

USB2 MINI-SPECTROMETERS
C13016
DLL Function

specifications

Doc Version 1.2

Release Note

Feb.2015 First Edition

Initial release

Oct.2015 A Edition

Deleted MICRO_USB2_getExposureCycle

Deleted MICRO_USB2_setExposureCycle

Changed MICRO_USB2_getModuleProperty

Changed MICRO_USB2_captureStart

Changed MICRO_USB2_getImageData

Changed Environment

Changed Specification title

Nov.2015 B Edition

Changed Environment

List

<i>Release Note</i>	2
1. <i>Overview</i>	4
2. <i>MICRO_USB2_initialize</i>	5
3. <i>MICRO_USB2_getModuleConnectionList</i>	6
4. <i>MICRO_USB2_open</i>	7
5. <i>MICRO_USB2_close</i>	8
6. <i>MICRO_USB2_uninitialize</i>	9
7. <i>MICRO_USB2_getModuleInformation</i>	10
8. <i>MICRO_USB2_getSpectroInformation</i>	11
9. <i>MICRO_USB2_getModuleProperty</i>	12
10. <i>MICRO_USB2_getExposureProperty</i>	13
11. <i>MICRO_USB2_getCaptureMode</i>	14
12. <i>MICRO_USB2_setCaptureMode</i>	15
13. <i>MICRO_USB2_getExposureTime</i>	16
14. <i>MICRO_USB2_setExposureTime</i>	17
15. <i>MICRO_USB2_getTriggerPolarity</i>	18
16. <i>MICRO_USB2_setTriggerPolarity</i>	19
17. <i>MICRO_USB2_getTriggerOutput</i>	20
18. <i>MICRO_USB2_setTriggerOutput</i>	21
19. <i>MICRO_USB2_getCalibrationCoefficient</i>	22
20. <i>MICRO_USB2_setCalibrationCoefficient</i>	23
21. <i>MICRO_USB2_captureStart</i>	24
22. <i>MICRO_USB2_getCaptureStatus</i>	25
23. <i>MICRO_USB2_getImageData</i>	26
24. <i>MICRO_USB2_captureStop</i>	27
<i>Return Code</i>	28

1. Overview

[Environment]

Microsoft(R) Windows 7 32bit, 64bit

Microsoft(R) Windows 8 32bit, 64bit

Microsoft(R) Windows 10 32bit, 64bit

[Development environment]

Microsoft Visual Studio 2008 Professional Edition + SP1

[Driver]

Use WinUSB.SYS

[DLL]

MICRO_USB2_CLR.dll

Mini-spectrometers are designed to connect to a PC via the USB 2.0 interface.

Maximum 8 unit of mini spectrometer can be connected to 1PC for synchronous working

2. MICRO_USB2_initialize

[Function]

Initialization of library.

[Format]

```
long MICRO_USB2_initialize( void ) ;
```

[Arguments]

Nothing

[Return]

usb2Success	...Successful initialization of the library.
usb2Err_unsuccess	...Initialization of library is failed.
usb2Err_initialized	...Already initialized.

[Comment]

The library is initialized. After initialization, if MICRO_USB2_initialize is called again, then it will return back. If you want to reinitialize, please reinitialize by using MICRO_USB2_unitalize function after opening a library.

[References]

MICRO_USB2_uninitialize

3. **MICRO_USB2_getModuleConnectionList**

[Function]

Get the list of connected modules.

[Format]

```
long MICRO_USB2_getModuleConnectionList (
    array<short>^ list ,
    unsigned short% number
);
```

[Arguments]

list ...handle to store the list of connected modules.
number ...Stores the number of connected modules.

[Return]

usb2Success	...Successful in acquiring the list.
usb2Err_notList	...Failed in acquiring the list.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.

[Comment]

Get the list of connected modules. The list is a Device index of all connected modules. Please allocate enough space to the list to acquire the list. Detected module acquires Device index greater than 0. If the module is not detected then acquires -1 value. number is number of detected modules. For example if 5 devices is detected then 5 is acquired.

[References]

4. MICRO_USB2_open

[Function]

Opens device.

[Format]

```
long MICRO_USB2_open (  
    long id  
);
```

[Arguments]

id ...Device index.

[Return]

usb2Success	...Successful in opening the device.
usb2unsuccess	...Failed in opening the device.
usb2Err_invalidDevice index	...Invalid specified Device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_opened	...Already opened.

[Comment]

The measuring instrument device is initialized. Please specify the Device index. The Device index can be acquired from MICRO_USB2_getModuleConnectionList function. After using application, please release initialized measurement device by using MICRO_USB2_close function. If the USB device is removed from PC without releasing, Windows system may get affected.

