USB Hardware Interface Software Development Kit (SDK)

Prepared by:

Beta Innovations Inc.
1071 Ambleside Drive
Suite 104, Access Code 0034
Ottawa, Ontario
K2B 6V4
Canada

APPROVALS

Originator	Leo Lacava		Nov. 16, 2006.
Originator			
Director			
	Name	Signature	Date

REV. NO.	APPROVAL DATE	PAGES MODIFIED	DESCRIPTION
00	Nov. 16, 2006		Initial write-up.
01	May 30, 2007		Added new function calls.
02	Jan 5, 2009		Added new function calls.

Table of Contents

BIUSB.DLL Module

Filename: biusb.dll

Description: USB HID device interface module.

BIUSB.DLL is a HID class (Human Interface Device) dynamic link library module that can be used to interface with all custom BI USB devices (i.e. Plasma, GammaRay, etc.).

This dynamic link library module can be late loaded at runtime or linked at compile time using the "biusb.lib" included with this SDK package. The "biusb.h" contains all the necessary function declarations and constants for programs written in C/C++/MFC.

"ApiDeclarations.bas" has been supplied as part of this SDK package for Visual Basic programs.

C/C++ Exported Functions

The following exported functions are part of this DLL module.

DetectHID

```
BOOL DetectHID (
        UINT *oDev_Cnt,
        DeviceParam *oDeviceList,
        UINT iFlag
) ;
```

The **DetectHID** function detects all compatible vendor specific USB HID devices. This should be the first function called prior to all other calls or to refresh the device list.

Parameters

oDev_Cnt

[out] Return pointer to the total number of matching devices detected.

oDeviceList

[out] Pointer to a device parameter structure. All returned data is stored at this structure location.

iFlag

[in] This parameter specifies the device class to detect.

Value	Meaning
DT_FLASH	Reserved
DT_DEVICES	Reserved
DT_REPORT	Reserved
DT_HID	Detect vendor defined HID class modules. Will ignore all Joystick class devices such Plasma series of modules.
DT_ALL	Detect all HID class modules including Joystick class devices such as Plasma series modules.

Return Codes

TRUE	Success detecting devices.
FALSE	Failure detecting devices or no devices found.

DetectDevice

```
BOOL DetectDevice (
            UINT *oDev_Cnt,
            DeviceParam *oDeviceList,
            UINT iProductID,
            UINT iFlag
) ;
```

The **DetectDevice** function detects only compatible vendor specific USB HID devices matching the specified product ID. This should be the first function called prior to all other calls or to refresh the device list.

Parameters

oDev_Cnt

[out] Return pointer to the total number of matching devices detected.

oDeviceList

[out] Pointer to a device parameter structure. All returned data is stored at this structure location.

iProductID

[in] This parameter specifies the product type to detect.

HID Class Devices	**Name**
PID_0X64B	0x64B input module
PID_GAMMARAY	GammaRay-256 input module
PID_GAMMARAY64	GammaRay-64 input module
PID_ELECTRONFLUX	ElectronFlux input/output module
PID_GAMMARAY_V2	GammaRay V2 input module
PID_GAMMARAY_V3	GammaRay V3 input module
PID_GAMMATRON	GammaTron input module
PID_NITRO_SLG	Nitro-SLG output module
Joystick Class Devices	**Name**
PID_4X24BH	4x24BH joystick module
PID_6X16B	6x16B joystick module
PID_6X13B	6x13B joystick module
PID_5X18BH	5x18BH joystick module
PID_PLASMA	Plasma V1 joystick module
PID_PLASMA_HOTAS	Plasma V1 in HOTAS mode joystick module
PID_PLASMA_DUAL	Plasma V1 in DUAL mode joystick module
PID_PLASMA_HOTAS_DUAL	Plasma V1 in HOTAS DUAL mode joystick module
PID_PLASMA_LITE	Plasma-Lite joystick module
PID_PLASMA_LITE_V2	Plasma-Lite V2 joystick module

PID_PLASMA_MM2	Plasma-MM2 joystick module

iFlag

[in] This parameter specifies the device class to detect.

Value	Meaning
DT_FLASH	Reserved
DT_DEVICES	Detect HID class modules that accept input commands or data from host. These are typically output class devices.
DT_REPORT	Reserved
DT_HID	Detect HID input class modules. Will ignore all Joystick type devices such Plasma class modules.
DT_ALL	Detect all HID input class modules including Joystick type devices such as Plasma modules.

Return Codes

TRUE	Success detecting devices.
FALSE	Failure detecting devices or no devices found.

RetrieveStatus

```
void RetrieveStatus (
        DeviceParam *iDeviceList,
        DeviceStatus *oDeviceStatus
) ;
```

The **RetrieveStatus** function returns the configuration status of all IO ports found on the module. Can be used to verify if ports are active before reading or writing to them. This function is typically called after **DetectHID**.

Parameters

iDeviceList

[in] Pointer to a device parameter structure. Must contain valid device information returned from call to **DetectHID.**

oDeviceStatus

[out] Pointer to a device status array. All returned data will be stored at this location.

Return Codes

None	

CloseDevices

```
BOOL CloseDevices (
          UINT iDev_Cnt,
          DeviceParam *iDeviceList
) ;
```

The **CloseDevices** function releases all detected modules, cancels all pending threads and frees all used memory blocks. This call should be made on program exit and prior to any calls to the **DetectHID** function in order to refresh the device list.

Parameters

iDev_Cnt

[in] Number of detected modules to close in the *iDeviceList*. This value is retuned by call to **DetectHID.**

iDeviceList

[in] Pointer to a device parameter structure. Must contain valid device information returned from call to **DetectHID.**

Return Codes

TRUE	Success closing devices.
FALSE	Failure closing devices or releasing memory.

