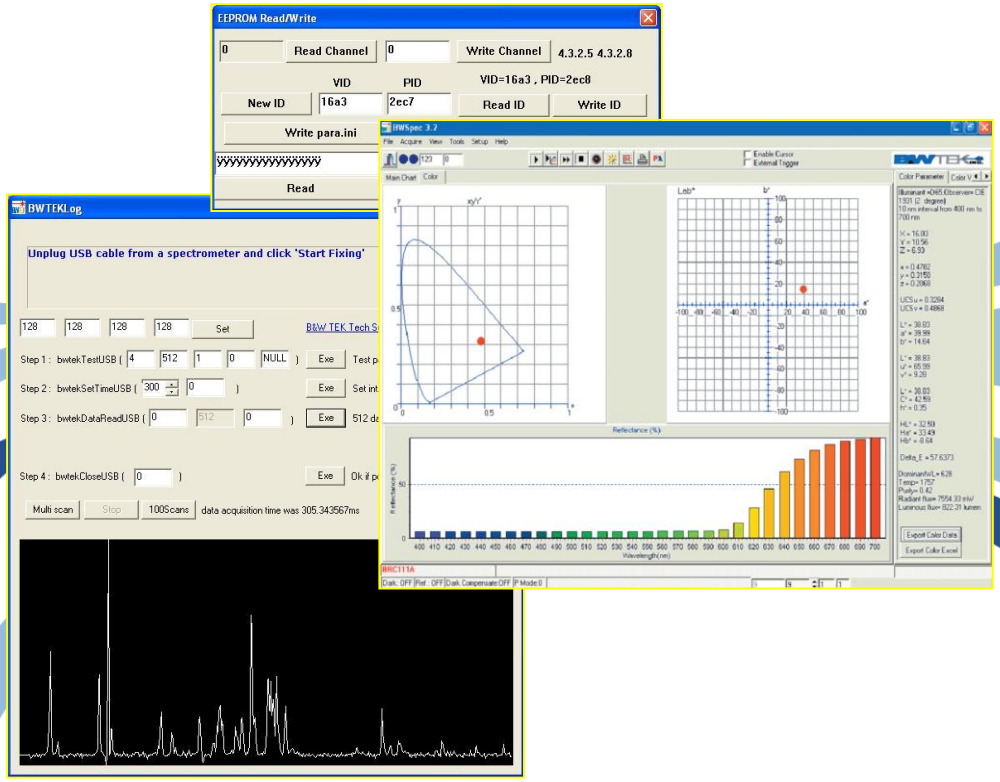


SDK-S
User Manual

Important Changes & Compatibility

USB Device Channel Numbers for spectrometers can no longer be assigned manually.
Channel Numbers are now dynamically assigned, see function ***bwtekSetupChannel*** for further details.

For quick reference on the order in which to call the new functions in this SDK as they relate to the important changes for Dynamically Assigned USB Device Channel Numbers, reference ***Appendix I: Dynamically Assigned Channel Numbers.***

Compatibility: If you are using an older version of the SDK-S or SDK-L which allows for USB Device Channel Numbers to be manually assigned, it will Not be compatible with this newer version which uses Dynamically Assigned USB Device Channel Numbers.

Introduction

B&W Tek, Inc. Software Developer's Kit (SDK) is designed for customers who need to develop a custom program for their USB based BTC and/or BRC spectrometers for specific applications for 32-Bit and 64-Bit windows based software.

For RS232 Devices, contact B&W Tek, Inc. with your model spectrometer and serial number and reference RS232 Command Set, (Doc# 4000000008).

The SDK program's .DLL files are compatible with any Windows XP, Windows 7 and Windows 8 32-Bit and 64-Bit Windows Operating Systems.

Version

This document is based on .DLL version #:

- **bwtekusb.dll (4.8.0.14)**

Installation

To use SDK-S you must first place specific files into folder on your computer. Find the Operating System of your Windows-based PC and then find the correct Bit OS you have. Follow the instructions for where to place the files on your PC.

1) Locate the Operating System and Bit (32-Bit 64-Bit) for your computer. Follow the instructions on where to place the files located into the SDK-S Package on your local computer.

2) File and Folder Destination Folders:

a) Windows XP

i) Windows XP - 32-Bit

(1) Windows XP – 32-Bit - USB 2.0 Capable Units:

- Copy "\32bit\Drivers\USB2\winxp\bwtekusb2.inf" → c:\windows\inf\
- Copy "\32bit\Drivers\USB2\winxp\ fx2lp.cat" → c:\windows\inf\
- Copy "\32bit\Drivers\USB2\winxp*.spt" → c:\windows\system32\drivers\bwtek\
- Copy "\32bit\Drivers\USB2\winxp\CyUSB.sys" → c:\windows\system32\Drivers\
- Copy "\32bit\Drivers\CyUSB.dll" files → c:\windows\system32\
- Copy "\32bit\DLL_32bit\bwtekusb.dll" → **the folder of your executable file.**

(2) Windows XP – 32-Bit - USB 3.0 Capable Units:

- Copy "\32bit\Drivers\USB3\winxp\bwtekusb3.inf" → c:\windows\inf\
- Copy "\32bit\Drivers\ USB3\winxp\cyusb3.cat" → c:\windows\inf\
- Copy "\32bit\Drivers\ USB3\winxp*.spt" → c:\windows\system32\drivers\bwtek\
- Copy "\32bit\Drivers\USB3\winxp\CyUSB3.sys" → c:\windows\system32\Drivers\
- Copy "\32bit\Drivers\USB3\winxp\WdfCoInstaller01009.dl" → c:\windows\system32\
- Copy "\32bit\Drivers\USB3\winxp\WdfCoInstaller01009.dll"→ c:\windows\system32\Drivers\
- Copy "\32bit\Drivers\CyUSB.dll" files → c:\windows\system32\
- Copy "\32bit\DLL_32bit\bwtekusb.dll" → **the folder of your executable file.**

b) Windows 7

i) Windows 7 - 32-Bit

(1) Windows 7 – 32-Bit - USB 2.0 Capable Units:

- Copy "\32bit\Drivers\USB2\win7_vista\bwtekusb2.inf" → c:\windows\inf\
- Copy "\32bit\Drivers\USB2\win7_vista\fx2lp.cat" → c:\windows\inf\
- Copy "\32bit\Drivers\USB2\win7_vista*.spt" → c:\windows\system32\drivers\bwtek\
- Copy "\32bit\Drivers\USB2\win7_vista\CyUSB.sys" → c:\windows\system32\Drivers\
- Copy "\32bit\Drivers\CyUSB.dll" → c:\windows\system32\
- Copy "\32bit\DLL_32bit\bwtekusb.dll" → c:\windows\system32\
- Copy "\32bit\DLL_32bit\bwtekusb.dll" → **the folder of your executable file.**

(2) Windows 7 – 32-Bit - USB 3.0 Capable Units:

- Copy "\32bit\Drivers\USB3\win7\bwtekusb3.inf" → c:\windows\inf\
- Copy "\32bit\Drivers\ USB3\win7\cyusb3.cat" → c:\windows\inf\
- Copy "\32bit\Drivers\ USB3\win7*.spt" → c:\windows\system32\drivers\bwtek\
- Copy "\32bit\Drivers\USB3\win7\CyUSB3.sys" → c:\windows\system32\Drivers\
- Copy "\32bit\Drivers\USB3\win7\WdfCoInstaller01009.dll" → c:\windows\system32\
- Copy "\32bit\Drivers\USB3\win7\WdfCoInstaller01009.dll" → c:\windows\system32\Drivers\
- Copy "\32bit\Drivers\CyUSB.dll" → c:\windows\system32\
- Copy "\32bit\DLL_32bit\bwtekusb.dll" → c:\windows\system32\
- Copy "\32bit\DLL_32bit\bwtekusb.dll" → **the folder of your executable file.**

ii) Windows 7 64-Bit

(1) Windows 7 – 64-Bit - USB 2.0 Capable Units:

- Copy "\64bit \Drivers\USB2\win7\bwtekusb2.inf" → c:\windows\inf\
- Copy "\64bit \Drivers\USB2\ win7\ fx2lp.cat" → c:\windows\inf\
- Copy "\64bit \Drivers\USB2\ win7*.spt" → c:\windows\system32\drivers\bwtek\
- Copy "\64bit \Drivers\USB2\ win7\CyUSB.sys" → c:\windows\system32\Drivers\
- Copy "\64bit \Drivers\CyUSB.dll" → c:\windows\system32\
- Copy "\64bit\DLL_64bit\bwtekusb.dll" → c:\windows\system32\
- Copy "\64bit\DLL_64bit\bwtekusb.dll" → **the folder of your executable file.**

(2) Windows 7 – 64-Bit - USB 3.0 Capable Units:

- Copy "\64bit\Drivers\USB3\win7\bwtekusb3.inf" → c:\windows\inf\
- Copy "\64bit\Drivers\ USB3\win7\cyusb3.cat" → c:\windows\inf\
- Copy "\64bit\Drivers\ USB3\win7*.spt" → c:\windows\system32\drivers\bwtek\
- Copy "\64bit\Drivers\USB3\win7\CyUSB3.sys" → c:\windows\system32\Drivers\
- Copy "\64bit\Drivers\USB3\win7\WdfCoInstaller01009.dll" → c:\windows\system32\
- Copy "\64bit\Drivers\USB3\win7\WdfCoInstaller01009.dll" → c:\windows\system32\Drivers\
- Copy "\64bit\Drivers\CyUSB.dll" files → c:\windows\system32\
- Copy "\64bit\DLL_64bit\bwtekusb.dll" → c:\windows\system32\
- Copy "\64bit\DLL_64bit\bwtekusb.dll" → **the folder of your executable file.**

c) Windows 8

i) Windows 8 - 32-Bit

(1) Windows 8 - 32-Bit – USB 2.0 Capable Units:

- Copy "\32bit\Drivers\USB2\win8\bwtekusb2.inf" → c:\windows\inf\
- Copy "\32bit\Drivers\USB2\win8\fx2lp.cat" → c:\windows\inf\
- Copy "\32bit\Drivers\USB2\win8*.spt" → c:\windows\system32\drivers\bwtek\
- Copy "\32bit\Drivers\USB2\win8\CyUSB.sys" → c:\windows\system32\Drivers\
- Copy "\32bit\Drivers\CyUSB.dll" → c:\windows\system32\
- Copy "\32bit\DLL_32bit\bwtekusb.dll" → c:\windows\system32\
- Copy "\32bit\DLL_32bit\bwtekusb.dll" → **the folder of your executable file.**

(2) Windows 8 - 32-Bit – USB 3.0 Capable Units:

- Copy "\32bit\Drivers\USB3\win8\bwtekusb3.inf" → c:\windows\inf\
- Copy "\32bit\Drivers\USB3\win8\cyusb3.cat" → c:\windows\inf\
- Copy "\32bit\Drivers\USB3\win8*.spt" → c:\windows\system32\drivers\bwtek\
- Copy "\32bit\Drivers\USB3\win8\CyUSB3.sys" → c:\windows\system32\Drivers\
- Copy "\32bit\Drivers\USB3\win8\WdfCoInstaller01009.dll" → c:\windows\system32\
- Copy "\32bit\Drivers\USB3\win8\WdfCoInstaller01009.dll" → c:\windows\system32\Drivers\
- Copy "\32bit\Drivers\CyUSB.dll" files → c:\windows\system32\
- Copy "\32bit\DLL_32bit\bwtekusb.dll" → c:\windows\system32\
- Copy "\32bit\DLL_32bit\bwtekusb.dll" → **the folder of your executable file.**

ii) Windows 8 - 64-Bit

(1) Windows 8 - 64-Bit – USB 2.0 Capable Units:

- Copy "\64bit\Drivers\USB2\win8\bwtekusb2.inf" → c:\windows\inf\
- Copy "\64bit\Drivers\USB2\win8\fx2lp.cat" → c:\windows\inf\
- Copy "\64bit\Drivers\USB2\win8*.spt" → c:\windows\system32\drivers\bwtek\
- Copy "\64bit\Drivers\USB2\win8\CyUSB.sys" → c:\windows\system32\Drivers\
- Copy "\64bit\Drivers\CyUSB.dll" → c:\windows\system32\
- Copy "\64bit\DLL_64bit\bwtekusb.dll" → c:\windows\system32\
- Copy "\64bit\DLL_64bit\bwtekusb.dll" → **the folder of your executable file.**

(2) Windows 8 - 64-Bit – USB 3.0 Capable Units:

- Copy "\64bit\Drivers\USB3\win8\bwtekusb3.inf" → c:\windows\inf\
- Copy "\64bit\Drivers\USB3\win8\cyusb3.cat" → c:\windows\inf\
- Copy "\64bit\Drivers\USB3\win8*.spt" → c:\windows\system32\drivers\bwtek\
- Copy "\64bit\Drivers\USB3\win8\CyUSB3.sys" → c:\windows\system32\Drivers\
- Copy "\64bit\Drivers\USB3\win8\WdfCoInstaller01009.dll" → c:\windows\system32\
- Copy "\64bit\Drivers\USB3\win8\WdfCoInstaller01009.dll" → c:\windows\system32\Drivers\
- Copy "\64bit\Drivers\CyUSB.dll" → c:\windows\system32\
- Copy "\64bit\DLL_64bit\bwtekusb.dll" → c:\windows\system32\
- Copy "\64bit\DLL_64bit\bwtekusb.dll" → **the folder of your executable file.**

USB 3.0/2.0/1.1 Interface Spectrometers

Below are the basic functions for the USB interface spectrometers for writing custom software.
Further in the document you will find Supplementary functions and model specific functions.