[References]

MICRO_USB2_close

5. MICRO_USB2_close

[Function]

Closes device.

[Format]

```
long MICRO_USB2_Close (  
    long id  
);
```

[Arguments]

id ...Device index.

[Return]

usb2Success	...Successful in closing the device.
usb2Err_invalidDevice index	...Invalid specified Device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.

[Comment]

Measurement device in use is released. Please use it when the application ends. If the USB device is removed from PC without releasing, Windows system may get affected.

[References]

MICRO_USB2_open

6. MICRO_USB2_uninitialize

[Function]

Exit device driver and library.

[Format]

```
long MICRO_USB2_uninitialize (  
    void  
);
```

[Arguments]

Nothing

[Return]

usb2Success	... Successful in exiting library
usb2Err_existDevice	... Exist the device which is not closed.
usb2Err_nowCapturing	... Capturing
usb2Err_nowFirmUpdate	... Updating firmware

[Comment]

Device driver and library is released. If the device is not closed, usb2Err_existDevice is returned. Please release the library after all devices are closed by MICRO_USB2_close function.

[References]

MICRO_USB2_initialize

7. MICRO_USB2_getModuleInformation

[Function]

Product information of module (Camera) is acquired.

[Format]

```
long MICRO_USB2_getModuleInformation (
    long id,
    CModuleInformation^ information
);
```

[Arguments]

id ... Specified device index.
information ...Handle to store product information.

```
public ref class CModuleInformation {
    String^ vender;            //Company name
    String^ product;         //Model No
    String^ serial;          //Serial No
    String^ firmware;        //Firmware Ver
    String^ driver;          //Driver Ver
    String^ library;         //DLLVer
}
```

[Return]

usb2Success ... Successful in acquiring data.
usb2Err_invalidDevice index ... Invalid specified Device index.
usb2Err_notAccess ... Failed to access specified device.
usb2Err_nowCapturing ... Capturing.
usb2Err_nowFirmUpdate ... Updating firmware.
usb2Err_firmCorrupted ... Firm is not properly written.

[Comment]

Product information of module is acquired. Acquired information is company name, model number, serial No, Firmware Ver, Driver Ver, DLLVer.

[References]

8. MICRO_USB2_getSpectroInformation

[Function]

Product information of module (Spectrometer) is acquired.(Only for spectrometer)

[Format]

```
long MICRO_USB2_getSpectroInformation (
    long id,
    CSpectroInformation^ information
);
```

[Arguments]

id ... Device index.
information ...Handle to store product information.

```
public ref class CSpectroInformation{
    String^ unit;     //Unit No
    String^ sensor;  //Sensor name
};
```

[Return]

usb2Success	... Successful in acquiring data.
usb2Err_invalidDevice index	... Invalid specified Device index.
usb2Err_notAccess	... Failed to access specified device.
usb2Err_nowCapturing	... Capturing.
usb2Err_nowFirmUpdate	... Updating firmware.
usb2Err_firmCorrupted	... Firm is not properly written.

[Comment]

Product information of module is acquired. Acquired information is unit no, sensor name.

[References]

9. MICRO_USB2_getModuleProperty

[Function]

Get functionality of the module.

[Format]

```
long MICRO_USB2_getModuleProperty (
    long id,
    CModuleProperty^ ModuleProperty
);
```

[Arguments]

id ... Device index.

ModuleProperty ...Handle to save functionality information.

```
public ref class CModuleProperty {
    long pixel;           //Sensor pixel
    long resolution;      //A/D resolution
    long capturemode;     //Available capture mode
};
```

Corresponding bit of capturemode,binningmode,triggermode,triggerpolarity will be 1.

- Capture mode

Bit	Mode
1, 3-31	Reserved bit
2	Trigger measurement mode
0	Specified no of times measurement mode

[Return]

```
usb2Success           ... Successful in acquiring data.
usb2Err_invalidDevice index ... Invalid specified Device index.
usb2Err_notAccess     ... Failed to access specified device.
usb2Err_nowCapturing ... Capturing.
usb2Err_nowFirmUpdate  ... Updating firmware.
usb2Err_firmCorrupted  ... Firm is not properly written.
```

[Comment]

Functionality information is acquired. Acquired information is sensor pixel, A/D resolution, Available capture mode.

[References]

MICRO_USB2_getExposureProperty,MICRO_MICRO_USB2_getAdOffsetProperty,

10. *MICRO_USB2_getExposureProperty*

[Function]

Usable functional information related to exposure is acquired.