ReadInputData

```
LONG ReadInputData (
        DeviceParam *iDeviceList,
        char *oDataBuffer,
        UINT iFlag
) ;
```

The **ReadInputData** function is a non-blocking call, which returns input data. The first call to this function initiates a request for input data from the specified module as indicated by the *DeviceParam* structure and immediately returns from the function. This prevents the main calling application from locking while the module is being polled. The return flag indicates the current status of the request.

A return status of *DEV_INPUT* indicates new data is available at the location pointed to by *oDataBuffer* for the module specified in the *DeviceParam* structure. The *NumberInputIndices* within the *DeviceParam* structure indicates the number of input values to be read and size of the required array.

If no new data is available, the previously buffered data will be returned in addition to the *DEV_WAIT* flag indicating the module has not yet responded to the request for input data.

Depending on the specific module, polling frequency may vary in the range from 10ms to 40ms or more before data is flagged as being available. As a general rule, this function should be polled every 10 ms if real-time input data is required.

If a device should become unplugged or is no longer responding, the *DEV_TIMEOUT* flag will be returned after 1 second has elapsed.

Parameters

iDeviceList

[in] Pointer to a device parameter structure. Must contain valid device information returned from call to **DetectHID.**

oDataBuffer

[out] Pointer to the returned data buffer array. *NumberInputIndices* within the *DeviceParam* structure indicates the required length of this data buffer array specifying the number of detected inputs for the module.

iFlag

[in] Must always be set to FALSE.

Return Codes

DEV_TIMEOUT	Device did not respond within 1 second.
DEV_FAILED	Failure reading from device.
DEV_WAIT	No new data available. Waiting for device response.
DEV_INPUT	New data available.

ClearLCD

```
LONG ClearLCD (
        DeviceParam *iDeviceList,
        UCHAR inLCD
) ;
```

The **ClearLCD** function sends a command to clear contents of the LCD screen. This function can be used on modules supporting character based LCDs.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inLCD

[in] Zero based index of LCD to be cleared. Currently not support. Leave as 0.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteLCD

```
LONG WriteLCD (
        DeviceParam *iDeviceList,
        UCHAR inLCD,
        UCHAR inLine,
        char *iDataBuffer,
        UCHAR inLevel
);
```

The **WriteLCD** function writes contents of *iDataBuffer* to the LCD specified by *inLCD* at the line number indicated by *inLine*. This function can be used on modules supporting character based LCDs.

Parameters

iDeviceList
[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inLCD
[in] Zero based index of LCD to write to. Currently not support. Leave as 0.

inLine
[in] Line number to write buffer to. Values range from 1 to 4.

iDataBuffer
[in] Pointer to character array. Must be at least 20 characters in length. Empty spaces on the LCD should be filled with the ASCII value of 0x20 (blank space). Do not NULL terminate this buffer. NULL value will display character 0x00 stored in the LCD ROM.

inLevel
[in] LCD backlight brightness level if supported. Values range from 0 (min brightness) to 255 (maximum brightness).

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteDirectOutput

```
LONG WriteDirectOutput (
        DeviceParam *iDeviceList,
        UCHAR inPort,
        DIRECT_OUTPUT *iDataBuffer
) ;
```

The **WriteDirectOutput** function writes contents of *iDataBuffer* to the output pins on the port specified by *inPort*. This function can be used on modules supporting direct outputs.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to data array defined by DIRECT_OUTPUT structure.

```
typedef struct _STRUCT_DIRECT_OUTPUT {
        unsigned char state:1;
        unsigned char level:7;
} DIRECT_OUTPUT ;
```

Bit Field	Description
state:1;	Output state value 0 for OFF and 1 for ON.
level:7;	Not currently supported. Leave as 0.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteMuxOutput

```
LONG WriteMuxOutput (
          DeviceParam *iDeviceList,
          UCHAR inPort,
          DIRECT_OUTPUT *iDataBuffer
) ;
```

The **WriteMuxOutput** function writes contents of *iDataBuffer* to the multiplexed output pins on the port specified by *inPort*. This function can be used on modules supporting multiplexed outputs.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to data array defined by DIRECT_OUTPUT structure.

```
typedef struct _STRUCT_DIRECT_OUTPUT {
          unsigned char state:1;
          unsigned char level:7;
} DIRECT_OUTPUT ;
```

Bit Field	**Description**
state:1;	Output state value 0 for OFF and 1 for ON.
level:7;	Output level range from 0 to 10. This value sets the pulse width for controlling the intensity of the multiplexed output.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteDisplayOutput

```
LONG WriteDisplayOutput (
          DeviceParam *iDeviceList,
          UCHAR inPort,
          char *iDataBuffer,
          UCHAR inLevel
);
```

The **WriteDisplayOutput** function writes contents of *iDataBuffer* to the 7-segment display on the port specified by *inPort*. This function can be used on modules supporting 7 x 7-segment displays.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to character array. Maximum number of display characters is 7, however, decimals do not count as character spaces. Supported display characters are limited to ASCII characters “**0**” to “**9**”, “**.**”, “**-**“ and blank space (NULL).

inLevel

[in] Output level range from 0 to 10. This value sets the pulse width of the multiplexed display output controlling intensity.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

Write7Segment

```
LONG Write7Segment (
        DeviceParam *iDeviceList,
        UCHAR *iData
);
```

The **Write7Segment** function writes contents of *iData* to a single 7-segment display on the device specified by *iDeviceList*. This function can be used on modules supporting a 7-segment display and a single indicator output typically used for a shift light.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

iData

[in] Pointer to a 2 Byte character array. The first Byte value contains the character to be displayed on a 7-segment display. Supported values are 0x00 to 0x09 in hex which will display the corresponding ASCII characters “**0**” to “**9**” when using industry standard decoders. Non-supported values will blank the display. Note: do not use the ASCII value to be displayed. Example, to display the value “1”, set *iData* to the hex value 0x01, not 0x31 corresponding to the ASCII value of “1”.

