern Generator Mode
37	IGAAGetVersion	Retrieves the library version number, in string format
38	IGAAGetDriverVersion	Retrieves the driver version number, in string format
39	IGAAGetFirmwareVersion	Retrieves the firmware version number, in a character string format
40	IGAAGetLastError	Retrieves the last-error code
41	IGAAGetDebugInformation	Retrieves the Debug Information string
42	IGAAGetDebugStream	Retrieves the Debug Numbers
43	IGAAGetRegisters	Retrieves the Device Registers
44	IGAASetRegisters	Sets the Device Registers

45	IGAASetGain	Sets the sensor gain
46	IGAAGetGain	Retrieves the senor gain
47	IGAACaptureToFile	Captures the specified number of lines into a file

5.1 Parameter Definition

5.1.1 IGAA.h

[Device State]

IGAA_DEVSTATE_NON	Non-connection, No device found
IGAA_DEVSTATE_DEVICE	Non-connection, Device found
IGAA_DEVSTATE_NODEVICE	Connection, No device found
IGAA_DEVSTATE_CONNECT	Connection, Device found
IGAA_DEVSTATE_NOT_ACCESS	Connection, Device found but not accessible

[Pixel Mode]

IGAA_PMODE_256	256-pixel mode
IGAA_PMODE_512	512-pixel mode

[Interface Speed]

IGAA_ISPEED_NORMAL	Normal-speed Sensor Interface
IGAA_ISPEED_HIGH	High-speed Sensor Interface

[Interface Style]

IGAA_ISTYLE_SEQUENTIAL	Sequential sensor interface
IGAA_ISTYLE_PARALLEL	Parallel sensor interface

[Mux Control]

IGAA_MUXC_NORMAL	Normal multiplexor control (even pixel first)
IGAA_MUXC_INVERTED	Inverted polarity multiplexor control (odd pixel first)

[Pattern Generator Mode]

IGAA_PGM_OFF	Pattern generator is OFF
IGAA_PGM_ON	Pattern generator is ON

[Device Information]

IGAA_DEVINF_TYPE	Device Type
IGAA_DEVINF_SERIALNO	Serial Number of Device
IGAA_DEVINF_VERSION	Device Version

[USB Transfer Rate Type]

IGAA_TRANSRATE_USB11	USB 1.1 standard
IGAA_TRANSRATE_USB20	USB 2.0 standard
IGAA_TRANSRATE_USB30	USB 3.0 standard

[Sensor Gain]

IGAA_GAIN_LOW	Low Gain
IGAA_GAIN_HIGH	High Gain

5.2 Error Code Table

5.2.1 IGAAStatusCode.h

0	dcCode_Success	Normal termination
1	dcCode_Unknown	An unknown error has occurred
2	dcCode_NoInit	Library is not initialized
3	dcCode_AlreadyInit	Already in-use
4	dcCode_NoDriver	No driver was detected
5	dcCode_NoMemory	Memory is insufficient
6	dcCode_NotConnected	The device is not connected
9	dcCode_InvalidParam	Invalid parameter
100	dcCode_DeviceDefect	The device is not functioning
111	dcCode_Timeout	Timeout has occurred
120	dcCode_AlreadyStarted	Already started
121	dcCode_NotStarted	Not started
122	dcCode_CtrlStarted	Control Already started
123	dcCode_NoDebugInfo	Empty Debug Information
124	dcCode_NoImageData	No Image Data Received
125	dcCode_NoAccess	Can't read Device registers

6 Function Details

6.1 IGAA.dll

6.1.1 IGAAInitialize

bool IGAAInitialize(void)

[Summary]

Initializes the library by performing the following steps:

1. Get parent handle hWnd value.
2. Initialize USB structure data.
3. Initialize control and FPGA register buffers.
4. Create USB_Device

[Arguments]

None

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

Always run this function first before running other functions.
An error occurs if the library has already been initialized.
Only one process can use this library.
In addition, this function should be used in combination with the "IGAAUninitialize".