[Format]

```
long MICRO_USB2_getExposureProperty (
    long id ,
    CExposureProperty^ ExposureProperty
);
```

[Arguments]

```
id                ... Device index.
ExposureProperty ...Handle to store functional information.
public ref class CExposureProperty {
    double lowertime; // Usable minimum exposure time
    double upptime;   // Usable maximum exposure time
    double timestep;  //Change interval of usable exposure time
    double lowercycle; //Available minimum exposure period
    double uppercycle; // Available maximum exposure period
    double cyclestep; // Change interval of usable exposure period
};
```

[Return]

usb2Success	...Successful in acquiring data.
usb2Err_invalidDevice index	...Invalid specified Device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Usable functional information related to exposure is acquired. Acquired information is usable exposure time range, change interval of usable exposure time, Available exposure period, Change interval of usable exposure period. Unit is usec.

[Reference]

```
MICRO_USB2_getModuleProperty
MICRO_USB2_getAdOffsetProperty
```

11. *MICRO_USB2_getCaptureMode*

[Function]

Get capture mode .

[Format]

```
long MICRO_USB2_getCaptureMode (
    long id,
    long% mode
);
```

[Arguments]

id ...Device index.
mode ...Stores capture mode.

[Return]

usb2Success	...Successful in acquiring capture mode.
usb2Err_invalidDevice index	...Invalid specified Device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Capture mode is acquired. Possible value is one of the following values.
0x00000000 ... Specified number of measurement mode
0x00000002 ... Trigger mode

[Reference]

MICRO_USB2_setCaptureMode

12. MICRO_USB2_setCaptureMode

[Function]

Sets capture mode.

[Format]

```
long MICRO_USB2_setCaptureMode (
    long id,
    long mode
);
```

[Arguments]

id ...Device index.

mode ...Capture mode.

[Return]

usb2Success	...Successful in setting capture mode.
usb2Err_invalidDevice index	...Invalid specified Device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_invalidMode	...Invalid capture mode.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Capture mode is set. One of the following values can be set. If the value other than following value is set, this value is not reflected.

0x00000000 ...Specified number of measurement mode

0x00000002 ...Trigger mode

[Reference]

MICRO_USB2_getCaptureMode

13. MICRO_USB2_getExposureTime

[Function]

Get exposure time.

[Format]

```
long MICRO_USB2_getExposureTime (  
    long id ,  
    double% sec  
);
```

[Arguments]

id ...Device index.
sec ...Stores exposure time.

[Return]

usb2Success	...Successful in getting exposure time.
usb2Err_invalidDevice index	...Invalid specified Device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Currently set exposure time is acquired. Unit is usec.

[Reference]

MICRO_USB2_setExposureTime

14. *MICRO_USB2_setExposureTime*

[Function]

Set exposure time.

[Format]

```
long MICRO_USB2_setExposureTime (
    long id,
    double sec
);
```

[Arguments]

id ...Device index.
sec ...Exposure time.

[Return]

usb2Success	...Exposure time is successfully acquired.
usb2Err_invalidDevice index	...Invalid specified Device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_overTime	...The exposure time exceeds the specified upper limit.
usb2Err_underTime	...The exposure time exceeds the specified lower limit.
usb2Err_invalidParameter	...Value greater than the exposure period.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Exposure time is set. Unit is usec.

[Reference]

MICRO_USB2_getExposureTime

15. *MICRO_USB2_getTriggerPolarity*

[Function]

Get trigger polarity.

[Format]

```
long MICRO_USB2_getTriggerPolarity (
    long id ,
    long% polarity
);
```

[Arguments]

id ...Device index.
polarity ...Stores trigger polarity

[Return]

usb2Success	...Trigger polarity is successfully acquired.
usb2Err_invalidDevice index	...Invalid specified Device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Currently set trigger polarity is acquired. Value to be stored will be one of the following values.
The default value is set to 0x00000001. Moreover, the trigger function cannot be used, though default value 0x00000001 is set for the device that doesn't maintain trigger function.

0x00000001 ...Falling edge

0x00000011 ...Rising edge

[Reference]

MICRO_USB2_setTriggerPolarity

16. *MICRO_USB2_setTriggerPolarity*

[Function]

Set trigger polarity.

[Format]

```
long MICRO_USB2_setTriggerPolarity (
    long id,
    long polarity
);
```

[Arguments]

id ...Device index.
polarity ...trigger polarity value

[Return]

usb2Success	...Trigger polarity is successfully acquired.
usb2Err_invalidDevice index	...Invalid specified Device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_invalidTriggerpolarity	...Invalid specified trigger polarity value.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Currently set trigger polarity is set. Value to be stored will be one of the following values. If the value other than following value is set, this value is not reflected.