The second byte value is used for a shift light indicator with 0 for OFF and 1 for ON.

Value	Displayed symbol
0	0 or N if supported
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	-
11	L
12	P
13	=
14	Blank
15	R

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteSPIDAC

```
LONG WriteSPIDAC (
        DeviceParam *iDeviceList,
        UCHAR inPort,
        DAC_OUTPUTS *iDataBuffer,
        UCHAR inLevel
) ;
```

The **WriteSPIDAC** function writes contents of *iDataBuffer* to SPI compatible peripherals or DACs on the port specified by *inPort*. This function can be used on modules supporting either DACs or SPI ports.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to data array defined by DAC_OUTPUTS structure.

```
typedef struct _STRUCT_DAC_OUTPUTS {
        unsigned long DAC_1;
        unsigned long DAC_2;
        unsigned long DAC_3;
        unsigned long DAC_4;
        unsigned long DAC_5;
        unsigned long DAC_6;
        unsigned long DAC_7;
        unsigned long DAC_8;
        unsigned long DAC_9;
        unsigned long DAC_10;
} DAC_OUTPUTS ;
```

A port supporting 10 DAC or SPI outputs will require a 40 Byte buffer, each 32 bits long word corresponding to the data to be written to the corresponding DAC/SPI output. Note that all SPI/DACs on a port are written to at the same time. Writing to a single SPI/DAC is not possible; therefore SPI/DAC buffer data must be updated in *iDataBuffer* for all SPI/DACs on the specified port.

Actual data length output on a corresponding SPI/DAC channel is determined by the module's capability, not the maximum word length of 32 bits per channel as defined in the structure above.

inLevel

[in] Output level range from 0 to 7. This value sets the pulse width of the backlight if supported.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

CmdGLCD

```
LONG CmdGLCD (
            DeviceParam *iDeviceList,
            UCHAR inPort,
            UCHAR iBGColor,
            UCHAR iCmd
) ;
```

The **CmdGLCD** function sends the specified command in *iCmd* to a graphic LCD on the port specified by *inPort*. This function can be used on modules supporting graphic based LCDs.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iBGColor

[in] Set the background color of the LCD display. 0 = Black text on white background, 1 = White text on black background.

iCmd

[in] GLCD Commands:

```
#define GLCD_CLEAR          0x00
#define GLCD_TEST           0x01
#define GLCD_BGCOLOR        0x03
```

Define	Description
GLCD_CLEAR	Clear contents of LCD.
GLCD_TEST	Display test message.
GLCD_BGCOLOR	Set background color of LCD specified by *iGBColor.*

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteGLCD

```
LONG WriteGLCD (
        DeviceParam *iDeviceList,
        UCHAR inPort,
        UCHAR inLine,
        UCHAR iNumLines,
        UCHAR inBGColor,
        char *iDataBuffer
);
```

The **WriteGLCD** function writes contents of *iDataBuffer* to the graphic LCD located on the port specified by *inPort* at the line number indicated by *inLine*. This function can be used on modules supporting graphic based LCDs.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

inLine

[in] Starting line number to write contents of *iDataBuffer* to LCD. Values range from 1 to 5.

iNumLines

[in] Number of lines contained in *iDataBuffer*. Maximum buffer length is 2 lines at 24 characters per line (48 bytes).

iBGColor

[in] Set the background color of the LCD display. 0 = Black text on white background, 1 = White text on black background.

iDataBuffer

[in] Pointer to character array. Empty spaces on the LCD should be filled with the ASCII value of 0x20 (blank space). Do not NULL terminate this buffer.

Special Characters.

ASCII	Description
0x01	⇕ Arrow glyph.
0x02	✱ Star glyph.
0x03	° Degree glyph.
0x04	✱ Star glyph.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteLatchedOutput

```
LONG WriteLatchedOutput (
            DeviceParam *iDeviceList,
            UCHAR inPort,
            UCHAR *iDataBuffer,
            UCHAR inLevel
) ;
```

The **WriteLatchedOutput** function writes contents of *iDataBuffer* to the port specified by *inPort* configured for latched output mode.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to a 64 byte array (64 outputs). Output state value 0 for OFF and 1 for ON.

inLevel

[in] Output level range from 0 to 10. This value sets the pulse width of the output, controlling the intensity if supported.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteCRTRWR

```
LONG WriteCRTRWR (
          DeviceParam *iDeviceList,
          UCHAR inPort,
          UCHAR inContacts,
          UCHAR inSymbolCnt,
          UCHAR inContactIndex,
          CRTRWR_OUTPUT *iRWRBuffer
) ;
```

The **WriteCRTRWR** function writes contents of *iRWRBuffer* to the port specified by *inPort* configured for CRTRWR mode.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

inContacts

[in] Total number of contacts to display on screen. This value is typically same as *inSymbolCnt*.

inSymbolCnt

[in] Number of symbols stored in *iRWRBuffer.*

inContactIndex

[in] index within *iRWRBuffer* of the first symbol.

iRWRBuffer

[in] Pointer to data array defined by CRTRWR_OUTPUT structure.

```
typedef struct _STRUCT_CRTRWR_OUTPUT {
          unsigned symbol:8;
          unsigned xPos:8;
          unsigned yPos:8;
          unsigned missileActivity:1;
          unsigned missileLaunch:1;
          unsigned newguy:1;
          unsigned selected:1;
          unsigned :4;
} CRTRWR_OUTPUT ;
```