USB Basic Functions

InitDevices

BOOL InitDevices
(
);

This function initializes space in memory by creating a USB device object for all connected USB spectrometers.
*****This is the first function that should be called every time.**

RETURN
If the function call is successful, a True value will be returned.

bwtekSetupChannel

```
int bwtekSetupChannel
(
    int nFlag,                  // Flag Value should always be -1
    char *nChannelStatus        //Pointer to 32 byte array
);
```

This function *reads* the value of for the 32 available spectrometer channel numbers.
A total of 32 spectrometers can be used at one time, each with its own unique channel number.
Valid Channel Numbers for this array will be 0 – 31
***Note**: Channel Numbers with this version of the SDK are now dynamically assigned.*
Static Channels numbers are no longer able to be assigned.

nFlag -1 is used to get the value of all 32 available channel numbers.
(This will work even if you do Not have 32 spectrometers connected)

nChannelStatus is a pointer to a 32 byte memory array which saves the channel-number values for all 32 available spectrometers.
*Note: values in the array which are >=32 are Inactive channels.

RETURN
If the function call is successful it will return a positive integer. Otherwise it will return a negative integer.

bwtekReadEEPROMUSB

```
int bwtekReadEEPROMUSB
(
    char *OutFileName,      // The filename in which data from EEPROM will be saved
    int nChannel            // Channel number of spectrometer received from the 'bwtekSetupChannel' function
);
```

This function is used for retrieving data from the spectrometer's EEPROM

OutFileName is used to specify the name of the output file which will be saved to the computer. The output file has a text ASCII format and the default folder is the current directory unless otherwise specified.
File Naming Examples:
bwtekReadEEPROMUSB("C:\para.ini",0)
bwtekReadEEPROMUSB("eeprom.txt",1)

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
FAIL: If there is No active spectrometer assigned to a given channel number that is being read a -1 value will be returned.
PASS: A Positive value be returned AND a output file will be created in the folder location designated from above.

NOTE: *The output file contains a lot of information used for internal use. Some of this data, for example, the C-code for the spectrometer, calibration coefficients, timing mode, input mode and number of pixels on the spectrometer's detector may be useful.*

bwtekTestUSB

```
int bwtekTestUSB
(
    int nTimingMode,    // USB Interface timing option
    int nPixelNo,       // number of pixels of a detector to be readout
    int nInputMode,     // signal conditioning stage gain value
    int nChannel,       // channel to get data from
    int pParam          // setting for RS232 – Use NULL FOR USB INTERFACES
);
```

This function is for initializing communication to a specific USB spectrometer device while knowing its specific nChannel number assignment.

nTimingMode is used to specify the USB firmware timing option.
Note: See Appendix E for **nTimingMode** values for the spectrometer model you are using OR this information can be retrieved from the *OutFileName created from the bwtekReadEEPROMUSB function.

nPixelNo should be set to the number of pixels of used by the detector array in the spectrometer device.
Note: See Appendix E for **nPixelNo** values for the spectrometer model you are using OR this information can be retrieved from the *OutFileName created from the bwtekReadEEPROMUSB function.

nInputMode is used to specify the ADC input range used in the spectrometer device being programmed. It should be set to 0 for ADCs for unipolar input such as 0 to +5V or 0 to +10V. It should be set to 1 for ADCs using bipolar input such as -5 to +5V or -10 to +10V (BTC and BRC111), and 2 for ADCs for unipolar and non-inverting input (BTC600).
Note: See Appendix E for **nInputMode** values for the spectrometer model you are using OR this information can be retrieved from the *OutFileName created from the bwtekReadEEPROMUSB function.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

pParam This pointer should be NULL FOR ALL USB INTERFACES

Return
If the spectrometer readout is successful, a positive integer or 0 will be returned.

bwtekSetTimeUSB

```
long bwtekSetTimeUSB
(
   long lTime,        // integration time setting
   int nChannel       // channel to get data from
);
```

This function is for setting the spectrometer integration time as specified.
*BRC115E, BRC115U, and BRC115V model, lTime always is microsecond.
lTime Range: 1,000 → 2,100,000,000 (us)

*BTC261P, BTC264P model, lTime always is microsecond.
lTime Range: 200 → 2,100,000,000 (us)

*BTC655E, BTC665E model, lTime always is microsecond.
lTime Range: 6,000 → 2,100,000,000 (us)

lTime is the integration time value to be set
*Used in conjunction with the *bwtekSetTimeUnitUSB* function above.
Refer to Appendix F: Default Parameters for bwtekTESTUSB for integration time range for your spectrometers.

(1) For spectrometers that have **2D detectors installed, (BTC6xx series)**, an offset integration time, *bwtekSetTimeBase0USB*, results from the extra time the sensing pixels are exposed to during the readout process of the area sensor arrays. The actual Exposure time of the sensor is, set integration time + offset time. The *bwtekSetTimeBase0USB* needs to be taken into account when calculating the actual integration time. Therefore the lTime value to be passed to the DLL is calculated by subtracting the *offset time* from the desired integration time.

For Example:

If your Offset Time is 26 (ms) and desired integration time is 50 (ms).
You would NOT set lTime to 50
You MUST set *lTime* = (desired exposure time) – Offset Time
lTime = 50 – 26 --> *lTime* = 24

This will cause the minimal integration time for the spectrometer to be affected. The minimal integration time will be Offset Time + 1.

(2) For the BTC263 model, if your integration time is set in microseconds (us), the actual exposure time you will receive is your desired exposure time + 80us.

For Example:

If you want to set the integration time to 200 (us) [desired exposure time]
You MUST set *ITime* = (desired exposure time) + Offset Time (80 us)
ITime = 200 + 80 --> *ITime* = 280 (us)

If the BTC263 model integration time is set to 1 millisecond (ms), there is NO Offset Time.
For Example: If you want to set the integration time to 10 (ms) [desired exposure time]
ITime = 10 (ms)

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful it returns the new integration time value. Otherwise it will return a negative value.

bwtekSetTimeBase0USB

```
long bwtekSetTimeBase0USB
(
   long lOffsetTime,      // offset integration time setting
   int nChannel           // channel to get data from
);
```

This function is for setting the spectrometer offset integration time as specified.

lOffsetTime is the offset time base value in milliseconds to be set

.

Refer to Appendix B for Offset Integration Time settings for your spectrometer.

**If the Offset Integration Time value in Appendix B is 'NA' you do Not need to call this function for your spectrometer.*

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful it returns the new offset integration time value. Otherwise it will return negative value.

bwtekSetTimingsUSB

```
int bwtekSetTimingsUSB
(
   long lTriggerExit,        // Setting external trigger timeout
   int nMultiple,            // A multiplier factor for integration times needed > 65,535 ms
   int nChannel              //channel number to get data from
);
```

This function is used for setting both an external trigger timeout and a multiplier factor when an integration time is needed > 65,535ms.

lTriggerExit is used to specify the wait time for receipt of an external trigger. If no trigger signal is received within this set time, the spectrometer will automatically take a single scan.
The real timeout period is 15ms * *lTriggerExit*,
Maximum timeout period: 15ms * 65,535 = 983,040ms.
****To avoid the spectrometer/software from timing out and continuously wait for an external trigger, the lTriggerExit value should equal 0***

nMultiple is used to specify the multiplying factor for the integration time. The default time base is 1 when multiple = 1, when multiple = 2, the time base is 2 and real integration time will be 2 * *lTime* (*lTime* parameter is from the **bwtekSetTimeUSB** function. The *lTime* time can be adjusted by using **bwtekSetTimeUSB** function. The real integration time period is [*nMultiple* * (*lTime*, - *lOffsetTime*)]
nMultiple range is 1 - 16.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the spectrometer device becomes disconnected or loses communication a negative integer will be returned, indicating a failure. Otherwise the *lTriggerExit* value will be returned.

bwtekDataReadUSB

```
int bwtekDataReadUSB
(
   int nTriggerMode,            // Base address for plug-in data acquisition board
   unsigned short *pArray,      // data value array from the read operation stored
   int nChannel                 // channel to get data from
);
```

This function is for reading out data form detector

nTriggerMode is used to set the trigger mode to initiate a trigger scan process. It should be set to 0 for free running (continuous scanning) mode and 1 for external trigger mode.
The external trigger signal should be supplied as a 5V TTL *tpulse*.
It is falling edge effective.
*Refer to the spectrometer's hardware user manual for the trigger pulse width definition.

pArray is a pointer to a data array memory space, size will depend on the total number of pixels on the CCD for the spectrometer model that is being used. Every element in this array should be an unsigned integer with a minimum of 2 bytes for 16 bit resolution spectrometers
.
NOTE: The pArray size should be equal to int *nPixelNo* in the **bwtekTestUSB** function.
See Appendix A for Pixel Number Values

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the spectrometer readout is successful the number of data points read will be returned.
Else a negative will be returned.

Notes:
The pArray buffer needs to be REVERSED if the EEPROM contains the "xaxis_data_reverse=1" entry.
Refer to the X-axis Reverse Appendix for the setting relating to specific model types.

pArray[0] →pArray[nPixelNo-1]
pArray[1] →pArray[nPixelNo-2]
.....

pArray[nPixelNo-2] →pArray[1]
pArray[nPixelNo-1] →pArray[0]

Example how to get the “xaxis_data_reverse=1” entry from the EEPROM:
(1) Read EEPROM context to para.ini file →bwtekReadEEPROMUSB(“para.ini”,0)
(2) Search for x-axis reverse flag in the [COMMON] section of the para.ini file:
    [COMMON]
    xaxis_data_reverse=1
By default the xaxis_data_reverse is set to 0.

bwtekReadResultUSB

```
int bwtekReadResultUSB
(
   int nTriggerMode,              // setting for trigger mode
   int nAverage,                  // number of scans to be averaged
   int nTypeSmoothing,        // setting for smoothing type
   int nValueSmoothing,           // setting for smoothing parameter
   unsigned short *pArray,        // an array memory space for scan data
   int nChannel                   // setting for channel number
);
```

This function is for reading out data from the detector then applying data smoothing.

nTriggerMode is used to set the trigger mode to initiate a trigger scan process. It should be set to 0 for free running (continuous scanning) mode and 1 for external trigger mode. The external trigger signal should be supplied as a 5V TTL *tpulse*. It is falling edge effective.
Refer to hardware user manual for the trigger pulse width definition.

nAverage is used to set the number of scans to be averaged. (Set value to 1 for no averaging)
nAverage range 1 - 16

nTypeSmoothing 0 for no smoothing function, 1 for FFT smoothing, 2 for Savitzky-Golay smoothing, 3 for Boxcar smoothing.

nValueSmoothing

- When using FFT smoothing (**nTypeSmoothing=1)**, this parameter indicates the percentage of cutoff frequency. The **nValueSmoothing** should be 0 to 100.
- When using Savitzky-Golay smoothing (**nTypeSmoothing=2)**, The **nValueSmoothing** should be 2 to 5.

Example:

When using Savitzky-Golay smoothing, if nValueSmoothing is set to 4, the total number of smoothing points will be 9. 1 point will be the "origin" pixel and the other 8 will be 4 pixels before the origin and 4 pixels after the origin added together with weighted factors and divided by 9.

- When using Boxcar smoothing (**nTypeSmoothing=3)**, The **nValueSmoothing** should be the number of pixels to smooth.

$$\bar{x}[i] = \frac{1}{2M+1} \sum_{j=-M}^{M} x[i+j]$$

pArray is a pointer to the data array's memory space dependent on the total number of pixels on the CCD for the spectrometer model that is being used. Every element in this array should be an unsigned integer with minimum 2 bytes for a 16 bit resolution.

NOTE: This Value should be equal to *int nPixelNo* in the **bwtekTestUSB** function.
See Appendix A for Pixel Number Values

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the spectrometer readout is successful as the result of this function call will be 0. Otherwise the return value is negative.

Notes:

The pArray buffer needs to be REVERSED if the EEPROM contains the "xaxis_data_reverse=1" entry.