[Reference]

None

[Example]

```
DWORDdwErrCode;
if(IGAAInitialize() != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.2 IGAAUninitialize

bool IGAAUninitialize(void)

[Summary]

Unloads the library resources and closes the device driver (deletes USB_Device)

[Arguments]

None

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

This function should be used in combination with the "IGAAInitialize".
In addition,this function should be called when quitting the program or when this library is not required.

[Reference]

None

[Example]

```
DWORDdwErrCode;
if(IGAAUninitialize() != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.3 IGAAOpen

bool IGAAOpen(void)

[Summary]

Opens the device by performing the following steps:
1. Open USB_Device
2. Get control and data USB_Device EndPoints.
3. Read all FPGA registers into FPGA register buffer.
4. Start data input thread. (if Image data started before - begin data reception)
Note: This function does not change any of the FPGA registers.

[Arguments]

None

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORDdwErrCode;
// Initialize Library
if(IGAAInitialize() != TRUE){
dwErrCode = IGAAGetLastError();
return;
}
// Open Device
if(IGAAOpen() != TRUE){
dwErrCode = IGAAGetLastError();
return;
}*/
```

6.1.4 IGAAClose

bool IGAAOpen(void)

[Summary]

Closes the device by performing the following steps:

1. Stop data input thread.
2. Close USB_Device.

[Arguments]

None

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORDdwErrCode;
// Close Device
if(IGAAClose() != TRUE){
dwErrCode = IGAAGetLastError();
return;
}
// Uninitialize Library
if(IGAAUninitialize() != TRUE){
dwErrCode = IGAAGetLastError();
}*
```

6.1.5 IGAAGetDeviceState

bool IGAAGetDeviceState(int* pState)

[Summary]

Retrieves the device state type

[Arguments]

pState Specifies the address of the variable where the device state type is to be stored.
Any one of the following values is obtained.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nState;
DWORD dwErrCode;
// Get device state
if(IGAAGetDeviceState(&nState) != TRUE){
dwErrCode = IGAAGetLastError();
}
// Remove the device
if(nState == IGAA_DEVSTATE_NODEVICE){
IGAAClose();
}
IGAA_DEVSTATE_NON : Non-connection, No device found
IGAA_DEVSTATE_DEVICE : Non-connection, Device found
IGAA_DEVSTATE_NODEVICE : Connection, No device found
IGAA_DEVSTATE_CONNECT : Connection, Device found
IGAA_DEVSTATE_NOT_ACCESS : Connection, Device found but not accessable
(during the boot process)*/
```

6.1.6 IGAAGetImageSize

bool IGAAGetImageSize (int* pWidth, int* pHeight)

[Summary]

Retrieves the width and height of the one frame data.

[Arguments]

pWidth Specifies the address of the variable where the image width is to be stored.
pHeight Specifies the address of the variable where the image height is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

Call this function for obtaining the size of image after calling IGAASetBinning or IGAASetMeasureDataCount that can be helpful in determining the memory size to be allocated.

[Reference]

None

[Example]

```
INT nWidth = 0;
INT nHeight = 0;
DWORD dwErrCode;
// Get size of image
if(IGAAGetImageSize(&nWidth, &nHeight) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.7 IGAASetPixelMode

bool IGAASetPixelMode(int nMode)

[Summary]

Sets the Pixel Mode

[Arguments]

nMode specifies the pixel mode as per "IGAAUSB.h":
IGAA_PMODE_256: sets 256-pixel mode
IGAA_PMODE_512: sets 512 pixel mode

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORD dwErrCode;
if(IGAASetPixelMode(IGAA_PMOD_256) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.8 IGAAGetPixelMode

bool IGAAGetPixelMode(int* pMode)

[Summary]

Retrieves the Pixel Mode

[Arguments]

pMode specifies the address of the variable where the pixel mode that is currently set is to be stored.
The values are as per "IGAAUSB.h":
IGAA_PMODE_256: sets 256-pixel mode
IGAA_PMODE_512: sets 512 pixel mode

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nMode;
DWORD dwErrCode;
if(IGAAGetPixelMode(&nMode) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.9 IGAASetInterfaceSpeed

bool IGAASetInterfaceSpeed(int nSpeed)

[Summary]