Bit Field	**Description**
symbol:8;	Symbol ID to display. (refer to RWR Symbol Class table Entries found in rwr.h)

xPos:8;	X position on screen to display symbol. Range 0 (left) to 255 (right).
yPos:8;	Y position on screen to display symbol. Range 0 (top) to 255 (bottom).
missileActivity:1;	If set to 1, display missile activity symbol.
missileLaunch:1;	If set to 1, display missile launch symbol.
newguy:1;	If set to 1, display new contact symbol.
selected:1;	If set to 1, draw contact selected symbol.
:4;	Spare unused.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteDotMatrix

```
LONG WriteDotMatrix (
        DeviceParam *iDeviceList,
        UCHAR inPort,
        char *iDataBuffer,
        UCHAR inLevel
) ;
```

The **WriteDotMatrix** function writes contents of *iDataBuffer* to 4 OSRAM type dot matrix displays on the port specified by *inPort*.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to a 16 byte character array. Maximum number of display characters is 4 per display (4 displays max per port). Supported display characters are limited to ASCII values (0 – 255). Refer to manufacturer's ACSII table for symbol defines.

inLevel

[in] Output level range from 0 to 10. This value sets the pulse width of the dot matrix display controlling intensity.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

C/C++ Structures & Defines

The following structures and defines are part of this DLL module.

DeviceParam Structure

```
typedef struct {
            HANDLE          DeviceHandle;
            USHORT          NumberInputIndices;
            USHORT          NumberOutputIndices;
            USHORT          NumberPortIndices;
            USHORT          NumberAnalogIndices;
            USHORT          NumberHATIndices;
            USHORT          NumberRotaryIndices;
            USHORT          NumberCRTRWRIndices;
            USHORT          NumberPWMIndices;
            USHORT          NumberLCDIndices;
            USHORT          NumberGLCDIndices;
            USHORT          NumberMuxDisplayIndices;
            USHORT          NumberAlphanumericIndices;
            USHORT          NumberDACIndices;
            USHORT          NumberSPIIndices;
            USHORT          NumberLatchedIndices;
            USHORT          NumberDotMatrixIndices;
            USHORT          VendorID;
            USHORT          ProductID;
            USHORT          VersionNumber;
            USHORT          FlashVersion;
            char            ProgramFlag;
            char            DevicePath[MAX_CHAR];
            ULONG           PathLength;
            char            DeviceName[MAX_CHAR];
            ULONG           DeviceNameLength;
            char            ManufName[MAX_CHAR];
            ULONG           ManufNameLength;
            char            SerialNum[4];
            ULONG           SerialNumLength;
            char            ConfigFlag;
            USHORT          DevIndex;
            USHORT          InputReportByteLength;
            USHORT          OutputReportByteLength;
            USHORT          Usage;
            USHORT          UsagePage;
} DeviceParam, *pDeviceParam ;
```

Device specific data structure.
Defined in: BIUSB.H

Parameters

DeviceHandle: The read/write handle for the device. This is a unique value and changes with each call to DetectHID.

NumberInputIndices: The number of unpacked inputs that will be returned from this device.

NumberOutputIndices: The number of unpacked outputs that will be read by this device.

NumberPortIndices: The number of separate IO ports on a device. These can also refer to JPs found on older devices.

NumberAnalogIndices: Not currently supported.

NumberHATIndices: Not currently supported.

NumberRotaryIndices: Not currently supported.

NumberCRTRWRIndices: Number of RWR supported.

NumberPWMIndices: Not currently supported.

NumberLCDIndices: Number of character based LCD's supported.

NumberGLCDIndices: Number of graphic based LCD's supported.

NumberMuxDisplayIndices: Number of multiplexed displays supported.

NumberAlphanumericIndices: Not currently supported.

NumberDACIndices: Number of DAC (Digital to Analog Converter) supported.

NumberSPIIndices: Number of SPI interface devices supported.

NumberLatchedIndices: Number of latched outputs supported.

NumberDotMatrixIndices: Number of OSRAM type dot matrix displays supported.

VendorID: USB.org assigned unique vendor ID number.

ProductID: Vendor assigned unique device ID number.

VersionNumber: Vendor assigned product version number.

FlashVersion: Current Firmware revision of device.

ProgramFlag: For internal use only.

DevicePath: Pointer to the device path key string as found in the Windows registry.

PathLength: Length in bytes of the *DevicePath* string.

DeviceName: Pointer to the device name string.

DeviceNameLength: Length in bytes of the *DeviceName* string.

ManufName: Pointer to the manufacturer name string.

ManufNameLength: Length in bytes of the ManufName string.

SerialNum: Pointer to the user assigned device serial number string.

SerialNumLength: Length in bytes of the *SerialNum* string.

ConfigFlag: For internal use only.

DevIndex: The zero based device index associated with this device.

InputReportByteLength: The number of packed bytes that will be returned by the device on each *ReadInputData* read request.

OutputReportByteLength: The number of packed bytes used by the device for output data.

Usage: For internal use only.

UsagePage: For internal use only.

DeviceStatus Structure

```
typedef struct {
        char      InputActive[MAX_INPUTS];
        char      OutputActive[MAX_OUTPUTS];
        char      PortActive[MAX_PORTS];
        char      PortIOMode[MAX_PORTS];
        char      PortMode[MAX_PORTS];
        char      OutputMode[MAX_PORTS];
        char      AnalogActive[MAX_ANALOG];
        char      HATActive[MAX_HATS];
        char      RotaryActive[MAX_ROTARY];
        char      CRTRWRActive[MAX_RWR];
        char      PWMActive[MAX_PWM];
        char      LCDActive[MAX_LCDS];
        char      GLCDActive[MAX_GLCDS];
        char      MuxDisplayActive[MAX_MUXDISPLAYS];
        char      AlphanumericActive[MAX_ALPHANUMERICS];
        char      DACActive[MAX_DACS];
        char      SPIActive[MAX_SPI];
        char      LatchedActive[MAX_LATCHED];
        char      DotMatrixActive[MAX_DOTMATRIX];
} DeviceStatus ;
```

Device specific data structure.
Defined in: BIUSB.H

Parameters

InputActive: Inputs is active if 1, disabled if 0.