Refer to Appendix F X-axis Inverse Settings.

pArray[0] →pArray[nPixelNo-1]
pArray[1] →pArray[nPixelNo-2]
.....
pArray[nPixelNo-2] →pArray[1]
pArray[nPixelNo-1] →pArray[0]

Example how to get the "xaxis_data_reverse=1" entry from the EEPROM:

(1) Read EEPROM context to para.ini file →bwtekReadEEPROMUSB("para.ini",0)

(2) Search for x-axis reverse flag in the [COMMON] section of the para.ini file:

[COMMON]

xaxis_data_reverse=1

By default the xaxis_data_reverse is set to 0.

bwtekDataReadUSB1

```
int bwtekDataReadUSB1
(
   int nScanNo                 // scan number
   int nTriggerMode,           // base address for plug-in data acquisition board
   unsigned short *pArray,     // data value array from the read operation stored
   double *pStartTime,         // array save the start time of scan data, unit is ms
   double *pEndTime,           // array save the end time of scan, unit is ms
   int nChannel                // channel to get data from
);
```

This function is for reading out data form detector

nScanNo is the number of data scans.

nTriggerMode is used to set the trigger mode to initiate a trigger scan process. It should be set to 0 for free running (continuous scanning) mode and 1 for external trigger mode. The external trigger signal should be supplied as a 5V TTL *tpulse*. It is falling edge effective.
Refer to hardware user manual for the trigger pulse width definition.

pArray is a pointer to the data array's memory space dependent on the total number of pixels on the CCD for the spectrometer model that is being used. Every element in this array should be an unsigned integer with minimum 2 bytes for a 16 bit resolution.

NOTE: This Value should be equal to *int nPixelNo* in the **bwtekTestUSB** function.
See Appendix A for Pixel Number Values

pStartTime is a pointer to a start time array's memory space dependent on the number of data scans *nScanNo*.

pEndTime is a pointer to an end time array's memory spaces dependent on the number of data scans *nScanNo*.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the spectrometer readout is successful as the result of this function call it returns the positive value. Otherwise the return value is negative.

Notes:

The pArray buffer needs to be REVERSED if the EEPROM contains the “xaxis_data_reverse=1” entry.

Refer to Appendix F X-axis Inverse Settings.

pArray[0] →pArray[nPixelNo-1]
pArray[1] →pArray[nPixelNo-2]
.....
pArray[nPixelNo-2] →pArray[1]
pArray[nPixelNo-1] →pArray[0]

Example how to get the “xaxis_data_reverse=1” entry from the EEPROM:

(1) Read EEPROM context to para.ini file →bwtekReadEEPROMUSB(“para.ini”,0)

(2) Search for x-axis reverse flag in the [COMMON] section of the para.ini file:

    [COMMON]
    xaxis_data_reverse=1

By default the xaxis_data_reverse is set to 0.

bwtekGetTimeUSB

```
long bwtekGetTimeUSB
(
   long *lTime,          // reads back the integration time setting
   int nChannel          // channel to get data from
);
```

This function is for reading back the spectrometer's set integration time (*lTime*) which was set in the *bwtekSetTimeUSB* function.

***lTime** is a pointer to an array that will return the integration time value (*lTime*) which was set in the *bwtekSetTimeUSB* function.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful it returns the value 1. Otherwise it will return negative value or *lTime* value equal 0.

GetDeviceCount

```
int GetDeviceCount
(
);
```

This function is returns the number of spectrometer connected to the host computer.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

GetUSBType

```
int GetUSBType
(
   int *nUSBType,        // pointer to an array for the spectrometer's USB Type
    int nChannel         // channel to get data from
);
```

This function will return the spectrometer's USB type.

***nUSBType** is a pointer to an array that will return the physical USB Type interface of the connected spectrometer unit.

nUSBType = 2, it is USB2.0 unit, nUSBType=3, it is USB3.0 unit.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful it returns the value 1. Otherwise it will return a negative value.

bwtekGetCCode (Obsolete)

OBSOLETE - Use *GetCC0de* function below.

GetCCode

```
int GetCCode
(
  byte *CCode,          // returned CCode of spectrometer
  int nChannel          // channel to get data from
);
```

This function is for reading the c-code of the spectrometer off the spectrometer's EEPROM.

***CCode** is a pointer to a 32-byte array which contains the c-code of the spectrometer.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekCloseUSB

```
int bwtekCloseUSB
(
   int nChannel      // channel to get data from
);
```

This function is for ending communication to a specific USB spectrometer device while knowing its specific nChannel number assignment. This should be used when closing the programmed spectrometer application which was initiated by the bwtekTestUSB function call.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If closing USB connection is successful as the result of this function call it returns positive integer otherwise the return value is negative.

CloseDevices

```
int CloseDevices
(
);
```

This function is for closing all USB Devices. This is the last function that should be called when closing your software program which was initialed by the InitDevices function call.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

Supplementary Functions

bwtekSmoothingUSB

```
int bwtekSmoothingUSB
(
   int nTypeSmoothing,        // Smoothing type setting
   int nValueSmoothing,       // Smoothing parameter
   double *pArray,            // an array memory space for desire smoothing data
   int nNum                   // array length
);
```

This function is for applying data smoothing..

nTypeSmoothing 0 for BoxCar , 1 for FFT smoothing, 2 for Savitzky-Golay smoothing

nValueSmoothing

- BoxCar smoothing (**nTypeSmoothing = 0), nValueSmoothing =** number of pixels to smooth.

$$\bar{x}[i] = \frac{1}{2M+1} \sum_{j=-M}^{M} x[i+j]$$

- FFT smoothing (**nTypeSmoothing = 1)**, **nValueSmoothing** should be 0 to 100. This indicates the percentage of cutoff frequency.
- Savitzky-Golay smoothing (**nTypeSmoothing = 2)**, **nValueSmoothing** should be 2 to 5.

Example:

When using Savitzky-Golay smoothing, if nValueSmoothing is set to 4, the total number of smoothing points will be 9. 1 point will be the “origin” pixel and the other 8 will be 4 pixels before the origin & 4 pixels after the origin added together with weighted factors and divided by 9.

pArray is a pointer to an array containing the desired smoothing variables. It should range from 0 to (**nNum**-1). After *the bwtekSmoothingUSB* function is finished, this array will contain the smoothed data.

nNum is the total number of data points (length) in *pArray*

RETURN

If the function call is successful it will return a positive integer. Otherwise it will return a negative integer.

Note: Setting nTypeSmoothing = 0 and nTypeSmoothing = 0, will perform no smoothing.

bwtekConvertDerivativeDouble

```
int bwtekConvertDerivativeDouble
(
   int nSelectMethod,
   int nPointHalf,
   int nPolynominalOrder,
   int nDerivativeOrder,
   double *pRawArray,
   double *pResultArray,
   int nNum
);
```

This function is for data derivative.

nSelectMethod 0 for Point Diff, 1 for Savitzky-Golay.

nPointHalf is half of selected points.

nPolynominalOrder is degree of polynomial.

nDerivativeOrder is order of derivate. 0 is 1^{st} derivate. 1 is 2^{nd} derivate.

pRawArray is pointer of desired derivate data array.

pResultArray is pointer of derivate data array.

nNum is the total number of data points (length) in **pRawArray**

RETURN

If the function call is successful it will return a positive integer. Otherwise it will return a negative integer.

bwtekPolyFit

```
int bwtekPolyFit
(
   double *x,          // Array of independent variables
   double *y,          // Array of dependent variables
   int const numPts,   // Number of points in the independent and dependent arrays
   double *coefs,      // Pointer to array containing calculated coefficients [index from 0 to 'order' parameter]
   int const order     // Desired order of polynomial fit
);
```

This function is to perform a curve fit to a specified polynomial function based on supplied data values and 'least square' algorithm. This function may be used for converting pixel numbers to wavelengths.

If used for wavelength conversion, a wavelength calibration step needs to be performed, during which a calibration light source with known spectral features or a series of narrow band light radiation will be necessary. The known wavelength features of the calibration source along with the pixel numbers they are falling onto are the Y and X data values that need to be passed to the function for curve fitting. The array indices should range from 0 -- [Number_of_Points - 1]. They are passed to the function along with a desired polynomial fit order and an array large enough to hold the coefficients (). This array may be used with ***bwtekPolyCalc*** to calculate wavelength from pixels.

x is a pointer to an array containing the independent variables. It should range from 0 to (numPts-1).

y is a pointer to an array containing the dependent variables. It should range from 0 to (numPts-1).

numPts is the total number of data pairs in the variable arrays

coefs is a pointer to an array which contains the polynomial curve fit coefficients whose number of memory spaces are (*order* +1).

order is the desired polynomial curve fit order. The third order fit is recommended for use for pixel to wavelength conversions.

RETURN

If the function call is successful it will return a positive integer. Otherwise it will return a negative integer.

bwtekPolyCalc

```
void bwtekPolyCalc
(
   double *coefs,        // pointer to the polynomial coefficients
   int const order,      // polynomial order to use
   double const x,       // pixel number
   double *y             // wavelength value
);
```

This function performs calculations by using the following formula:

$$y = a0 + a1*x^1 + a2*x^2 + ... + aN*x^N$$

coefs is a pointer to an array containing the polynomial coefficients. These can be calculated using the *bwtekPolyFit* function from above.

order is the polynomial order to be used and must be less than or equal to the length of the **coefs** array.

x is the input variable, in this case, the pixel number.

y is the value to be calculated, in this case, wavelength (nm).

RETURN

If the function call is successful it will return a positive integer. Otherwise it will return a negative integer.

bwtekDataExport

```
int bwtekDataExport
(
   DataExport_Parameter_Struct *pDataExport_Parameter_Struct, // parameter structure of export
   SpectrometerCoefs_Struct *pSpectrometerCoefs_Struct,       // parameter structure of spectrometer coeff.
   double *pSrcArray,                                         // pointer to pixel data array
   double *pResultArray,                                      // pointer to converted data
   int *ResultArrayLength                                     // length of 'ResultArray'
);
```

This function is to be used to export x-axis data evenly.
X-axis units that can be exported evenly are the following: Wavelength / Wavenumber / Raman shift.

DataExport_Parameter_Struct is structure of export parameters.

```
    typedef struct DataExport_Parameter_Struct
{
      int nOptionX;          // 0 for wavelength, 1 for Wavenumber, 2 for Raman shift
      int nStartX;           // x-axis start location based on format (nOptionX)
      int nEndX;             // x-axis end location based on format (nOptionX)
      int nIncX;             // x-axis increments based on format (nOptionX)
      int nPixelNumber;      // Spectrometer detector's total number of Pixels (Refer to Appendix B)
}
```

SpectrometerCoefs_Struct is the structure of the spectrometer's coefficients.

```
typedef struct SpectrometerCoefs_Struct
{
      double fExcitationWavelength;   // FOR RAMAN SHIFT ONLY – Measured Excitation wavelength of Laser
      double fCoefsA[4];              // coefficients for pixel conversion to wavelength
      double fCoefsB[4];              // coefficients for wavelength conversion to pixel
}
```

- **fExcitationWavelength** is the value of measured laser excitation wavelength for Raman shift

- **fCoefsA** are the A coefficients for pixel to wavelength conversion. Can be found on the EEPROM of the spectrometer, "**coefs_a0... coefs_a3"**

- **fCoefsB** are the B coefficients for wavelength to pixel conversion. Can be found on the EEPROM of the spectrometer, "**coefs_b0... coefs_b3"**

pSrcArray is a pointer to the data source array, the data to be converted. This data will be in pixel format.

pResultArray is a pointer to the data which has been converted from the data source. This data will be in either evenly spaced; Wavelength, Wavenumber or Raman shift values.

pResultArrayLength is the length of the *pResultArray*.

RETURN

If the function call is successful it will return a positive integer. Otherwise it will return a negative integer.

Notes:

How to set the parameter of this function:

Example:

BTC112 Spectrometer coefficient's are as follows:

Coefs_a0…a3=321.107087610289 **/** 0.446657617771052 **/** -3.40125796972623E-5 / -1.27782145731415E-9

Coefs_b0…b3= -728.75518074818 **/** 2.29668100409253 **/** -0.00025083531593495 / 4.59557309079057E-7

Real Excitation wavelength= 784.85 nm

Export / Parameter	Wavelength (Evenly) 600--900 nm	Wave Number (Evenly) 19000--11000 (cm-1)	Raman Shift (Evenly) -6500 -- 1500 (cm-1)
nOptionX	0	1	2
nStartX	600 (nm)	19000 (cm-1)	-6500 (cm-1)
nEndX	900 (nm)	11000 (cm-1)	1500 (cm-1)
nIncX	1 (nm)	1 (cm-1)	1 (cm-1)
nPixelNumber	2048	2048	2048
fExcitationWavelength	784.85 (nm)	784.85 (nm)	784.85 (nm)
fCoefsA [0]	321.107087610289	321.107087610289	321.107087610289
fcoefsA [1]	0.446657617771052	0.446657617771052	0.446657617771052
fCoefsA [2]	-3.40125796972623E-5	-3.40125796972623E-5	-3.40125796972623E-5
fCoefsA [3]	-1.27782145731415E-9	-1.27782145731415E-9	-1.27782145731415E-9
fCoefsB [0]	-728.75518074818	-728.75518074818	-728.75518074818
fcoefsB [1]	2.29668100409253	2.29668100409253	2.29668100409253
fcoefsB [2]	-0.00025083531593495	-0.00025083531593495	-0.00025083531593495
fcoefsB [3]	4.59557309079057E-7	4.59557309079057E-7	4.59557309079057E-7

bwtekSaveEEPROMChannel (Obsolete)

OBSOLETE – Channel Numbers are now dynamically assigned.

bwtekGetXaxisInverseByte

```
int bwtekGetXaxisInverseByte
(
  int *nXaxisInverseByte     // returned x-axis inverse flag of the spectrometer
  int nChannel               // channel to get data from
);
```

This function is for reading the x-axis inverse flag of the spectrometer from the spectrometer's EEPROM.