Sets the Interface Speed.
When High-Speed is selected the image sensor interface uses staggered Odd/Even clocks with pixel output every 4 clocks.[1]
When Normal-Speed is selected the image sensor interface uses non-staggered Odd/Even clocks.[1]
[1] Note: Odd/Even clock activity is as applicable per specified pixel mode – 256 or 512 pixels, refer to Section 6.1.7

[Arguments]

nSpeed sets the interface speed as per "IGAAUSB.h":
IGAA_ISPEED_HIGH sets the interface to High-Speed.
IGAA_ISPEED_NORMAL sets the interface to Normal-Speed.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORD dwErrCode;
if(IGAASetInterfaceSpeed(IGAA_ISPEED_HIGH) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.10 IGAAGetInterfaceSpeed

bool IGAAGetInterfaceSpeed(int* pSpeed);

[Summary]

Retrieves the Interface Speed.
Refer to section 6.1.9 for the functional description of High-Speed vs. Normal-Speed interface.

[Arguments]

pSpeed Specifies the address of the variable where the interface speed that is currently set is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nSpeed;
DWORD dwErrCode;
if(IGAAGetInterfaceSpeed(&nSpeed) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.11 IGAASetInterfaceStyle

bool IGAASetInterfaceStyle(int nStyle)

[Summary]

Sets the Interface Style.
The styles available are Sequential and Parallel.
When Sequential Interface Style is selected the Odd and Even clocks are activated sequentially in bursts of 8 – Odd/Even/Odd/Even.[1]
When Parallel Interface Style is selected the Odd and Even clocks are activated concurrently.[1]

[1] Note: Odd/Even clock activity is as applicable per specified pixel mode – 256 or 512 pixels, refer to Section 6.1.7

[Arguments]

nStyle specifies the interface style as per "IGAAUSB.h".
IGAA_ISTYLE_SEQUENTIAL sets the interface style to Sequential.
IGAA_ISTYLE_PARALLEL sets the interface style to Parallel.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORD dwErrCode;
if(IGAASetInterfaceStyle(IGAA_ISTYLE_SEQUENTIAL) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.12 IGAAGetInterfaceStyle

bool IGAAGetInterfaceStyle(int* pStyle)

[Summary]

Retrieves the Interface Style.
Refer to section 6.1.11 for the functional description of Sequential vs. Parallel Interface Style.

[Arguments]

pStyle Specifies the address of the variable where the interface style that is currently set is to be stored.
The values are as per "IGAAUSB.h".

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nStyle;
DWORD dwErrCode;
if(IGAAGetInterfaceStyle(&nStyle) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.13 IGAASetMuxControl

bool IGAASetMuxControl(int nPolarity)

[Summary]

Sets the Mux Control Polarity.
The multiplexor being controlled is U19 on G920x InGaAs Array Reference Design.
The control represents the polarity of the EVEN_ODD control signal when the corresponding even or odd data is being clocked out of the image sensor.

[Arguments]

nInvert To specify the Mux Control Polarity, that are specified in "IGAAUSB.h".
IGAA_MUXC_NORMAL is used to associate odd data with odd clocks, even data with even clocks
IGAA_MUXC_INVERTED is used for test only to associate odd data with even clocks and even data with odd clocks.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORD dwErrCode;
if(IGAASetMuxControl(IGAA_MUXC_NORMAL) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.14 IGAAGetMuxControl

bool IGAAGetMuxControl(int* pPolarity)

[Summary]

Retrieves the Mux Control Polarity.
Refer to Section 6.1.13 for functional description of the Mux Control Polarity control

[Arguments]

pPolarity specifies the address of the variable where the Mux Control Polarity that is currently set is to be stored.
Mux Control Polarity is specified as per "IGAAUSB.h".