OutputActive: Output is active if 1, disabled if 0.

PortActive: Port is active if 1, disabled if 0.

PortIOMode: Port is set to input mode if 1, output mode if 0.

PortMode: Currently only valid for ElectronFlux class devices. Return values are:

```
#define MODE_DIRECT        0
#define MODE_MUX           1
#define MODE_RWR           2
#define MODE_GLCD          3
#define MODE_DAC           4
#define MODE_LATCHED       5
#define MODE_SPI           6
#define MODE_DOTMATRIX     7
```

OutputMode: Not currently supported.

AnalogActive: Analog channels active if 1, disabled if 0.

HATActive: Not currently supported.

CRTRWRActive: CRT RWR channels active if 1, disabled if 0.

PWMActive: Not currently supported.

LCDActive: Not currently supported.

GLCDActive: Graphic LCD channels active if 1, disabled if 0.

MuxDisplayActive: Multiplexed display channels active if 1, disabled if 0.

AlphanumericActive: Not currently supported.

DACActive: DAC channels active if 1, disabled if 0.

SPIActive: SPI channels active if 1, disabled if 0.

LatchedActive: Latched channels active if 1, disabled if 0.

DotMatrixActive: DotMatrix channels active if 1, disabled if 0.

VB Exported Functions

The following exported functions are part of this DLL module.

DetectHID

```
Public Declare Function DetectHID Lib "biusb.dll" (
            ByRef oDev_Cnt As Integer,
            ByRef opDeviceList As DeviceParam,
            ByVal iFlag As Long
) As Boolean
```

The **DetectHID** function detects all compatible vendor specific USB HID devices. This should be the first function called prior to all other calls or to refresh the device list.

Parameters

oDev_Cnt
[out] Return pointer to the total number of matching devices detected.

oDeviceList
[out] Pointer to a device parameter structure. All returned data will be stored in this structure location.

iFlag
[in] This parameter specifies the device class to detect.

Value	Meaning
DT_FLASH	Reserved
DT_DEVICES	Reserved
DT_REPORT	Reserved
DT_HID	Detect vendor defined HID class modules. Will ignore all Joystick class devices such Plasma series of modules.
DT_ALL	Detect all HID class modules including Joystick class devices such as Plasma series modules.

Return Codes

TRUE	Success detecting devices.
FALSE	Failure detecting devices or no devices found.

DetectDevice

```
Public Declare Function DetectDevice Lib "biusb.dll" (
            ByRef oDev_Cnt As Integer,
            ByRef opDeviceList As DeviceParam,
            ByVal iProductID As Long,
            ByVal iFlag As Long
) As Boolean
```

The **DetectDevice** function detects only compatible vendor specific USB HID devices matching the specified product ID. This should be the first function called prior to all other calls or to refresh the device list.

Parameters

oDev_Cnt

[out] Return pointer to the total number of matching devices detected.

oDeviceList

[out] Pointer to a device parameter structure. All returned data is stored at this structure location.

iProductID

[in] This parameter specifies the product type to detect.

HID Class Devices	**Name**
PID_0X64B	0x64B input module
PID_GAMMARAY	GammaRay-256 input module
PID_GAMMARAY64	GammaRay-64 input module
PID_ELECTRONFLUX	ElectronFlux input/output module
PID_GAMMARAY_V2	GammaRay V2 input module
PID_GAMMARAY_V3	GammaRay V3 input module
PID_GAMMATRON	GammaTron input module
PID_NITRO_SLG	Nitro-SLG output module
Joystick Class Devices	**Name**
PID_4X24BH	4x24BH joystick module
PID_6X16B	6x16B joystick module
PID_6X13B	6x13B joystick module
PID_5X18BH	5x18BH joystick module
PID_PLASMA	Plasma V1 joystick module
PID_PLASMA_HOTAS	Plasma V1 in HOTAS mode joystick module
PID_PLASMA_DUAL	Plasma V1 in DUAL mode joystick module
PID_PLASMA_HOTAS_DUAL	Plasma V1 in HOTAS DUAL mode joystick module
PID_PLASMA_LITE	Plasma-Lite joystick module
PID_PLASMA_LITE_V2	Plasma-Lite V2 joystick module

PID_PLASMA_MM2	Plasma-MM2 joystick module

iFlag

[in] This parameter specifies the device class to detect.

Value	Meaning
DT_FLASH	Reserved
DT_DEVICES	Detect HID class modules that accept input commands or data from host. These are typically output class devices.
DT_REPORT	Reserved
DT_HID	Detect HID input class modules. Will ignore all Joystick type devices such Plasma class modules.
DT_ALL	Detect all HID input class modules including Joystick type devices such as Plasma modules.

Return Codes

TRUE	Success detecting devices.
FALSE	Failure detecting devices or no devices found.

RetrieveStatus

```
Public Declare Sub RetrieveStatus Lib "biusb.dll" (
            ByRef iDeviceList As DeviceParam,
            ByRef oDeviceStatus As DeviceStatus
)
```

The **RetrieveStatus** function returns the configuration status of all IO ports found on the module. Can be used to verify if ports are active before reading or writing to them. This function is typically called after **DetectHID**.

Parameters

iDeviceList

[in] Pointer to a device parameter structure. Must contain valid device information returned from call to **DetectHID.**

oDeviceStatus

[out] Pointer to a device status array. All returned data will be stored in this location.

Return Codes

None	

CloseDevices

```
Public Declare Function CloseDevices Lib "biusb.dll" (
            ByVal iDev_Cnt As Long,
            ByRef iDeviceList As DeviceParam
) As Boolean
```

The **CloseDevices** function releases all detected modules, cancels all pending threads and frees all used memory blocks. This call should be made on program exit and prior to any calls to the **DetectHID** function in order to refresh the device list.