***nXaxisInverseByte** is a pointer to an array that will contain the x-axis inverse value of the spectrometer.
(1 = x-axis inversed, 0 = x-axis NOT inversed)

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

USB 3.0 Interface Spectrometer

bwtekDSPDataReadUSB

```
int bwtekDSPDataReadUSB
(
   int nAveNum,
   int nSmoothing,
   int nDarkCompensate,
   int nTriggerMode,
   unsigned short *pArray,
   int nChannel
);
```

'Smart Scan Mode'

This function applies onboard averaging, smoothing, dark compensate on the firmware side of the spectrometer before outputting the data to the computer.

***Note:** This function will work ONLY with spectrometer with a USB 3.0 interface connector.

nAveNum is used to set the number of scans to be averaged, (Range 0 – 65535)

nSmoothing is used to apply smoothing to the scanned data. 0 for no smoothing function, 1 for Savitzky-Golay smoothing.

***NOTE**: Only Savitzky-Golay is available for smoothing.

nDarkCompensate is used to apply dark compensate to the scanned data. 0 for no dark compensate function, 1 for dark compensate function.

nTriggerMode is used to set trigger mode to initiate a trigger scan process.

0 = Free Running Mode, 1 = External Trigger Mode.

The external trigger signal should be supplied as a 5V TTL pulse. It is falling edge effective.

Refer to hardware user manual for the trigger pulse width definitions.

pArray is a pointer to data array memory spaces dependent on the total number of pixels on the CCD for the spectrometer model that is being used. Every element in this array should be an unsigned integer with minimum 2 bytes for a 16 bit resolution.

***NOTE**: This Value should be equal to int *nPixelNo* in the **bwtekTestUSB** function.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the spectrometer readout is successful the number of data points read will be returned. Otherwise a negative value will be returned.

bwtekFrameDataReadUSB

```
int bwtekFrameDataReadUSB
(
   int nFrameNum,
   int nTriggerMode,
   unsigned short *pArray,
   int nChannel
);
```

***Update Note: This Function was formally called bwtekDataReadUSB**

'Burst Mode'

This function allows for the spectrometer to acquire X number of scans (nFrameNum) onboard on the firmware side of the spectrometer before outputting the data to the computer.

***Note:** This function will work ONLY with spectrometer with a USB 3.0 interface connector.

For Example: BRC115E

.

nFrameNum is the number of scans to be acquired.

nTriggerMode is used to set trigger mode to initiate a trigger scan process.

0 = Free Running Mode, 1 = External Trigger Mode.

The external trigger signal should be supplied as a 5V TTL pulse. It is falling edge effective.

Refer to hardware user manual for the trigger pulse width definitions.

pArray is a pointer to data array memory spaces dependent on the total number of pixels on the CCD for the spectrometer model that is being used. Every element in this array should be an unsigned integer with minimum 2 bytes for a 16 bit resolution.

***NOTE**: This Value should be equal to int *nPixelNo* in the **bwtekTestUSB** function.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the spectrometer readout is successful the number of data points read will be returned. Otherwise a negative value will be returned.

bwtekWriteBlockUSB

```
int bwtekWriteBlockUSB
(
   unsigned int nAddress,
   byte *pArray,
   int nNum,
   int nChannel
);
```

This function is to write to a reserved location on the EEPROM by the customer.
This memory block must be first erased before rewriting to this location.
See bwtekEraseBlockUSB function below.
***NOTE**: Only USB3.0 unit support this function.

nAddress is an address in EEPROM to write, location: 0 – 0xFFFF bytes.

pArray points to the data array memory space.

nNum is the number of desired bytes to be written: (0 – 0xFFFF bytes)
***NOTE**: nAddress + nNum MUST BE <= 0xFFFF

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned

bwtekReadBlockUSB

```
int bwtekReadBlockUSB
(
   unsigned int nAddress,
   byte *pArray,
   int nNum,
   int nChannel
);
```

This function is to Read the reserved location on the EEPROM written to by the customer.
***NOTE**: Only USB3.0 unit support this function.

nAddress is an address in EEPROM to read, location: 0 – 0xFFFF bytes.

pArray points to the data array memory space.

nNum is the number of desired bytes to be read: (0 – 0xFFFF bytes)
***NOTE**: nAddress + nNum MUST BE <= 0xFFFF

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned

bwtekEraseBlockUSB

```
int bwtekEraseBlockUSB
(
   int nChannel
);
```

This function is to ERASE the entire block memory reserved location (0 – 0xFFFF bytes) on the EEPROM written to by the customer.
***NOTE**: Only USB3.0 unit support this function.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned

bwtekReadTemperature

```
int bwtekReadTemperature
(
   int nCommand,
   int *nADValue,              \\ AD Value Range: 0 - 4095
   double *nVoltageValue,      \\ Temperature will be in Celsius
   int nChannel
);
```

This function is for reading the temperature of the CCD Detector OR read the ambient temperature.

nCommand is a flag of specific temperature:
0x10 for CCD detector temperature,
0x11 for ambient temperature reading.

***nADValue** is a pointer to an integer return value. It is reserved.

***nVoltageValue** is a pointer to float pointer value of temperature. The temperature units are degrees Celsius.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

AUX Port Functions 1 - Multi-Purpose TTL Output Functions

bwtekGetExtStatus

```
int bwtekGetExtStatus
(
   int nChannel      //channel to get data from
);
```

This function is used to retrieve the status of the TTL Input pin.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the status of input pin is low, a 0 will be returned. If the status of input pin is high, a positive integer will be returned. If the function call is unsuccessful, a negative integer will be returned.

bwtekSetExtLaser

```
int bwtekSetExtLaser
(
   int nOnOff,       //switch for On/Off
   int nChannel      //channel to get data from
);
```

This function is used for controlling the On/Off capabilities of an external device, typically used for Laser control, through the Spectrometer's AUX Port, when wired correctly.

nOnOff is used to set the external control signal. If 0 is assigned, the output pin signal will be low. If 1 is assigned, the output pin signal will be high.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekSetExtShutter

```
int bwtekSetExtShutter
(
   int nOnOff,        // switch for On/Off
   int nChannel       // channel to get data from
);
```

This function is used for controlling the Open/Closed capabilities of a Shutter (If Installed).

nOnOff is used to set the Shutter control signal. If 0 is assigned the output signal will be low. If 1 is assigned the output signal will be high.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

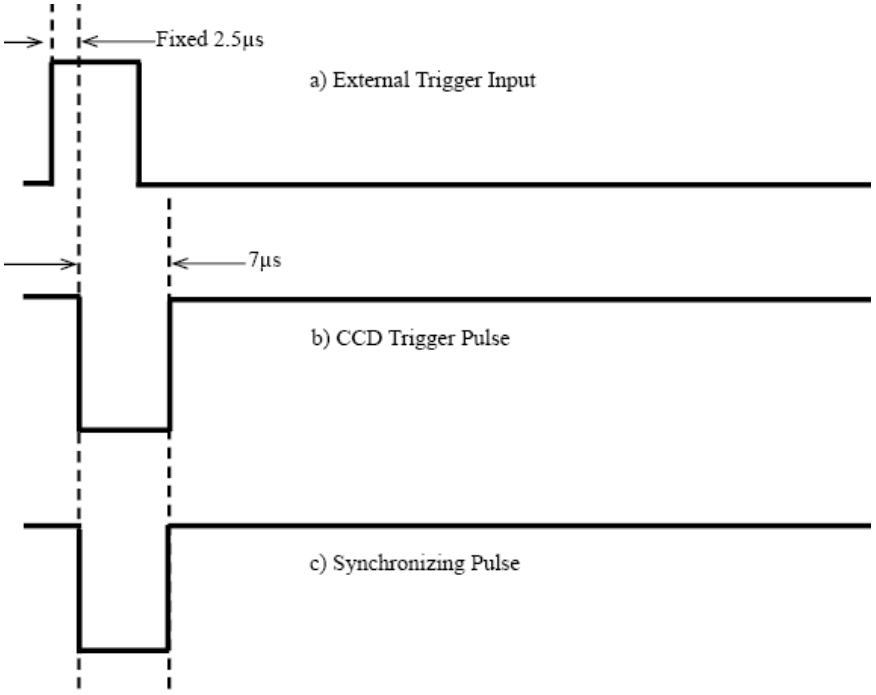
RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekSetExtSync

```
int bwtekSetExtSync
(
   int nOnOff,       //switch for On/Off
   int nChannel      //channel to get data from
);
```

This function is used to generate a pulse, which synchronizes an outside source with the external trigger signal for the spectrometer for detector scanning.



nOnOff is used to control the synchronizing pulse. If 0 assigned, the output signal will not be generated (no signal). If 1 is assigned, the output signal will be generated (pulse signal).

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

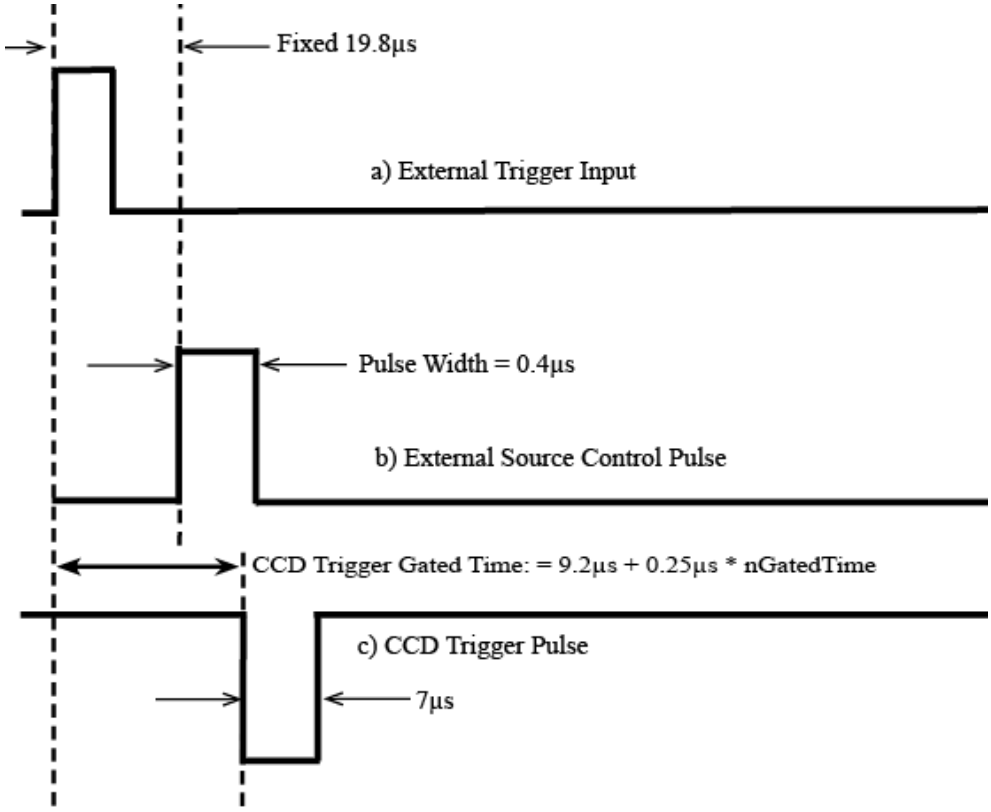
RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekGatedMode

```
int bwtekGatedMode
(
   int nGatedTime,      // Gated Time between an External trigger and a CCD trigger
   int nChannel         // channel to get data from
);
```

This function is for controlling CCD gated delay time for specific models:
See Appendix H for list of spectrometer models which can use this function.



nGatedTime is used to determine the time delay before the CCD starts to acquire data after an external pulse has been delivered. The parameter can be assigned an integer value of 0 to 65535. This value is in addition to the inherent delay time.
Example: When nGatedTime is 44 the gated time will be:
(9.2µs + 0.25µs*44) = 20.2µs = (9.2µs + 0.25µs*nGatedTime)
****NOTE**** When 0 is assigned to nGatedTime, the function will be disabled.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