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nPolarity;
DWORD dwErrCode;
if(IGAAGetMuxControl(&nPolarity) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.15 IGAAGetCaptureBytes

bool IGAAGetCaptureBytes(int* pBytes)

[Summary]

Retrieves the number of bytes of one captured image

[Arguments]

pBytes Specifies the address of the variable where the number of bytes of one captured image is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nBytes = 0;
DWORD dwErrCode;
// Get size of captured image in bytes.
if(IGAAGetFrameBytes(&nBytes) != TRUE){
dwErrCode = IGAAGetLastError();
```

6.1.16 IGAAGetTotalCaptureBytes

```
bool IGAAGetTotalCaptureBytes(int* pBytes)
```

[Summary]

Retrieves the total number of bytes received in capturing images.

[Arguments]

pBytes Specifies the address of the variable where the total number of bytes received in capturing images is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nBytes = 0;
DWORD dwErrCode;
// Get total bytes of captured images.
if(IGAAGetFrameBytes(&nBytes) != TRUE){
dwErrCode = IGAAGetLastError();
```

6.1.17 IGAACapture

bool IGAACapture(LPVOID pImageBuff, int nBuffSize)

[Summary]

Return one image received in capturing images from the device, if capturing started.

[Arguments]

pImageBuff Specifies the starting address of the buffer where the image data is to be stored.
nBuffSize Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

1) This function issues an instruction to start capturing the image. Since the image capturing is not complete even when this function ends, use the IGAAWait function to check whether image capturing is complete.
2) The necessary buffer size can be obtained with the IGAAGetCaptureBytes or IGAAGetImageSize functions.

[Reference]

None

[Example]

```
res = IGAACapture(&m_InputData[0], numBytes);
if(!res)
        break;
        if(numBytes != 512){
        m_data_WR = numBytes;
}
```

6.1.18 IGAACaptureBuffer

bool IGAACaptureBuffer(LPVOID pImageBuff, int nBuffSize)

[Summary]

Starts to capture images from the device.

[Arguments]

None

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
res = IGAACaptureBuffer(&m_InputData[m_data_WR], readBytes);
if(res) {
        m_data_WR += readBytes/2;
        saveDataToFile();
}
```

6.1.19 IGAAStart

bool IGAAStop (void)

[Summary]

Starts to capture images from the device.
Note: this function sets FPGA registers as follows: FPGA_Reg_1 |= 0xC0;

[Arguments]

None

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

bool res = IGAAStart();

6.1.20 IGAAStop

bool IGAAStop (void)

[Summary]

Stops capturing the image.
Note: this function sets the FPGA registers as follows: FPGA_Reg_1 &= 0x3F

[Arguments]

None

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
bool res = IGAAStop();
```

6.1.21 IGAAWait

bool IGAAWait(DWORD* pStatus, int nTimeout)

[Summary]

Waits till the image is captured.

[Arguments]

pStatus Specifies the address of the variable where the image capturing completion status is to be stored.
Whether image capturing is complete or not can be checked by the value in this variable.
The value is either of the following: nTimeout Specifies the length of timeout in milliseconds.

When "IGAA_WAIT_INFINITE" is specified here, the process waits until image capturing is finished.
When "0" is specified, control is returned immediately after checking the status.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

None

6.1.22 IGAASetPixelClockDivider

bool IGAASetPixelClockDivider(int nValue)

[Summary]

Sets the Pixel clock divider value.

[Arguments]

nValue Specifies the Pixel clock divider value.
Valid range is 1 to 511. Refer to Table 1 Clock_Divider column.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORD dwErrCode;
if(IGAASetPixelClockDivider(160) != TRUE){
dwErrCode = IGAAGetLastError();
```

6.1.23 IGAAGetPixelClockDivider

bool IGAAGetPixelClockDivider(int* pValue)

[Summary]

Retrieves the Pixel clock divider value.
Valid range is 1 to 511. Refer to Table 1 Clock_Divider column.

[Arguments]

pValue Retrieves the Pixel clock divider value

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nValue;
DWORD dwErrCode;
if(IGAAGetPixelClockDivider(&nValue) != TRUE){
dwErrCode = IGAAGetLastError();
```

6.1.24 IGAASetExposureTime

bool IGAASetExposureTime(unsigned int nTime)

[Summary]

Sets the exposure time (duration of the reset interval).