Parameters

iDev_Cnt

[in] Number of detected modules to close in the *iDeviceList*. This value is retuned by call to **DetectHID.**

iDeviceList

[in] Pointer to a device parameter structure. Must contain valid device information returned from call to **DetectHID.**

Return Codes

TRUE	Success closing devices.
FALSE	Failure closing devices or releasing memory.

ReadInputData

```
Public Declare Function ReadInputData Lib "biusb.dll" (
        ByRef iDeviceList As DeviceParam,
        ByRef oDataBuffer As Byte,
        ByVal iFlag As Long
) As Long
```

The **ReadInputData** function is a non-blocking call, which returns input data. The first call to this function initiates a request for input data from the specified module as indicated by the *DeviceParam* structure and immediately returns from the function. This prevents the main calling application from locking while the module is being polled. The return flag indicates the current status of the request.

A return status of *DEV_INPUT* indicates new data is available at the location pointed to by *oDataBuffer* for the module specified in the *DeviceParam* structure. The *NumberInputIndices* within the *DeviceParam* structure indicates the number of input values to be read and size of the required array.

If no new data is available, the previously buffered data will be returned in addition to the *DEV_WAIT* flag indicating the module has not yet responded to the request for input data.

Depending on the specific module, polling frequency may vary in the range from 10ms to 40ms or more before data is flagged as being available. As a general rule, this function should be polled every 10 ms if real-time input data is required.

If a device should become unplugged or is no longer responding, the *DEV_TIMEOUT* flag will be returned after 1 second has elapsed.

Parameters

iDeviceList

[in] Pointer to a device parameter structure. Must contain valid device information returned from call to **DetectHID.**

oDataBuffer

[out] Pointer to the returned data buffer array. *NumberInputIndices* within the *DeviceParam* structure indicates the required length of this data buffer array specifying the number of detected inputs for the module.

iFlag

[in] Must always be set to FALSE.

Return Codes

DEV_TIMEOUT	Device did not respond within 1 second.
DEV_FAILED	Failure reading from device.
DEV_WAIT	No new data available. Waiting for device response.
DEV_INPUT	New data available.

ClearLCD

```
Public Declare Function ClearLCD Lib "biusb.dll" (
            ByRef iDeviceList As DeviceParam,
            ByVal inLCD As Byte
) As Long
```

The **ClearLCD** function sends a command to clear contents of the LCD screen. This function can be used on modules supporting character based LCDs.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inLCD

[in] Zero based index of LCD to be cleared. Currently not support. Leave as 0.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteLCD

```
Public Declare Function WriteLCD Lib "biusb.dll" (
            ByRef iDeviceList As DeviceParam,
            ByVal inLCD As Byte,
            ByVal inLine As Byte,
            ByRef iDataBuffer As Byte,
            ByVal inLevel As Byte
) As Long
```

The **WriteLCD** function writes contents of *iDataBuffer* to the LCD specified by *inLCD* at the line number indicated by *inLine*. This function can be used on modules supporting character based LCDs.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inLCD

[in] Zero based index of LCD to write to. Currently not support. Leave as 0.

inLine

[in] Line number to write buffer to. Values range from 1 to 4.

iDataBuffer

[in] Pointer to character array. Must be at least 20 characters in length. Empty spaces on the LCD should be filled with the ASCII value of &H20 (blank space). Do not NULL terminate this buffer. NULL value will display character &H0 stored in the LCD ROM.

inLevel

[in] LCD backlight brightness level if supported. Values range from 0 (min brightness) to 255 (maximum brightness).

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteDirectOutput

```
Public Declare Function WriteDirectOutput Lib "biusb.dll" (
            ByRef iDeviceList As DeviceParam,
            ByVal inPort As Byte,
            ByRef iDataBuffer As DIRECT_OUTPUT
) As Long
```

The **WriteDirectOutput** function writes contents of *iDataBuffer* to the output pins on the port specified by *inPort*. This function can be used on modules supporting direct outputs.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to data array defined by DIRECT_OUTPUT structure.

```
Public Type DIRECT_OUTPUT
    state As Byte
End Type
```

Field	Description
state	Output state value 0 for OFF and 1 for ON.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteMuxOutput

```
Public Declare Function WriteMuxOutput Lib "biusb.dll" (
          ByRef iDeviceList As DeviceParam,
          ByVal nPort As Byte,
          ByRef oDataBuffer As DIRECT_OUTPUT
) As Long
```

The **WriteMuxOutput** function writes contents of *iDataBuffer* to the multiplexed output pins on the port specified by *inPort*. This function can be used on modules supporting multiplexed outputs.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to data array defined by DIRECT_OUTPUT structure. The state parameter is comprised of bit fields as described below.

```
Public Type DIRECT_OUTPUT
   state As Byte
End Type
```

Field	**Description**
state	bit 1: Output state value 0 for OFF and 1 for ON. bits 2 - 8: Output level range from 0 to 10. This value sets the pulse width of the multiplexed output controlling intensity.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteDisplayOutput

```
Public Declare Function WriteDisplayOutput Lib "biusb.dll" (
            ByRef iDeviceList As DeviceParam,
            ByVal inPort As Byte,
            ByRef iDataBuffer As Byte,
            ByVal inLevel As Byte
) As Long
```

The **WriteDisplayOutput** function writes contents of *iDataBuffer* to the 7-segment display on the port specified by *inPort*. This function can be used on modules supporting 7 x 7-segment displays.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to character array. Maximum number of display characters is 7, however, decimals do not count as character spaces. Supported display characters are limited to ASCII characters “**0**” to “**9**”, “**.**”, “**-**“ and blank space (NULL).

inLevel

[in] Output level range from 0 to 10. This value sets the pulse width of the multiplexed display output controlling intensity.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

Write7Segment

```
Public Declare Function Write7Segment Lib "biusb.dll" (
            ByRef iDeviceList As DeviceParam,
            ByRef iData As Byte
) As Long
```

The **Write7Segment** function writes contents of *iData* to a single 7-segment display on the device specified by *iDeviceList*. This function can be used on modules supporting a 7-segment display and a single indicator output typically used for a shift light.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

iData

[in] Pointer to a 2 Byte character array. The first Byte value contains the character to be displayed on a 7-segment display. Supported values are &H0 to &H09 in hex which will display the corresponding ASCII characters "**0**" to "**9**" when using industry standard decoders. Non-supported values will blank the display. Note: do not use the ASCII value to be displayed. Example, to display the value "1", set *iData* to the hex value &H1, not &H31 corresponding to the ASCII value of "1".