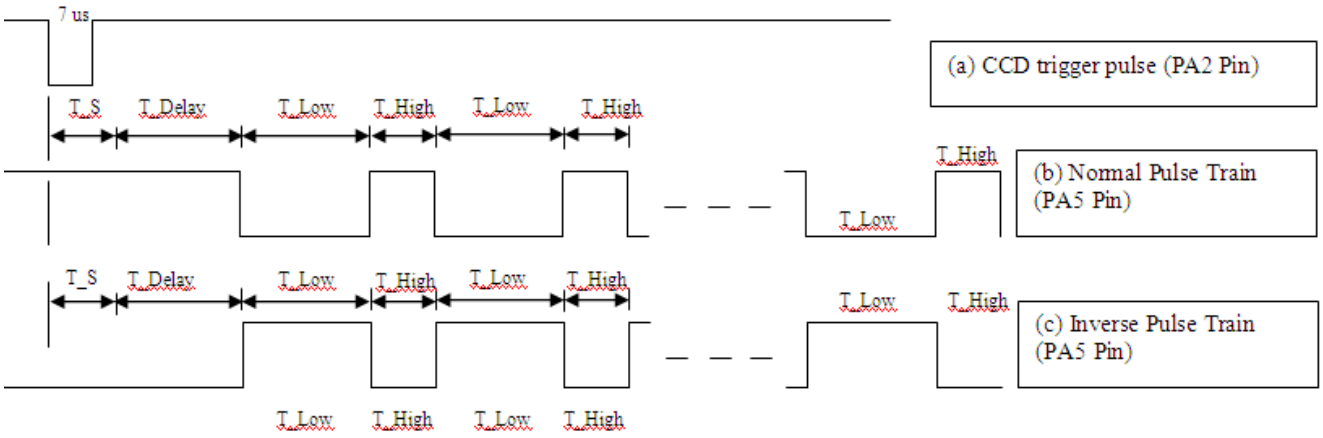
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekSetExtPulse

```
int bwtekSetExtPulse
(
   int nOnOff,          // switch to turn on or off pulse train
   int nDelayTime,      // delay time between the CCD trigger and the first pulse
   int nHigh,           // time duration for high pulse width
   int nLow,            // time duration for low pulse width
   int nPulse,          // number of pulses to be generated
   int nInverse,        // pulse output will be inverted
   int nChannel         // channel to get data from
);
```

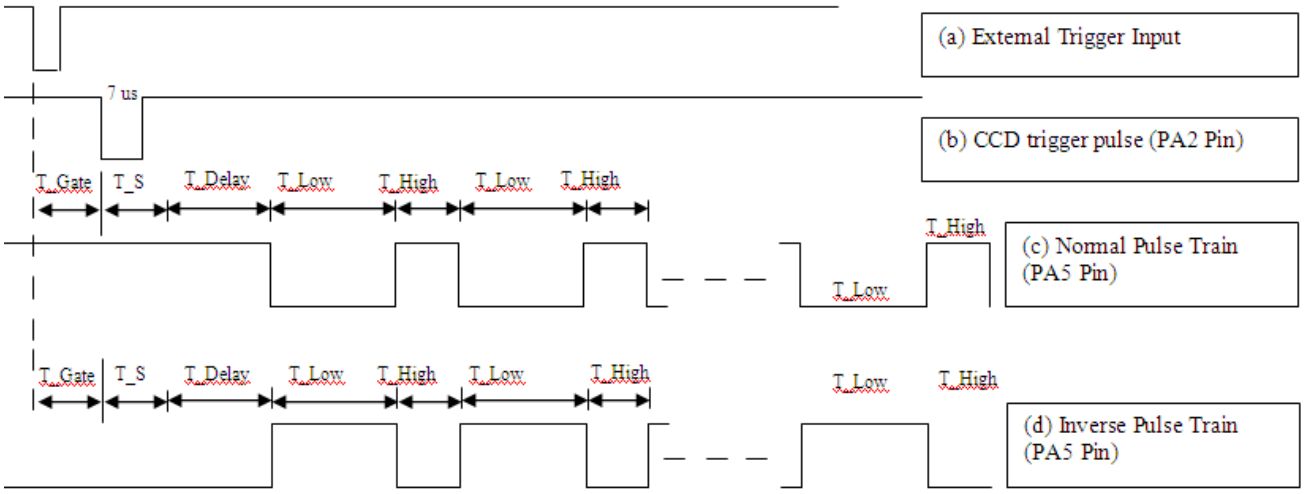
To define output pulse, use this function first before acquiring data.
This function is used to generate a pulse train defined by nHigh, nLow and nPulse.
This *bwteSetExtPulse* function would be called Before the *bwtekDataReadUSB* function.

Internal Trigger Mode



USB Models	T_S (Software access time, const value)	T_Delay	T_High	T_Low
BRC111A	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC112P	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC112E	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC115E	N/A	N/A	N/A	N/A
BRC115U	N/A	N/A	N/A	N/A
BRC115V	N/A	N/A	N/A	N/A
BRC642E-2048	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC711E-512	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC711E-1024	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC741E-512	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC741E-1024	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC111E	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC112E	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC162E	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC261E-256	N/A	N/A	N/A	N/A
BTC261E-512	N/A	N/A	N/A	N/A
BTC261E-1024	N/A	N/A	N/A	N/A
BTC261P-512	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC262A-256	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC262E-256	N/A	N/A	N/A	N/A
BTC262E-512	N/A	N/A	N/A	N/A
BTC262P-512	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC263E-256	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC264P-256	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC264P-512	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC264P-1024	9.6us	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC611E-512	9.6us	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC611E-1024	9.6us	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC613E-512	9.6us	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC613E-1024	9.6us	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC621E-512	9.6us	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC621E-1024	9.6us	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC651E-512	9.6us	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC651E-1024	9.6us	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC655E	N/A	N/A	N/A	N/A
BTC661E	9.6us	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC665E	N/A	N/A	N/A	N/A

External Trigger Mode



USB Models	T_S (Software access time, it is const value)	T_Gate	T_Delay	T_High	T_Low
BRC111A	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC112P	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC112E	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC115E	N/A	N/A	N/A	N/A	N/A
BRC115U	N/A	N/A	N/A	N/A	N/A
BRC115V	N/A	N/A	N/A	N/A	N/A
BRC642E-2048	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC711E-512	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC711E-1024	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC741E-512	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BRC741E-1024	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC111E	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC112E	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC162E	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC261E-256	N/A	N/A	N/A	N/A	N/A
BTC261E-512	N/A	N/A	N/A	N/A	N/A
BTC261E-1024	N/A	N/A	N/A	N/A	N/A
BTC261P-512	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC262A-256	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC262E-256	N/A	N/A	N/A	N/A	N/A
BTC262E-512	N/A	N/A	N/A	N/A	N/A
BTC262P-512	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC263E-256	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC264P-256	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC264P-512	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC264P-1024	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*nHigh	0.25us*nLow
BTC611E-512	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC611E-1024	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC613E-512	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC613E-1024	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC621E-512	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC621E-1024	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC651E-512	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC651E-1024	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC655E	N/A	N/A	N/A	N/A	N/A
BTC661E	9.6us	0.25us*nGateTime	0.25us*(nDelayTime+1)	0.25us*(65536-nHigh)	0.25us*(65536-nLow)
BTC665E	N/A	N/A	N/A	N/A	N/A

nOnOff will turn on or off the output pulse train. If 0 is assigned, the output signal train will not be generated (no signal). If 1 is assigned, the output signal train will be synchronized with the CCD trigger pulse that is generated after the normal delay time of (0.25us*(nDelayTime+1)).

nDelayTime is used to determine the delay time between the CCD trigger and the first pulse. This parameter can be assigned an integer value of 1 to 65535. This value is in addition to the inherent delay time. Example: When nDelayTime is 50, the total delay time will be:
12.75us=(0.25us*(nDelayTime+1))

nHigh is used to determine the high pulse width. The parameter can be assigned an integer value of 1 to 65535. This value is in addition to the inherent delay time.
Example:
BTC611E - When nHigh is 63, the high pulse width will be 16368.25us=0.25us*(65536-nHigh)

Other (BTC112, BTC261, BTC641 etc.) - When nHigh is 63, the high pulse width will be 15.75us=(0.25us*nHigh)

nLow is used to determine the low pulse width. The parameter can be assigned an integer value of 1 to 65535. This value is in addition to the inherent delay time.
Example:
BTC611E - When nLow is 63, the low pulse width will be 16368.25us=0.25us*(65536-nLow)

Other (BTC112, BTC261, BTC641 etc.) - When nLow is 63, the low pulse width will be 15.75us=(0.25us*nLow)

nPulse is used to set the number of pulses that are to be generated. The parameter can be assigned an integer value of 1 to 65535.

****NOTE**** In the function bwtekSetTime the value assigned to the lTime parameter must be longer than (T_S+T_Gate+T_Delay+(T_High+T_Low)*nPulse) if using external trigger. Or must be longer than (T_S+T_Delay+(T_High+T_Low)*nPulse) if using internal trigger.

nInverse is used to invert the output pulse train. 1 will output the inversed pulse train. 0 will output normal pulse train.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

AUX Port Functions 2---Shutter Functions:

The following shutter functions work for specific model types which have shutters installed.

bwtekShutterOpen (OBSOLETE)

```
int bwtekShutterOpen
(
   int nChannel         // channel to get data from
);
```

Please use the below function, *bwtekShutterControl* in place of this function.

bwtekShutterClose (OBSOLETE)

```
int bwtekShutterClose
(
   int nChannel         // channel to get data from
);
```

Please use the below function, *bwtekShutterControl* in place of this function.

bwtekShutterControl

```
int bwtekShutterControl
(
   int nSetShutter1,      // sets Shutter 1 state
   int nSetShutter2,      // sets Shutter 2 state
   int nChannel           // channel to get data from
);
```

This function controls the 'Shutter 1' and 'Shutter 2' of a Spectrometer's AUX Port pin assignment.
**Note:* *Check your spectrometer's User Manual Aux Pin assignment to see if 'Shutter 1' is a valid Pin Assignment. If 'Shutter 1' does not appear, this function will Not work with your spectrometer.*

nSetShutter1 is used to set Shutter 1 to open or close.
0 = Close Shutter, 1 = Open Shutter, -1 = No Action.

nSetShutter2 is used to set Shutter 2 to open or close.
0 = Close Shutter, 1= Open Shutter, -1 = No Action.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

AUX Port Functions 3

bwtekGetExtStatus

```
int bwtekGetExtStatus
(
   int nChannel      //channel to get data from
);
```

This function is used to retrieve the status of the TTL Input pin.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the status of input pin is low, a 0 will be returned. If the status of input pin is high, a positive integer will be returned. If the function call is unsuccessful, a negative integer will be returned.

bwtekGetTTLIn

```
int bwtekGetTTLIn
(
   int nNo,              // TTL Sensing Number
   int *nTTLStatus,      // Return value of TTL Input Signal
   int nChannel          // Channel
);
```

This function is used to retrieve the status of the TTL input pin.
(Reference the Spectrometer's User Manual for Pin location)

nNo is used to select which TTL Input feature to select from the spectrometer's AUX port. The default value should be 0 when the spectrometer has only a single TTL Input or selecting the first TTL Input pin. If using the second TTL Input feature a value of 1 should be used.
***NOTE** this Value is Not related to the actual Pin Number.

***nTTLStatus** is a pointer to an array that will return the status of a TTL Sensing Input signal, where 1 = High and 0 = Low.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekSetTTLOut

```
int bwtekSetTTLOut
(
   int nNo,              // TTL Output Number
  int nTTLStatus,        // Set the TTL output status
   int nInverse,         // TTL pin signal inversed
   int nChannel          // Channel
);
```

This function is used to set the output pin On / Off (High / Low).

This function is used for controlling the TTL Out 1 (See Spectrometer User Manual for AUX Pin definitions) when properly connected to a Laser Control Line. This function controls the On/Off status of the Laser.

nNo is used to select which TTL Output feature to select from the spectrometer's AUX port. The default value should be 0 when the spectrometer has only a single TTL Output or selecting the first TTL Output pin. If using the second TTL Output feature a value of 1 should be used.
***NOTE** this Value is Not related to the actual Pin Number.

nTTLStatus is used to set the status of an TTL output signal. It can be set from 0 to 1. (0 = Low, 1 = High)

nInverse is used to invert the output pin signal for **nTTLStatus**. 1 will invert the **nTTLStatus** values, and 0 will keep them at default.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekGetAnalogIn

```
int bwtekGetAnalogIn
(
   int nNo,              // Analog Input Number
  int *nADValue,         // AD value
   double *lValtage,     // Voltage value
   int nChannel          // Channel
);
```

This function is used to get the value of analog signal input (**Analog Sensing**).

nNo is used to select the Analog Input Number feature to select from the spectrometer's AUX port. The default value should be 0 when the spectrometer has only a single TTL Output or selecting the first TTL Output pin. If using the second TTL Output feature a value of 1 should be used.
***NOTE** this Value is Not related to the actual Pin Number.

nADValue is A/D Value of analog input. Valid range is 0...4095 (12 bit)

lVoltage is voltage of analog input. Valid range is 0…2.5 (V)

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

BTC261E & BTC262E Functions:

The following functions work only with the BTC261E & BTC262E spectrometers.

bwtekSetABGain (Obsolete)

*** Obsolete Note*** Units built **before** 2008 can still use this function.
Units built around 2008 and later this function will not work.
The ability to control the A and B Channels of the detector was changed from Software Control to Hardware Control. If your spectrometer has at least two trimpots on the front panel, at the bottom left corner from the SMA Plate then your unit has Hardware Control. If you are still unsure if you can use this function, please contact B&W Tek, Inc. (www.bwtek.com)

```
int bwtekSetABGain
(
   int nAB,          // select between the odd(A) and even(B) pixel channels
   int nGain,        // set the gain value
   int nChannel      // channel to get data from
);
```

This function is for setting the gain value of the analog output on either A or B channel.

nAB is used to choose between the **odd(A)** and **even(B)** pixel channels. Use 0 to control the odd(A) pixel channel and use 1 to control the even(B) pixel channel.
**NOTE For the BTC262E nAB must always be 0. The BTC262E uses one channel for both odd and even pixels.

nGain is used to set the gain value of an analog output signal. It can be set from 1 to 1023.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative interger will be returned.