Moreover, the trigger function cannot be used, though it is possible to do settings to the device that doesn't maintain trigger function.

0x00000001 ... Falling edge

0x00000011 ... Rising edge

[Reference]

MICRO_USB2_getTriggerPolarity

17. **MICRO_USB2_getTriggerOutput**

[Function]

Get the output operating status of trigger.

[Format]

```
long MICRO_USB2_getTriggerOutput (
    long id ,
    long% output
);
```

[Arguments]

id ...Device index.

output ...Stores the output operating status of trigger.

[Return]

usb2Success	...Trigger output operating status is successfully acquired.
usb2Err_invalidDeviceIndex	...Invalid specified device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Trigger output operating status is acquired. The value obtained is one of the following values. The default value is set to 0x00000001. Moreover, the trigger function cannot be used, though default value 0x00000001 is set for the device that doesn't maintain trigger function.

0x00000001 ...OFF

0x00000011 ...ON

[References]

MICRO_USB2_setTriggerOutput

18. *MICRO_USB2_setTriggerOutput*

[Function]

Set the output operating status of trigger.

[Format]

```
long MICRO_USB2_setTriggerOutput (
    long id,
    long output
);
```

[Arguments]

id	...Device index
output	...output operating status of trigger value

[Return]

usb2Success	...Trigger output operating status is successfully set.
usb2Err_invalidDeviceIndex	...Invalid specified device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_invalidTriggeroutput	...Invalid specified trigger output operating status.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Trigger output operating status is set. The value obtained is one of the following values.

The default value is set to 0x00000001. Moreover, the trigger function cannot be used, though default value 0x00000001 is set for the device that doesn't maintain trigger function.

0x00000001 ... OFF

0x00000011 ... ON

[References]

MICRO_USB2_setTriggerOutput

19. MICRO_USB2_getCalibrationCoefficient

[Function]

Get calibration coefficient.(For spectrometer)

[Format]

```
long MICRO_USB2_getCalibrationCoefficient (
    long id ,
    long index ,
    array<double>^ coefficient
);
```

[Arguments]

id ...Device index
 index ...Index specified out of 2 types of calibration coefficient.
 0x00000000 ...Wavelength calibration coefficient
 0x00000001 ...Sensitivity calibration coefficient
 coefficient ...Set calibration coefficient value

[Return]

usb2Success	...Calibration coefficient is successfully set.
usb2Err_invalidDeviceIndex	...Invalid specified device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_invalidIndex	...Invalid specified index of calibration coefficient.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Currently set calibration coefficient is acquired. Calibration coefficient is a double value as shown in example given below.

Example : $\pm 1.123456789E \pm 05$

There are 6 types of calibration coefficient. Array order is A0, B1, B2, B3, B4, and B5.

The calibration type using the wavelength calibration coefficient is shown below.

$$\lambda \text{ [nm]} = A0 + B1\text{pix} + B2\text{pix}^2 + B3\text{pix}^3 + B4\text{pix}^4 + B5\text{pix}^5$$

pix is arbitrary pixel number.

Calibration coefficient for sensitivity is not used now, but it may be useful in future.

[References]

20. MICRO_USB2_setCalibrationCoefficient

[Function]

Set calibration coefficient.(For spectrometer)

[Format]

```
long MICRO_USB2_setCalibrationCoefficient (
    long id ,
    long index ,
    array<double>^ coefficient
);
```

[Arguments]

id ...Device index
 index ...Index specified out of 2 types of calibration coefficient.
 0x00000000 ...Wavelength calibration coefficient
 0x00000001 ...Sensitivity calibration coefficient
 coefficient ...Set calibration coefficient value

[Return]

usb2Success	...Calibration coefficient is successfully set.
usb2Err_invalidDeviceIndex	...Invalid specified device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_invalidIndex	...Invalid specified index of calibration coefficient.
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Currently set calibration coefficient is acquired. Calibration coefficient is a double value as shown in example given below.

Example : $\pm 1.123456789E \pm 05$

There are 6 types of calibration coefficient. Array order is A0, B1, B2, B3, B4, and B5.

The calibration type using the wavelength calibration coefficient is shown below.

$$\lambda \text{ [nm]} = A0 + B1\text{pix} + B2\text{pix}^2 + B3\text{pix}^3 + B4\text{pix}^4 + B5\text{pix}^5$$

pix is arbitrary pixel number.