Value	Displayed symbol
0	0 or N if supported
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	-
11	L
12	P
13	=
14	Blank
15	R

The second byte value is used for a shift light indicator with 0 for OFF and 1 for ON.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteSPIDAC

```
Public Declare Function WriteSPIDAC Lib "biusb.dll" (
            ByRef iDeviceList As DeviceParam,
            ByVal inPort As Byte,
            ByRef iDataBuffer As DAC_OUTPUTS,
            ByVal inLevel As Byte
) As Long
```

The **WriteSPIDAC** function writes contents of *iDataBuffer* to SPI compatible peripherals or DACs on the port specified by *inPort*. This function can be used on modules supporting either DACs or SPI ports.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to data array defined by DAC_OUTPUTS structure.

```
Public Type DAC_OUTPUTS
   DAC_1 As Integer
   DAC_2 As Integer
   DAC_3 As Integer
   DAC_4 As Integer
   DAC_5 As Integer
   DAC_6 As Integer
   DAC_7 As Integer
   DAC_8 As Integer
   DAC_9 As Integer
   DAC_10 As Integer
End Type
```

A port supporting 10 DAC or SPI outputs will require a 40 Byte array, each 32 bits long word corresponding to the data to be written to the corresponding DAC/SPI output. Note that all SPI/DACs on a port are written to at the same time. Writing to a single SPI/DAC is not possible; therefore SPI/DAC buffer data must be updated in *iDataBuffer* for all SPI/DACs on the specified port.

Actual data length output on a corresponding SPI/DAC channel is determined by the module's capability, not the maximum word length of 32 bits per channel as defined in the structure above.

inLevel

[in] Output level range from 0 to 7. This value sets the pulse width of the backlight if supported.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

CmdGLCD

```
Public Declare Function CmdGLCD Lib "biusb.dll" (
            ByRef iDeviceList As DeviceParam,
            ByVal inPort As Byte,
            ByVal iBGColor As Byte,
            ByVal iCmd As Byte
) As Long
```

The **CmdGLCD** function sends the specified command in *iCmd* to a graphic LCD on the port specified by *inPort*. This function can be used on modules supporting graphic based LCDs.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iBGColor

[in] Set the background color of the LCD display. 0 = Black text on white background, 1 = White text on black background.

iCmd

[in] GLCD Commands:

```
Public Const GLCD_CLEAR = 0
Public Const GLCD_TEST = 1
Public Const GLCD_BGCOLOR = 3
```

Define	Description
GLCD_CLEAR	Clear contents of LCD.
GLCD_TEST	Display test message.
GLCD_BGCOLOR	Set background color of LCD specified by *iGBColor*.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteGLCD

```
Public Declare Function WriteGLCD Lib "biusb.dll" (
            ByRef iDeviceList As DeviceParam,
            ByVal inPort As Byte,
            ByVal inLine As Byte,
            ByVal iNumLines As Byte,
            ByVal iBGColor As Byte,
            ByRef iDataBuffer As Byte
) As Long
```

The **WriteGLCD** function writes contents of *iDataBuffer* to the graphic LCD located on the port specified by *inPort* at the line number indicated by *inLine*. This function can be used on modules supporting graphic based LCDs.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

inLine

[in] Starting line number to write contents of *iDataBuffer* to LCD. Values range from 1 to 5.

iNumLines

[in] Number of lines contained in *iDataBuffer*. Maximum buffer length is 2 lines at 24 characters per line (48 bytes).

iBGColor

[in] Set the background color of the LCD display. 0 = Black text on white background, 1 = White text on black background.

iDataBuffer

[in] Pointer to character array. Empty spaces on the LCD should be filled with the ASCII value of &H20 (blank space). Do not NULL terminate this buffer.

Special Characters.

ASCII	Description
&H1	Arrow glyph.
&H2	Star glyph.
&H3	Degree glyph.
&H4	Star glyph.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteLatchedOutput

```
Public Declare Function WriteLatchedOutput Lib "biusb.dll" (
            ByRef iDeviceList As DeviceParam,
            ByVal iPort As Byte,
            ByRef iDataBuffer As Byte,
            ByVal inLevel As Byte
) As Long
```

The **WriteLatchedOutput** function writes contents of *iDataBuffer* to the port specified by *iPort* configured for latched output mode.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to a 64 byte array (64 outputs). Output state value 0 for OFF and 1 for ON.

inLevel

[in] Output level range from 0 to 10. This value sets the pulse width of the output, controlling the intensity if supported.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteCRTRWR

```
Public Declare Function WriteCRTRWR Lib "biusb.dll" (
        ByRef iDeviceList As DeviceParam,
        ByVal iPort As Byte,
        ByVal inContacts As Byte,
        ByVal inSymbolCnt As Byte,
        ByVal inContactIndex As Byte,
        ByRef iDataBuffer As CRTRWR_OUTPUT
) As Long
```

The **WriteLatchedOutput** function writes contents of *iDataBuffer* to the port specified by *inPort* configured for CRTRWR output mode.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

inContacts

[in] Total number of contacts to display on screen. This value is typically same as *inSymbolCnt*.