[Arguments]

nTime Specifies the exposure time.
The resultant integration (reset) time duration is a product of the nTime parameter of the present function and the odd/even clock period (the time unit) resultant from IGAASetPixelClockDivider function call.
Refer to Table 1 for odd/even clock rate information.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
if(IGAASetExposureTime(120) != TRUE){
dwErrCode = IGAAGetLastError();
```

6.1.25 IGAAGetExposureTime

bool IGAAGetExposureTime(unsigned int* pTime)

[Summary]

Retrieves the exposure time.

[Arguments]

pTime Retrieves the exposure time that is currently set in standard time units.
Refer to IGAASetExposureTime function for the explanation of the time units.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nTime;
DWORD dwErrCode;
if(IGAAGetExposureTime(&nTime) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.26 IGAASetDigitalPotentiometer1

bool IGAASetDigitalPotentiometer1(int nValue)

[Summary]

Sets the Digital Potentiometer 1 value.
The digital potentiometer being controlled is U4-A on G920x InGaAs Array Reference Design.

[Arguments]

nValue Specifies the Digital Potentiometer 1 value.
The valid range of nValue is 0 to 1023.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORD dwErrCode;
if(IGAASetDigitalPotentiometer1(500) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.27 IGAAGetDigitalPotentiometer1

bool IGAAGetDigitalPotentiometer1(int* pValue)

[Summary]

Retrieves the Digital Potentiometer 1 value.
The digital potentiometer is U4-A on G920x InGaAs Array Reference Design.

[Arguments]

pValue Retrieves the Digital Potentiometer 1 value that is currently set

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nValue;
DWORD dwErrCode;
if(IGAAGetDigitalPotentiometer1(&nValue) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.28 IGAASetDigitalPotentiometer2

bool IGAASetDigitalPotentiometer2(int nValue)

[Summary]

Sets the Digital Potentiometer 2 value.
The digital potentiometer being controlled is U4-B on G920x InGaAs Array Reference Design.

[Arguments]

nValue Specifies the Digital Potentiometer 2 value.
The valid range of nValue is 0 to 1023.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORD dwErrCode;
if(IGAASetDigitalPotentiometer2(500) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.29 IGAAGetDigitalPotentiometer2

bool IGAAGetDigitalPotentiometer2(int* pValue)

[Summary]

Retrieves the Digital Potentiometer 2 value.
The digital potentiometer is U4-B on G920x InGaAs Array Reference Design.

[Arguments]

pValue Retrieves the Digital Potentiometer 2 value that is currently set

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nValue;
DWORD dwErrCode;
if(IGAAGetDigitalPotentiometer2(&nValue) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.30 IGAAGetAtoDChannel0

bool IGAAGetAtoDChannel0(int* pValue)

[Summary]

Retrieves the Analog to Digital channel 0 converter value.
The ADC being controlled is U10 on G920x InGaAs Array Reference design.

[Arguments]

pValue Retrieves the Analog to Digital channel 0 converter value

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nValue;
DWORD dwErrCode;
if(IGAAGetAtoDChannel0(&nValue) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.31 IGAAGetAtoDChannel1

bool IGAAGetAtoDChannel1(int* pValue)

[Summary]

Retrieves the Analog to Digital channel 1 converter value.
The ADC being controlled is U10 on G920x InGaAs Array Reference design.

[Arguments]

pValue Retrieves the Analog to Digital channel 1 converter value

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nValue;
DWORD dwErrCode;
if(IGAAGetAtoDChannel1(&nValue) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.32 IGAAGetAtoDChannel2

bool IGAAGetAtoDChannel2(int* pValue)

[Summary]

Retrieves the Analog to Digital channel 2 converter value.
The ADC being controlled is U10 on G920x InGaAs Array Reference design.

[Arguments]

pValue Retrieves the Analog to Digital channel 2 converter value

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nValue;
DWORD dwErrCode;
if(IGAAGetAtoDChannel2(&nValue) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.33 IGAASetDtoAConverter

bool IGAASetDtoAConverter(int nValue)

[Summary]

Sets the Digital to Analog converter value.
The DAC being controlled is U9 on G920x InGaAs Array Reference design.

[Arguments]

nValue Specifies the Digital to Analog converter value.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORD dwErrCode;
if(IGAASetDtoAConverter(240) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.34 IGAAGetDtoAConverter

bool IGAAGetDtoAConverter(int* pValue)

[Summary]

Retrieves the Digital to Analog converter value.
The DAC being controlled is U9 on G920x InGaAs Array Reference design.

[Arguments]

pValue Retrieves the Digital to Analog converter value

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nValue;
DWORD dwErrCode;
if(IGAAGetDtoAConverter(&nValue) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.35 IGAASetPatternGenerator

bool IGAASetPatternGenerator(int nMode)

[Summary]

Sets the Pattern Generator Mode.

[Arguments]

nMode specifies the Pattern Generator mode as per "IGAAUSB.h":
IGAA_PGM_OFF: Pattern generator is OFF.
IGAA_PGM_ON: Pattern generator is ON.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORD dwErrCode;
if(IGAASetPatternGenerator(IGAA_PGM_ON) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.36 IGAAGetPatternGenerator

bool IGAAGetPatternGenerator(int* pMode)

[Summary]

Retrieves the Pattern Generator Mode.

[Arguments]

pMode Specifies the address of the variable where the Pattern Generator mode that is currently set is to be stored.
Acquire Pattern Generator mode that are retrieved, that is specified in the "IGAAUSB.h".

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nMode;
DWORD dwErrCode;
if(IGAAGetPatternGenerator(&nMode) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.37 IGAAGetVersion

bool IGAAGetVersion(char* szVersion, int nBufSize)

[Summary]

Retrieves the library version number, in string format.

[Arguments]

szVersion Specifies the starting address of the character string buffer where the version of the library is to be stored.
nBufSize Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
char szVersion[256];
DWORD dwErrCode;
if(IGAAGetVersion(szVersion, sizeof(szVersin)) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.38 IGAAGetDriverVersion

bool IGAAGetDriverVersion(char* szVersion, int nBufSize)

[Summary]

Retrieves the driver version number, in string format.

[Arguments]

szVersion Specifies the starting address of the character string buffer where the version of the driver is to be stored.
nBufSize Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
char szVersion[256];
DWORD dwErrCode;
if(IGAAGetDriverVersion(szVersion, sizeof(szVersin)) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.39 IGAAGetFirmwareVersion

bool IGAAGetFirmwareVersion(char* szFirmVersion, int nBufSize)

[Summary]

Retrieves the firmware version number, in a character string format.

[Arguments]

szFirmVersion Specifies the starting address of the character string buffer where the version of the firmware is to be stored.
nBufSize Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
char szVersion[256];
DWORD dwErrCode;
if(IGAAGetFirmwareVersion(szVersion, sizeof(szVersin)) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.40 IGAAGetLastError

DWORD IGAAGetLastError(void)

[Summary]

Retrieves the last-error code.

[Arguments]

None

[Return Value]

The last error code is returned. For details on error code, refer to the error code table in Section 5.2.1.

[Note]

None

[Reference]

None

[Example]

None

6.1.41 IGAAGetDebugInformation

bool IGAAGetDebugInformation(char* pszBuff, int nBufSize)

[Summary]

Retrieves the Debug Information string.

[Arguments]

pszBuff Specifies the starting address of the character string buffer where the Debug Information is to be stored.
nBufSize Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).

[Note]

None

[Reference]

None

[Example]