Calibration coefficient for sensitivity is not used now, but it may be useful in future.

[References]

21. *MICRO_USB2_captureStart*

[Function]

Capture operation is started.

[Format]

long MICRO_USB2_captureStart (long id);

[Arguments]

id ...Device index

[Return]

usb2Success	...Capture operation is started successfully.
usb2Err_invalidDeviceIndex	...Invalid specified device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_invalidIndex	...Invalid specified capture index.
usb2Err_notFrame	...Frame is not allocated.
usb2Err_notAcPower	...The unit is not connected to AC adapter (for models which requires an external power supply)
usb2Err_nowCapturing	...Capturing.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Capture operation is started.

[References]

MICRO_USB2_captureStop

22. *MICRO_USB2_getCaptureStatus*

[Function]

Capture status is acquired.

[Format]

```
long MICRO_USB2_getCaptureStatus ( long id );
```

[Arguments]

id ...Device index

[Return]

usb2Success	...Captured frames exist.
usb2Err_unsuccess	...Captured frames does not exist.
usb2Err_notFrame	...Loss of data generated.
usb2Err_invalidDeviceIndex	...Invalid specified device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Current capture status is acquired. Status can be obtained only when there is a captured frame. If there is no frame has been captured usb2Err_unsuccess returns. If you miss a piece of data generated during the measurement, usb2Err_notFrame is returned.

In this case, data is not guaranteed after this, so please stop capturing and restart the measurements.

[References]

23. *MICRO_USB2_getImageData*

[Function]

Latest capture data is acquired from buffer.(Acquire data only)

[Format]

```
long MICRO_USB2_getImageData (
    long id ,
    array<unsigned short>^% imageData ,
    array<unsigned long>^% time
);
```

[Arguments]

id ...Device index.
 imageData ...pointer to store data
 time ...pointer to store time from the measurement beginning to data acquisition

[Return]

usb2Success	...Data is acquired successfully.
usb2Err_invalidData	...Data is retrieved successfully but contains invalid data.
usb2Err_notImage	...Failed to acquire data
usb2Err_invalidDeviceIndex	...Invalid specified device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Latest capture data is acquired. Only data is retrieved and header information is discarded.

While obtaining it, check the 0th address (acquisition state) of header information, if the error value is (1-3), then usb2Err_invalidData is returned.

Also, if there is no data in the buffer, usb2Err_notImage is returned.

[References]

24. *MICRO_USB2_captureStop*

[Function]

Stop capture operation.

[Format]

```
long MICRO_USB2_captureStop (  
    long id  
);
```

[Arguments]

id ...Device index

[Return]

usb2Success	...Capture operation is stopped successfully.
usb2Err_invalidDeviceIndex	...Invalid specified device index.
usb2Err_notAccess	...Failed to access specified device.
usb2Err_nowFirmUpdate	...Updating firmware.
usb2Err_firmCorrupted	...Firm is not properly written.

[Comment]

Capture operation is stopped.

[References]

MICRO_USB2_captureStart

Return Code

Symbol	Value
usb2Success	0
usb2Err_unsuccess	1
usb2Err_notList	2
usb2Err_notAccess	3
usb2Err_notFrame	4
usb2Err_notAcPower	5
usb2Err_notImage	6
usb2Err_notBuffer	7
usb2Err_notData	8
usb2Err_notDownload	9
Usb2Err_invalidDeviceIndex	10
usb2Err_invalidMode	11
usb2Err_invalidParameter	12
usb2Err_invalidTriggermode	13
usb2Err_invalidTriggerpolarity	14
usb2Err_invalidTriggeroutput	15
usb2Err_invalidBinning	16
usb2Err_invalidPosition	17
usb2Err_invalidSize	18
usb2Err_invalidGain	19
usb2Err_invalidOffset	20
usb2Err_invalidCoolingstatus	21
usb2Err_invalidFrame	22
usb2Err_invalidEventID	23
usb2Err_invalidEventhandle	24
usb2Err_invalidData	25
usb2Err_invalidIndex	26
usb2Err_invalidNumber	27
usb2Err_invalidAddress	28
usb2Err_invalidLength	29
usb2Err_overTime	30
usb2Err_underTime	31
usb2Err_overTemperature	32
usb2Err_underTemperature	33
usb2Err_overArea	34

usb2Err_nowCapturing	35
usb2Err_nowFirmUpdate	36
usb2Err_initialized	37
usb2Err_opened	38
usb2Err_existDevice	39
usb2Err_firmCorrupted	40
usb2Err_ReadImageStatus	41
usb2Err_InvalidEepromParameter	42