Notes: When this function is used, the difference between desired set value and current value should less than 90. For example, the current value that you read out is 100, you want to set 1000. So you should set it as following code:

```
Dim OldValue as integer
Dim NewValue as integer
Dim tmp_Count as integer
Dim i ,channel as integer

Channel=0
OldValue=100
NewValue=1000
tmp_count=Int (abs(NewValue-OldValue+1))/90
For i=1 to tmp_count
  tmp_ret:=bwtekSetABGain(0,OldValue+ i*90,channel)
Next i
tmp_ret=bwtekSetABGain(0, NewValue, channel)
```


bwtekSetABOffset (Obsolete)

*** Obsolete Note*** Units built **before** 2008 can still use this function.
Units built around 2008 and later this function will not work.
The ability to control the A and B Channels of the detector was changed from Software Control to Hardware Control. If your spectrometer has at least two trimpots on the front panel, at the bottom left corner from the SMA Plate then your unit has Hardware Control. If you are still unsure if you can use this function, please contact B&W Tek, Inc. (www.bwtek.com)

```
int bwtekSetABOffset
(
   int nAB,          // select between the odd(A) and the even(B) pixel channels
   int nOffset,      // set the offset value
   int nChannel      // channel to get data from
);
```

This function is for setting the offset value of the analog output on either A or B channel

nAB is used to choose between the **odd(A)** and **even(B)** pixel channels. Use 0 to control the odd (A) pixel channel and use 1 to control the even (B) pixel channel.
**NOTE For the BTC262E nAB must always be 0. The BTC262E uses one channel for both odd and even pixels.

nOffset is used to set the offset value of an analog output signal. It can be set from 1 to 1023.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

Notes: when calling this function, the difference between desired set value and current value should less than 90.
For example, if the current readout value is 100 and you want to set it to 1000, use the following code:

```
Dim OldValue as integer
Dim NewValue as integer
Dim tmp_Count as integer
Dim i ,channel as integer

Channel=0
```



```
OldValue=100
NewValue=1000
tmp_count=Int (abs(NewValue-OldValue+1))/90
For i=1 to tmp_count
  tmp_ret:= bwtekSetABOffset (0,OldValue+ i*90,channel)
Next i
tmp_ret= bwtekSetABOffset (0, NewValue, channel)
```


bwtekGetABGain (Obsolete)

*** Obsolete Note*** Units built **before** 2008 can still use this function.
Units built around 2008 and later this function will not work.
The ability to control the A and B Channels of the detector was changed from Software Control to Hardware Control. If your spectrometer has at least two trimpots on the front panel, at the bottom left corner from the SMA Plate then your unit has Hardware Control. If you are still unsure if you can use this function, please contact B&W Tek, Inc. (www.bwtek.com)

```
int bwtekGetABGain
(
  int nAB,          // select between the odd(A) and even(B) pixel channels
  int *nGain,       // return value of gain
  int nChannel      // channel to get data from
);
```

This function is for getting the gain value of the analog output on either A or B channel

nAB is used to choose between the **odd(A)** and **even(B)** pixel channels. Use 0 to control the odd (A) pixel channel and use 1 to control the even (B) pixel channel.
**NOTE For the BTC262E nAB must always be 0. The BTC262E uses one channel for both odd and even pixels.

***nGain** is a pointer to an array that will return the gain value of an analog output signal.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekGetABOffset (Obsolete)

*** Obsolete Note*** Units built **before** 2008 can still use this function.
Units built around 2008 and later this function will not work.
The ability to control the A and B Channels of the detector was changed from Software Control to Hardware Control. If your spectrometer has at least two trimpots on the front panel, at the bottom left corner from the SMA Plate then your unit has Hardware Control. If you are still unsure if you can use this function, please contact B&W Tek, Inc. (www.bwtek.com)

```
int bwtekGetABOffset
(
  int nAB,          // select between the odd(A) and even(B) pixel channels
  int *nOffset,     // return value of offset
  int nChannel      // channel to get data from
);
```

This function is for getting the offset value of the analog output on either A or B channel

nAB is used to choose between the **odd(A)** and **even (B)** pixel channels. Use 0 to control the odd (A) pixel channel and use 1 to control the even (B) pixel channel.
**NOTE For the BTC262E nAB must always be 0. The BTC262E uses one channel for both odd and even pixels.

***nOffset** is a pointer to an array that will return the offset value of an analog output signal.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekSetInGaAsMode

```
int bwtekSetInGaAsMode
(
   int nMode,        // select feedback capacitor mode
   int nChannel      // channel to get data from
);
```

This function is for choosing the detector mode: high dynamic range or high sensitivity range.

nMode is used to choose the feedback capacitor mode of the detector.

nMode	Detector Mode
0	High Sensitivity
1	High Dynamic Range

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekGetInGaAsMode

```
int bwtekGetInGaAsMode
(
   int *nMode,       // select detector mode
   int nChannel      // channel to get data from
);
```

This function is for retrieving the detector mode setting: high dynamic range or high sensitivity range.

nMode is a pointer to an array that will return the value of the feedback capacitor mode setting.

*nMode	Detector Mode
0	High Sensitivity
1	High Dynamic Range

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekQueryTemperature

```
int bwtekQueryTemperature
(
   int nCommand,
   int *nReserved,
   double *nTempValue,
   int nChannel
);
```

This function is for reading the temperature of the CCD Detector OR temperature of the sample holder.
****Note** your spectrometer hardware must be compatible for this function to work.
If this function is not working for you correct, contact BWTEK to verify that your spectrometer has compatible hardware for this function.

nCommand is a flag of specific temperature. 11 for BTC261E CCD detector temperature, 81 for BTC261E sample holder temperature.

***nReserved** is a pointer to an integer return value. It is reserved.

***nTempValue** is a pointer to float pointer value of temperature. The temperature units are degrees Celsius.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekAccessDeltaTemp

```
int bwtekAccessDeltaTemp
(
   int nReadWrite,
   double *pDeltaTempValue,
   int nChannel
);
```

This function is for read/write the adjust value of temperature of sample holder and CCD Detector.

nReadWrite is a flag of Read or write. 0 for read delta of temperature, 1 for write delta of temperature.

***nDeltaTempValue** is a pointer to float pointer value of temperature delta. The temperature units are degrees Celsius. The real temperature is sum of **nDeltaTempValue** and **nTempValue**

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekAccessDeltaTemp1

```
int bwtekAccessDeltaTemp1
(
   int nReadWrite,
   double *pDeltaTempValue,
   double *pDeltaTempValue1,
   int nChannel
);
```

This function is for read/write the adjust value of temperature of sample holder and CCD Detector.

nReadWrite is a flag of Read or write. 0 for read delta of temperature, 1 for write delta of temperature.

***nDeltaTempValue** is a pointer to float pointer value of temperature delta of sample holder. The temperature units are degrees Celsius. The real temperature is sum of **nDeltaTempValue** and **nTempValue.** Please see notes for detail information.

***nDeltaTempValue1** is a pointer to float pointer value of temperature delta of BTC261E/BTC262E CCD detector. The temperature units are degrees Celsius. The real temperature is sum of **nDeltaTempValue1** and **nTempValue1** . Please see notes for detail information.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

Notes:

(1) For BTC261E/BTC262E sample holder temperature

Call function bwtekQueryTemperature (81, nReserved, **nTempValue,** channel), get value **nTempValue**
Call function BwtekAccessDeltaTemp1 (0, **nDeltaTempValue,** nDeltaTempValue1**,** channel), get value **nDeltaTempValue**

The BTC261E/BTC262E sample holder temperature equal (**nTempValue** + **nDeltaTempValue**)

(2) For BTC261E/BTC262E CCD detector temperature

Call function bwtekQueryTemperature (11, nReserved, **nTempValue1,** channel), get value **nTempValue1**
Call function BwtekAccessDeltaTemp1 (0, nDeltaTempValue, **nDeltaTempValue1,** channel), get value **nDeltaTempValue1**

The BTC261E/BTC262E CCD detector temperature equal (**nTempValue1** + **nDeltaTempValue1**)

bwtekWriteValue

```
int bwtekWriteValue
(
   int nItem,
   int nSetValue,
   int nChannel
);
```

This function is to set the parameter of integration time.

nItem control which variable for integration time setting to change.
0 is for setting the Integration time units.
1 is for setting the integration time multiplier.
2 is for setting the integration time.

nSetValue is a set value for the 'nItem' parameter used.
When nItem = 0, use an 'nSetValue' of 0 for microseconds and 1 is for milliseconds.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

For example:

Value0=0

bwtekWriteValue(**0**, ***Value0***, channel); //Set the unit of integration time as microseconds (us).

Value0=1

bwtekWriteValue(**0**, ***Value0***, channel); //Set the unit of integration time as milliseconds (ms).

Value0=2

bwtekWriteValue(**1**, ***Value0***, channel); //Set the multiply of integration time as 2

Value0=34

bwtekWriteValue(**2**, ***Value0***, channel); //Set the integration time to 34.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekReadValue

```
int bwtekReadValue
(
   int nItem,
   int *nGetValue,
   int nChannel
);
```

This function is for read/write the adjust value of temperature of sample holder and CCD Detector.

nItem control which variable for integration time setting to read.
0 is for reading the Integration time units.
1 is for reading the integration time multiplier.
2 is for reading the integration time.

***nGetValue** is a pointer to return value of desired parameter of integration time.

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

For example:

bwtekReadValue (**0**, ***&Value0***, channel); // Value0 is the integration time unit. value0 = 0 = microseconds (us),
// if value0 = 1 = milliseconds (ms)

bwtekReadValue (**1**, ***&Value0***, channel); // Value0 is the multiplier value.

bwtekReadValue (**2**, ***&Value0***, channel); // the Value0 is the integration value.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

BTC262A &BTC263E Functions:

The following functions work with the BTC263E and BTC262A spectrometers:

bwtekSetTimeUnitUSB

```
int bwtekSetTimeUnitUSB
(
   int nTimeUnit,
   int nChannel
);
```

This function is for setting the integration time 'unit'.

nTimeUnit is a unit of integration time, 0 for microsecond (us), 1 for millisecond (ms).

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

Is the function call is successful the time nTimeUnit value will be returned else a negative integer will be returned.

bwtekGetTimeUnitUSB

```
int bwtekGetTimeUnitUSB
(
   int *TimeUnit,
   int nChannel
);
```

This function is for reading the integration time ‘unit.’

***nTimeUnit** is a pointer to an array that will return the unit value for the integration time, 0 for microsecond (us), 1 for millisecond (ms).

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

Is the function call is successful the time the time ‘unit’ will be returned else a negative integer will be returned.

bwtekSetInGaAsMode

```
int bwtekSetInGaAsMode
(
   int nSetModeValue,        // select detector mode
   int nChannel              // channel to get data from
);
```

This function is for setting the detector mode.

nSetModeValue is used to choose detector mode .

nSetModeValue	Detector Mode
0x00	Maximum Sensitivity (default)
0x80	High Sensitivity
0x180	High Dynamic Range
0x100	Maximum Dynamic Range

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekGetInGaAsMode

```
int bwtekGetInGaAsMode
(
   int *nMode,       // select detector mode
   int nChannel      // channel to get data from
);
```

This function is for retrieving the detector mode setting.
***Prerequisite Note:** The function call 'bwtekSetInGaAsMode' must be called first for this function to work.

***nMode** is a pointer to an array that will return the value of the feedback capacitor mode setting

nSetModeValue	Detector Mode
0x00	Maximum Sensitivity (default)
0x80	High Sensitivity
0x180	High Dynamic Range
0x100	Maximum Dynamic Range

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

BTC261P & BTC262P & BTC264P Functions:

The following functions work with the BTC261P & BTC262P & BTC264P spectrometers:

bwtekSetTimeUnitUSB (OBSOLETE)

```
int bwtekSetTimeUnitUSB
(
   int nTimeUnit,
   int nChannel
);
```

This function is for setting the integration time ‘unit’.

OBOLSETE NOTE
The integration time unit range for these model spectrometers are microseconds (us).
Range: 200us – 2,100,000,000us

bwtekGetTimeUnitUSB (OBSOLETE)

```
int bwtekGetTimeUnitUSB
(
   int *TimeUnit,
   int nChannel
);
```

This function is for reading the integration time ‘unit.’

OBOLSETE NOTE
The integration time unit for these model spectrometers will always be microseconds (us).
Range: 200us – 2,100,000,000us

bwtekSetInGaAsMode

```
int bwtekSetInGaAsMode
(
   int nSetModeValue,      // select detector mode
   int nChannel            // channel to get data from
);
```

This function is for setting the detector mode.

nSetModeValue is used to choose detector mode .

nSetModeValue	Detector Mode
0	High Sensitivity
1	High Dynamic Range

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN

If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

bwtekGetInGaAsMode

```
int bwtekGetInGaAsMode
(
   int *nMode,       // select detector mode
   int nChannel      // channel to get data from
);
```

This function is for retrieving the detector mode setting.
***Prerequisite Note:** The function call 'bwtekSetInGaAsMode' must be called first for this function to work.