```
r = IGAAGetDebugInformation(pCMainFrame->m_debugString, DEBUG_STRING_LEGTH);
```

6.1.42 IGAAGetDebugStream

bool IGAAGetDebugStream(char* pszBuff, int nBufSize)

[Summary]

Retrieves the Debug Numbers.

[Arguments]

pszBuff Specifies the starting address of the integer values buffer where the Debug Numbers is to be stored.
nBufSize Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).

[Note]

None

[Reference]

None

[Example]

```
r = IGAAGetDebugStream((char*) &nPackets, sizeof(int));
if(r){
        if(nPackets){
                pCMainFrame->m_dataLen = nPackets;
        }
}
```

6.1.43 IGAAGetRegisters

bool IGAAGetRegisters(char* pszBuff, int nBufSize, int nAddress)

[Summary]

Retrieves the Device Registers.

[Arguments]

pszBuff Specifies the starting address of the buffer where the registers data is to be stored.
nBuffSize Specifies the buffer size and number of retrieving bytes.
nAddress Specifies the starting registers address

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
bool res = IGAAGetRegisters(tData, 1, nAddress);
if(res)
        return tData[0];
return 0xFF;
```

6.1.44 IGAASetRegisters

bool IGAASetRegisters(char* pszBuff, int nBufSize, int nAddress)

[Summary]

Sets the Device Registers.

[Arguments]

pszBuff Specifies the starting address of the buffer where the registers data is stored.
nBuffSize Specifies the buffer size and number of setting bytes.
nAddress Specifies the starting registers address

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
unsigned char tData[256];
tData[0] = nData;
bool res = IGAASetRegisters((char*)tData, 1, nAddress);
```

6.1.45 IGAASetGain

bool IGAASetGain(int nGain)

[Summary]

Sets the Sensor Gain

[Arguments]

nGain specifies the sensor gain as per "IGAAUSB.h":
IGAA_GAIN_LOW: the sensor gain is Low (CF_SELECT = 0)
IGAA_GAIN_HIGH: the sensor gain is High (CF_SELECT = 1)

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
DWORD dwErrCode;
if(IGAASetGain(IGAA_GAIN_HIGH) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.46 IGAAGetGain

bool IGAAGetGain(int* pGain)

[Summary]

Retrieves the Sensor Gain setting

[Arguments]

pGain specifies the address of the variable where the sensor gain that is currently set is to be stored.
The values are as per "IGAAUSB.h":
IGAA_GAIN_LOW: the sensor gain is Low (CF_SELECT = 0)
IGAA_GAIN_HIGH: the sensor gain is High (CF_SELECT = 1)

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
INT nGain;
DWORD dwErrCode;
if(IGAAGetGain (&nGain) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

6.1.47 IGAACaptureToFile

bool IGAACaptureToFile (int nImageNumber, CallbackProgressFunc Callback)

[Summary]

Captures nImageNumber of lines from the device and writes them into a file in a comma-separated value (.csv) format. The resultant file can be open directly using Microsoft Excel.

[Arguments]

nImageNumber specifies the number of lines (frames) to be captured, max is 1000.
CallBack specifies the address of the callback function used to show the capture progress status.

[Return Value]

If the function is successful, the return value is TRUE (1).
Otherwise, the return value is FALSE (0).
For details on error information, refer to the IGAAGetLastError function.

[Note]

None

[Reference]

None

[Example]

```
void __stdcall MyProgressFunc(int progress) DWORD dwErrCode;
{

}


DWORD dwErrCode;
if(IGAACaptureToFile(50, &MyProgressCallback) != TRUE){
dwErrCode = IGAAGetLastError();
}
```

7 Supplementary Information

Table 1 - Clock Divider Settings

Clock_Divider[8:0]		Odd/Even Clock Rate (KHz)	Actual Pixel Rate (KHz)
Decimal	*Binary*		
511	111111111	58.708	7.339
500	111110100	60.000	7.500
200	011001000	150.000	18.750
100	001100100	300.000	37.500
67	001000011	447.761	55.970
50	000110010	600.000	75.000
40	000101000	750.000	93.750
37	000100101	789.474	98.684
38	000100110	810.811	101.351
33	000100001	909.091	113.636
29	000011101	1034.483	129.310
25	000011001	1200.000	150.000
22	000010110	1363.636	170.455
20	000010100	1500.000	187.500
18	000010010	1666.667	208.333
17	000010001	1764.706	220.588
15	000001111	2000.000	250.000
14	000001110	2142.857	267.857
13	000001101	2307.692	288.462
12	000001100	2500.000	312.500
11	000001011	2727.273	340.909
10	000001010	3000.000	375.000
8	000001000	3750.000	468.750
7	000000111	4285.714	535.714
1	000000001	30000.000	3750.000

Refer to "Hamamatsu G920x Reference Design FPGA to USB Processor Interface Specification" for detailed FPGA register description.