***nMode** is a pointer to an array that will return the value of the feedback capacitor mode setting

*nSetModeValue	Detector Mode
0	High Sensitivity
1	High Dynamic Range

nChannel is used to address a specific spectrometer device to be operated when multiple spectrometer devices are involved. Users must call the *bwtekSetupChannel* function to determine which channel numbers are available. A total of 32 spectrometer devices may be connected at one time, where the nChannel value will range from 0 – 31.

RETURN
If the function call is successful, a positive integer will be returned, else a negative integer will be returned.

Appendix A: Model Descriptions

USB Model	Description	Digitizer Resolution (bits)
BRC111A	Non-Cooled 2048 CCD array spectrometer	16
BRC112P	Non-Cooled 2048 CCD array spectrometer	16
BRC112E	Non-Cooled 2048 CCD array spectrometer	16
BRC115E	Non-Cooled 2048 CCD array spectrometer	16
BRC115U	Non-Cooled 2048 CCD array spectrometer	16
BRC115V	Non-Cooled 2048 CCD array spectrometer	16
BRC642E-2048	Non-Cooled 2048 Back Thinned CCD	16
BRC711E-512	Non-Cooled 512 PDA array spectrometer	16
BRC711E-1024	Non-Cooled 1024 PDA array spectrometer	16
BRC741E-512	Non-Cooled 512 PDA array spectrometer	16
BRC741E-1024	Non-Cooled 1024 PDA array spectrometer	16
BTC111E	TE Cooled 2048 CCD array spectrometer	16
BTC112E	TE Cooled 2048 CCD array spectrometer	16
BTC162E	TE Cooled 2048 CCD array spectrometer	16
BTC261E-256	TE Cooled 256 InGaAs array spectrometer	16
BTC261E-512	TE Cooled 512 InGaAs array spectrometer	16
BTC261E-1024	TE Cooled 1024 InGaAs array spectrometer	16
BTC261P-512	TE Cooled 512 InGaAs array spectrometer	16
BTC262A-256	TE Cooled 256 InGaAs array spectrometer	16
BTC262E-256	TE Cooled 256 InGaAs array spectrometer	16
BTC262E-512	TE Cooled 512 InGaAs array spectrometer	16
BTC262P-512	TE Cooled 512 InGaAs array spectrometer	16
BTC263E-256	TE Cooled 256 InGaAs array spectrometer	16
BTC264P-256	TE Cooled 512 InGaAs array spectrometer	16
BTC264P-512	TE Cooled 512 InGaAs array spectrometer	16
BTC264P-1024	TE Cooled 512 InGaAs array spectrometer	16
BTC611E-512	TE Cooled 512 Back Thinned CCD	16
BTC611E-1024	TE Cooled 1024 Back Thinned CCD	16
BTC613E-512	TE Cooled 512 Back Thinned CCD	16
BTC613E-1024	TE Cooled 1024 Back Thinned CCD	16
BTC621E-512	TE Cooled 512 Back Thinned CCD	16
BTC621E-1024	TE Cooled 1024 Back Thinned CCD	16
BTC651E-512	TE Cooled 512 Back Thinned CCD	16
BTC651E-1024	TE Cooled 1024 Back Thinned CCD	16
BTC655E	TE Cooled 2048 Back Thinned CCD	16
BTC661E	TE Cooled 2048 Back Thinned CCD	16
BTC665E	TE Cooled 2048 Back Thinned CCD	16

Appendix B: USB Interface Capabilities

USB Models	USB 2.0 Connector: Uses FX2 USB Chip	USB 3.0 Connector: Uses FX3 USB Chip
BRC111A	Yes	
BRC112P	Yes	
BRC112E	Yes	
BRC115E		Yes
BRC115U		Yes
BRC115V		Yes
BRC642E-2048	Yes	
BRC711E-512	Yes	
BRC711E-1024	Yes	
BRC741E-512	Yes	
BRC741E-1024	Yes	
BTC111E	Yes	
BTC112E	Yes	
BTC162E	Yes	
BTC261E-256	Yes	
BTC261E-512	Yes	
BTC261E-1024	Yes	
BTC261P-512	Yes	
BTC262A-256	Yes	
BTC262E-256	Yes	
BTC262E-512	Yes	
BTC262P-512	Yes	
BTC263E-256	Yes	
BTC264P-256	Yes	
BTC264P-512	Yes	
BTC264P-1024	Yes	
BTC611E-512	Yes	
BTC611E-1024	Yes	
BTC613E-512	Yes	
BTC613E-1024	Yes	
BTC621E-512	Yes	
BTC621E-1024	Yes	
BTC651E-512	Yes	
BTC651E-1024	Yes	
BTC655E		Yes
BTC661E	Yes	
BTC665E		Yes

Appendix C: Spectrometer Driver Files

USB Models	VID (Hex)	PID (Hex)	SPT File	INF File	DLL File
BRC111A	16a3	2ec7	bwtekusb.spt	bwtekusb2.inf	bwtekusb.dll
BRC112P	16a3	2ec7	bwtekusb.spt	bwtekusb2.inf	bwtekusb.dll
BRC112E	16a3	2ec7	bwtekusb.spt	bwtekusb2.inf	bwtekusb.dll
BRC115E	16a3	2ed0	BRC115.spt	bwtekusb3.inf	bwtekusb.dll
BRC115U	16a3	2ed0	BRC115.spt	bwtekusb3.inf	bwtekusb.dll
BRC115V	16a3	2ed0	BRC115.spt	bwtekusb3.inf	bwtekusb.dll
BRC642E-2048	16a3	2ecd	bwteku14.spt	bwtekusb2.inf	bwtekusb.dll
BRC711E-512	16a3	2ec7	bwtekusb.spt	bwtekusb2.inf	bwtekusb.dll
BRC711E-1024	16a3	2ec7	bwtekusb.spt	bwtekusb2.inf	bwtekusb.dll
BRC741E-512	16a3	2ec7	bwtekusb.spt	bwtekusb2.inf	bwtekusb.dll
BRC741E-1024	16a3	2ec7	bwtekusb.spt	bwtekusb2.inf	bwtekusb.dll
BTC111E	16a3	2ec7	bwtekusb.spt	bwtekusb2.inf	bwtekusb.dll
BTC112E	16a3	2ec7	bwtekusb.spt	bwtekusb2.inf	bwtekusb.dll
BTC162E	16a3	2ec7	bwtekusb.spt	bwtekusb2.inf	bwtekusb.dll
BTC261E-256	16a3	2ec4	bwtekus6.spt	bwtekusb2.inf	bwtekusb.dll
BTC261E-512	16a3	2ec4	bwtekus6.spt	bwtekusb2.inf	bwtekusb.dll
BTC261E-1024	16a3	2ec4	bwtekus6.spt	bwtekusb2.inf	bwtekusb.dll
BTC261P-512	16a3	2ecf	bwteku16.spt	bwtekusb2.inf	bwtekusb.dll
BTC262A-256	16a3	2ece	bwteku15.spt	bwtekusb2.inf	bwtekusb.dll
BTC262E-256	16a3	2ec4	bwtekus6.spt	bwtekusb2.inf	bwtekusb.dll
BTC262E-512	16a3	2ec4	bwtekus6.spt	bwtekusb2.inf	bwtekusb.dll
BTC262P-512	16a3	2ecf	bwteku16.spt	bwtekusb2.inf	bwtekusb.dll
BTC263E-256	16a3	2ece	bwteku15.spt	bwtekusb2.inf	bwtekusb.dll
BTC264P-256	16a3	2ec3	bwteku16.spt	bwtekusb2.inf	bwtekusb.dll
BTC264P-512	16a3	2ec3	bwteku16.spt	bwtekusb2.inf	bwtekusb.dll
BTC264P-1024	16a3	2ec3	bwteku16.spt	bwtekusb2.inf	bwtekusb.dll
BTC611E-512	16a3	2ec2	bwtekus4.spt	bwtekusb2.inf	bwtekusb.dll
BTC611E-1024	16a3	2ec2	bwtekus4.spt	bwtekusb2.inf	bwtekusb.dll
BTC613E-512	16a3	2ec2	bwtekus4.spt	bwtekusb2.inf	bwtekusb.dll
BTC613E-1024	16a3	2ec2	bwtekus4.spt	bwtekusb2.inf	bwtekusb.dll
BTC621E-512	16a3	2ec2	bwtekus4.spt	bwtekusb2.inf	bwtekusb.dll
BTC621E-1024	16a3	2ec2	bwtekus4.spt	bwtekusb2.inf	bwtekusb.dll
BTC651E-512	16a3	2ec2	bwtekus4.spt	bwtekusb2.inf	bwtekusb.dll
BTC651E-1024	16a3	2ec2	bwtekus4.spt	bwtekusb2.inf	bwtekusb.dll
BTC655E	16a3	2ed3	BTC655.spt	bwtekusb3.inf	bwtekusb.dll
BTC661E	16a3	2ec2	bwtekus4.spt	bwtekusb2.inf	bwtekusb.dll
BTC665E	16a3	2ed1	BTC665.spt	bwtekusb3.inf	bwtekusb.dll

Appendix D: Spectrometer's USB Drivers File Destinations

Items	Target Location
INF File • bwtekusb2.inf • bwtekusb3.inf	C:\windows\inf\
CAT File • fx2lp.cat • cyusb3.cat	C:\windows\inf\
Spt File • bwtekusb.spt • bwtekusb2.spt • bwtekus4.spt • bwtekus6.spt • bwteku11.spt • bwteku14.spt • bwteku15.spt • bwteku16.spt • bwtekLC.spt • BRC115.spt • BTC655.spt • BTC665.spt • BTC675.spt	C:\windows\system32\drivers\bwtek\
SYS File • Cyusb.sys • Cyusb3.sys	C:\windows\system32\ C:\windows\system32\drivers\
DLL File • CyUSB.dll • WdfCoInstaller01009.dll	C:\windows\system32\ C:\windows\system32\drivers\
DLL File • BWTEKUSB.DLL	C:\windows\system32\ **+ The folder of your executable file.**

Appendix E: SDK Function Groups

USB Models	Supported Functions:
BRC111A	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • bwtekSetTTLOut
BRC112P	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • bwtekSetTTLOut
BRC112E	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • bwtekSetTTLOut
BRC115E	• USB Basic Functions • Supplementary Functions • AUX Port Functions 3: Shutter1...2
BRC115U	• USB Basic Functions • Supplementary Functions • AUX Port Functions 3: Shutter1...2
BRC115V	• USB Basic Functions • Supplementary Functions • AUX Port Functions 3: Shutter1...2
BRC642E-2048	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • bwtekSetTTLOut: Shutter1
BRC711E-512	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin
BRC711E-1024	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin
BRC741E-512	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin
BRC741E-1024	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin

USB Models	Supported Functions:
BTC111E	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin
BTC112E	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin
BTC162E	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin
BTC261E-256	• USB Basic Functions • Supplementary Functions • BTC261E & BTC262E Functions
BTC261E-512	• USB Basic Functions • Supplementary Functions • BTC261E & BTC262E Functions
BTC261E-1024	• USB Basic Functions • Supplementary Functions • BTC261E & BTC262E Functions
BTC261P-512	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • bwtekSetTTLOut(): Shutter1, Shutter2 • BTC261P & BTC262P & BTC264P Functions
BTC262A-256	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • bwtekSetTTLOut(): Shutter1, Shutter2 • BTC263E & BTC262A Functions
BTC262E-256	• USB Basic Functions • Supplementary Functions • BTC261E & BTC262E Functions
BTC262E-512	• USB Basic Functions • Supplementary Functions • BTC261E & BTC262E Functions
BTC262P-512	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • bwtekSetTTLOut(): Shutter1, Shutter2 • BTC261P & BTC262P & BTC264P Functions

USB Models	Supported Functions:
BTC263E-256	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • bwtekSetTTLOut(): Shutter1, Shutter2 • BTC263E & BTC262A Functions
BTC264P-256	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • bwtekSetTTLOut(): Shutter1, Shutter2 • BTC261P & BTC262P & BTC264P Functions
BTC264P-512	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • bwtekSetTTLOut(): Shutter1, Shutter2 • BTC261P & BTC262P & BTC264P Functions
BTC264P-1024	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • bwtekSetTTLOut(): Shutter1, Shutter2 • BTC261P & BTC262P & BTC264P Functions
BTC611E-512	• USB Basic Functions • Supplementary Functions
BTC611E-1024	• USB Basic Functions • Supplementary Functions
BTC613E-512	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • AUX Port Functions 2: Shutter Open/Close
BTC613E-1024	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • AUX Port Functions 2: Shutter Open/Close
BTC621E-512	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • AUX Port Functions 2: Shutter Open/Close
BTC621E-1024	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • AUX Port Functions 2: Shutter Open/Close

USB Models	Supported Functions:
BTC651E-512	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • AUX Port Functions 2: Shutter Open/Close
BTC651E-1024	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • AUX Port Functions 2: Shutter Open/Close
BTC655E	• USB Basic Functions • Supplementary Functions • AUX Port Functions 3: Shutter1...5
BTC661E	• USB Basic Functions • Supplementary Functions • AUX Port Functions 1: Multi-Purpose Pin • AUX Port Functions 2: Shutter Open/Close
BTC665E	• USB Basic Functions • Supplementary Functions • AUX Port Functions 3: Shutter1...5

Appendix F: Default Parameters for bwtekTESTUSB

USB Models	Timing Mode	Pixel # nPixelNo	Input Mode nInputMode	Integration Time Offset	Int.Time Range nTimingMode	Int. time unit settings	Array Location Dummy Pixels
BRC111A	1	2048	1	NA	9 – 65535ms	milliseconds (ms)	13 - 30
BRC112P	1	2048	1	NA	1 – 65535ms	milliseconds (ms)	13 - 30
BRC112E	1	2048	1	NA	1 – 65535ms	milliseconds (ms)	13 - 30
BRC115E	1	2048	1	NA	1,000us – 2,100,000,000us	**microseconds (us)**	13 - 30
BRC115U	1	2048	1	NA	1,000us – 2,100,000,000us	**microseconds (us)**	13 - 30
BRC115V	1	2048	1	NA	1,000us – 2,100,000,000us	**microseconds (us)**	13 - 30
BRC642E-2048	10	2048	2	6ms	7 - 65535 ms	milliseconds (ms)	4 - 6
BRC711E-512	3	512	2	NA	2 – 65535ms	milliseconds (ms)	NA
BRC711E-1024	3	1024	2	NA	3 – 65535ms	milliseconds (ms)	NA
BRC741E-512	3	512	2	NA	2 – 65535ms	milliseconds (ms)	NA
BRC741E-1024	3	1024	2	NA	3 – 65535ms	milliseconds (ms)	NA
BTC111E	1	2048	1	NA	9 – 65535ms	milliseconds (ms)	13 - 30
BTC112E	1	2048	1	NA	5 – 65535ms	milliseconds (ms)	13 - 30
BTC162E	1	2048	1	NA	5 – 65535ms	milliseconds (ms)	13 - 30
BTC261E-256	3	256	14	NA	0.010 – 65535ms	milliseconds (ms)	NA
BTC261E-512	3	512	13	NA	0.010 – 65535ms	milliseconds (ms)	NA
BTC261E-1024	3	1024	13	NA	0.010 – 65535ms	milliseconds (ms)	NA
BTC261P-512	1	512	2	NA	200us – 2,100,000,000us	**microseconds (us)**	NA
BTC262A-256	1	256	2	80us	0.250 – 65535ms	*microseconds (us)* *milliseconds (ms)*	NA
BTC262E-256	3	256	14	NA	0.010 - 65535ms	milliseconds (ms)	NA
BTC262E-512	3	512	13	NA	0.010 – 65535ms	milliseconds (ms)	NA
BTC262P-512	1	512	2	NA	200us – 2,100,000,000us	**microseconds (us)**	NA
BTC263E-256	1	256	2	80us	0.250 – 65535ms	*microseconds (us)* *milliseconds (ms)*	NA
BTC264P-256	1	256	2	NA	200us – 2,100,000,000us	**microseconds (us)**	NA
BTC264P-512	1	512	2	NA	200us – 2,100,000,000us	**microseconds (us)**	NA
BTC264P-1024	1	1024	2	NA	200us – 2,100,000,000us	**microseconds (us)**	NA
BTC611E-512	6	512	2	26ms	27 – 65535ms	milliseconds (ms)	4 - 9
BTC611E-1024	6	1024	2	49ms	50 – 65535ms	milliseconds (ms)	4 - 9

USB Models	Timing Mode	Pixel # nPixelNo	Input Mode nInputMode	Integration Time Offset	Int.Time Range nTimingMode	Int. time unit settings	Array Location Dummy Pixels
BTC613E-512	6	512	2	26ms	27 – 65535ms	milliseconds (ms)	4 - 9
BTC613E-1024	6	1024	2	49ms	50 – 65535ms	milliseconds (ms)	4 - 9
BTC621E-512	6	512	2	26ms	27 – 65535ms	milliseconds (ms)	4 - 9
BTC621E-1024	6	1024	2	49ms	50 – 65535ms	milliseconds (ms)	4 - 9
BTC651E-512	6	512	2	26ms	27 – 65535ms	milliseconds (ms)	4 - 9
BTC651E-1024	6	1024	2	49ms	50 – 65535ms	milliseconds (ms)	4 - 9
BTC655E	1	2048	2	5601us	6000us – 2,100,000,000us	**microseconds (us)**	4 - 9
BTC661E	6	2048	2	20ms	21 – 65535ms	milliseconds (ms)	4 - 9
BTC665E	1	2048	2	5601us	6000us – 2,100,000,000us	**microseconds (us)**	4 - 9

Appendix G: External IO Control

USB Model	External Trigger	Multi-Purpose TTL Output	Shutter Control 1 Output	Shutter Control 2 Output	TTL Sensing Input	Analog Input
BRC111A	Yes	Yes	Yes	Yes	Yes	No
BRC112P	Yes	Yes	Yes	Yes	Yes	No
BRC112E	Yes	Yes	Yes	Yes	Yes	No
BRC115E	Yes	Yes	Yes	Yes	Yes	Yes
BRC115U	Yes	Yes	Yes	Yes	Yes	Yes
BRC115V	Yes	Yes	Yes	Yes	Yes	Yes
BRC642E-2048	Yes	Yes	Yes	No	No	No
BRC711E-512	Yes	Yes	No	No	No	No
BRC711E-1024	Yes	Yes	No	No	No	No
BRC741E-512	Yes	Yes	No	No	No	No
BRC741E-1024	Yes	Yes	No	No	No	No
BTC111E	Yes	Yes	No	No	Yes	No
BTC112E	Yes	Yes	No	No	Yes	No
BTC162E	Yes	Yes	No	No	Yes	No
BTC261E-256	Yes	No	No	No	No	No
BTC261E-512	Yes	No	No	No	No	No
BTC261E-1024	Yes	No	No	No	No	No
BTC261P-512	Yes	Yes	Yes	Yes	Yes	No
BTC262A-256	Yes	No	Yes	Yes	Yes	Yes
BTC262E-256	Yes	No	No	No	No	No
BTC262E-512	Yes	No	No	No	No	No
BTC262P-512	Yes	Yes	Yes	Yes	Yes	No
BTC263E-256	Yes	No	Yes	Yes	Yes	Yes
BTC264P-256	Yes	Yes	Yes	Yes	Yes	No
BTC264P-512	Yes	Yes	Yes	Yes	Yes	No
BTC264P-1024	Yes	Yes	Yes	Yes	Yes	No
BTC611E-512	No	No	No	No	No	No
BTC611E-1024	No	No	No	No	No	No
BTC613E-512	Yes	Yes (♦)	Yes (♦)	Yes (♦)	No	No
BTC613E-1024	Yes	Yes (♦)	Yes (♦)	Yes (♦)	No	No
BTC621E-512	Yes	Yes (♦)	Yes (♦)	Yes (♦)	No	No
BTC621E-1024	Yes	Yes (♦)	Yes (♦)	Yes (♦)	No	No
BTC651E-512	Yes	Yes (♦)	Yes (♦)	Yes (●)	No	No
BTC651E-1024	Yes	Yes (♦)	Yes (♦)	Yes (●)	No	No
BTC655E	Yes	Yes	Yes	Yes	Yes	Yes
BTC661E	Yes	Yes (♦)	Yes (♦)	Yes (●)	No	No
BTC665E	Yes	Yes	Yes	Yes	Yes	Yes

(♦) “Multi-Purpose TTL Output” and “Shutter Control 1 Output” is same pin.

(●) “Shutter Control 2 Output” be used by the inside shutter of spectrometer.

Appendix H: X-axis Reverse Settings

Some spectrometer may have their x-axis reversed for production purposes.
These models will first have their pixel values reversed.
For example if the physical location on the detector's pixel array is Pixel 2047, the software will flip the detector array so pixel 2047 will be Pixel 0.

Our End User, BWSpec software, automatically accounts for this reversal.
In the para.ini (EEPROM) there is a line called "xaxis_data_reverse." When the value of xaxis_data_reverse=1" the unit will need its x-axis to be reversed. When xaxis_data_reverse=0, no additional steps need to be performed.

USB Models	X-axis Reversed	Comment
BRC111A	No	
BRC112P	**Yes**	**xaxis_data_reverse=1**
BRC112E	BRC112E: No, BRC112U: No, **BRC112V: Yes**	**xaxis_data_reverse=1**
BRC115E	No	
BRC115U	No	
BRC115V	No	
BRC642E-2048	**Yes**	**xaxis_data_reverse=1**
BRC711E-512	No	
BRC711E-1024	No	
BRC741E-512	**Yes**	**xaxis_data_reverse=1**
BRC741E-1024	**Yes**	**xaxis_data_reverse=1**
BTC111E	No	
BTC112E	No	
BTC162E	**Yes**	**xaxis_data_reverse=1**
BTC261E-256	**Yes**	**xaxis_data_reverse=1**
BTC261E-512	**Yes**	**xaxis_data_reverse=1**
BTC261E-1024	**Yes**	**xaxis_data_reverse=1**
BTC261P-512	**Yes**	**xaxis_data_reverse=1**
BTC262A-256	**Yes : For '+1 Order' configured spectrometers**	**xaxis_data_reverse=1**
BTC262E-256	**Yes**	**xaxis_data_reverse=1**
BTC262E-512	**Yes**	**xaxis_data_reverse=1**
BTC262P-512	**Yes**	**xaxis_data_reverse=1**
BTC263E-256	**Yes : For '+1 Order' configured spectrometers**	**xaxis_data_reverse=1**
BTC264P-256	**Yes**	**xaxis_data_reverse=1**
BTC264P-512	**Yes**	**xaxis_data_reverse=1**
BTC264P-1024	**Yes**	**xaxis_data_reverse=1**
BTC611E-512	No	
BTC611E-1024	No	

USB Models	X-axis Reversed	Comment
BTC613E-512	No	
BTC613E-1024	No	
BTC621E-512	**Yes**	**xaxis_data_reverse=1**
BTC621E-1024	**Yes**	**xaxis_data_reverse=1**
BTC651E-512	No	
BTC651E-1024	No	
BTC655E	No	
BTC661E	No	
BTC665E	**Yes**	**xaxis_data_reverse=1**

Appendix I: Dynamically Assigned Channel Numbers

Procedure:

(1) Call function **InitDevices()** → Initialize the USB devices

(2) Call function **bwtekSetupChannel()** → get channel array →channel[]

(3) Call function **GetDeviceCount()** → get connected USB spectrometers count

(4) Loop read all spectrometer's channel, c.code and EEPROM.

Sample Code:

The below sample code is an excerpt from the C#_2.1 sample code from Demo 2.1 located in the SDK-S v1.0.0.1 sample code folder.

Function calls from the SDK User Manual used below are in **Red Font** below.

```
public struct Spec_Para_Struct

{
   public int usbtype;              //2=USB2.0, 3=USB3.0
   public int channel;
   public string cCode;
   public string model;
   public string spectrometer_name;
   public int spectrometer_type;
   public int pixel_number;
   public int timing_mode;
   public int input_mode;
   public int xaxis_data_reverse;

   public double inttime;
   public int inttime_int;
   public double inttime_min;
   public int inttime_base;
   public int inttime_unit;

   public double coefficient_a0;
   public double coefficient_a1;
   public double coefficient_a2;
   public double coefficient_a3;
   public double coefficient_b0;
   public double coefficient_b1;
   public double coefficient_b2;
   public double coefficient_b3;
}
```



```
public static Spec_Para_Struct[] spec_para = new Spec_Para_Struct[32];

bool retcode = InitDevices();  //Initialize USB Device
if (retcode)
{
   byte[] channel = new byte[32];
   int retcode1 = bwtekSetupChannel(-1, channel); //Get all available channels to array
   if (retcode1 > 0)
   {
      device_count = GetDeviceCount();
      for (int i = 0; i < device_count; i++)   //Loop get channel, ccode and read eeprom
      {
         if (channel[i] < 32)
         {
            spec_para[i].channel = channel[i]; //Get channel
            byte[] tmp_pccode = new byte[8];
            int ret = GetCCode(tmp_pccode, i);
            spec_para[i].cCode = ASCIIEncoding.ASCII.GetString(tmp_pccode); //Get CCode

            int tmp_usbtype = 0;
            ret = GetUSBType(ref tmp_usbtype, i);
            spec_para[i].usbtype = tmp_usbtype;    //Get USB type

            string filename =
System.IO.Path.GetDirectoryName(System.Windows.Forms.Application.ExecutablePath) +
"\\para.ini";
            ret = bwtekReadEEPROMUSB(filename, spec_para[i].channel);  //Read EEPROM to file para.ini
            Load_Para(filename, i);       //Read spectrometer's parameter from file.
         }
      }
   }
}
```