inSymbolCnt

[in] Number of symbols stored in *iDataBuffer.*

inContactIndex

[in] index within *iDataBuffer* of the first symbol.

iRWRBuffer

[in] Pointer to data array defined by CRTRWR_OUTPUT structure.

```
Public Type CRTRWR_OUTPUT
        symbol As Byte
        xPos As Byte
        yPos As Byte
        options As Byte
        ' missileActivity bit 0 - draw flashing circle
        ' missileLaunch bit 1 - draw solid circle
        ' newguy bit 2
        ' selected bit 3 - draw diamond if 1
End Type
```

Bit Field	Description
symbol	Symbol ID to display. (refer to RWR Symbol Class table Entries found in rwr.bas)
xPos	X position on screen to display symbol. Range 0 (left) to 255

	(right).
yPos	Y position on screen to display symbol. Range 0 (top) to 255 (bottom).
Options	bit 1: missileActivity - If set to 1, display missile activity symbol. bit 2: missileLaunch - If set to 1, display missile launch symbol. bit 3: newguy - If set to 1, display new contact symbol. bit 4: selected - If set to 1, draw contact selected symbol. bits 5-8: not used.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

WriteDotMatrix

```
Public Declare Function WriteDotMatrix Lib "biusb.dll" (
            ByRef iDeviceList As DeviceParam,
            ByVal iPort As Byte,
            ByRef iDataBuffer As Byte,
            ByVal inLevel As Byte
) As Long
```

The **WriteDotMatrix** function writes contents of *iDataBuffer* to 4 OSRAM type dot matrix displays on the port specified by *inPort*.

Parameters

iDeviceList

[in] Pointer to the device parameter structure. Must contain valid device information returned from call to **DetectHID.**

inPort

[in] Zero based index of output Port to write to.

iDataBuffer

[in] Pointer to a 16 byte character array. Maximum number of display characters is 4 per display (4 displays max per port). Supported display characters are limited to ASCII values (0 – 255). Refer to manufacturer's ACSII table for symbol defines.

inLevel

[in] Output level range from 0 to 10. This value sets the pulse width of the dot matrix display controlling intensity.

Return Codes

0	Failure writing to device.
> 0	Success. Number of bytes written to device.

VB Structures & Defines

The following structures and defines are part of this DLL module.

DeviceParam Structure

```
Public Type DeviceParam
        DeviceHandle As Long
        NumberInputIndices As Integer
        NumberOutputIndices As Integer
        NumberPortIndices As Integer
        NumberAnalogIndices As Integer
        NumberHATIndices As Integer
        NumberRotaryIndices As Integer
        NumberCRTRWRIndices As Integer
        NumberPWMIndices As Integer
        NumberLCDIndices As Integer
        NumberGLCDIndices As Integer
        NumberMuxDisplayIndices As Integer
        NumberAlphanumericIndices As Integer
        NumberDACIndices As Integer
        NumberSPIIndices As Integer
        NumberLatchedIndices As Integer
        NumberDotMatrixIndices As Integer
        VendorID As Integer
        ProductID As Integer
        VersionNumber As Integer
        FlashVersion As Integer
        ProgramFlag As Byte
        DevicePath(MAX_STRING_CHARS) As Byte
        DevicePathLength As Long
        DeviceName(MAX_STRING_CHARS) As Byte
        DeviceNameLength As Long
        ManufName(MAX_STRING_CHARS) As Byte
        ManufNameLength As Long
        SerialNum(0 To 3) As Byte
        SerialNumLength As Long
        ConfigFlag As Byte
        DevIndex As Integer
        InputReportByteLength As Integer
        OutputReportByteLength As Integer
        Usage As Integer
        UsagePage As Integer
End Type
```

Device specific data structure.
Defined in: ApiDeclarations.bas

Parameters

DeviceHandle: The read/write handle for the device. This is a unique value and changes with each call to DetectHID.

NumberInputIndices: The number of unpacked inputs that will be returned from this device.

NumberOutputIndices: The number of unpacked outputs that will be read by this device.

NumberPortIndices: The number of separate IO ports on a device. These can also refer to JPs found on older devices.

NumberAnalogIndices: Not currently supported.

NumberHATIndices: Not currently supported.

NumberRotaryIndices: Not currently supported.

NumberCRTRWRIndices: Number of RWR supported.

NumberPWMIndices: Not currently supported.

NumberLCDIndices: Number of character based LCD's supported.

NumberGLCDIndices: Number of graphic based LCD's supported.

NumberMuxDisplayIndices: Number of multiplexed displays supported.

NumberAlphanumericIndices: Not currently supported.

NumberDACIndices: Number of DAC (Digital to Analog Converter) supported.

NumberSPIIndices: Number of SPI interface devices supported.

NumberLatchedIndices: Number of latched outputs supported.

NumberDotMatrixIndices: Number of OSRAM type dot matrix displays supported.

VendorID: USB.org assigned unique vendor ID number.

ProductID: Vendor assigned unique device ID number.

VersionNumber: Vendor assigned product version number.

FlashVersion: Current Firmware revision of device.

ProgramFlag: For internal use only.

DevicePath: Pointer to the device path key string as found in the Windows registry.

PathLength: Length in bytes of the *DevicePath* string.

DeviceName: Pointer to the device name string.

DeviceNameLength: Length in bytes of the *DeviceName* string.

ManufName: Pointer to the manufacturer name string.

ManufNameLength: Length in bytes of the ManufName string.

SerialNum: Pointer to the user assigned device serial number string.

SerialNumLength: Length in bytes of the *SerialNum* string.

ConfigFlag: For internal use only.

DevIndex: The zero based device index associated with this device.

InputReportByteLength: The number of packed bytes that will be returned by the device on each *ReadInputData* read request.

OutputReportByteLength: The number of packed bytes used by the device for output data.

Usage: For internal use only.

UsagePage: For internal use only.

DeviceStatus Structure

```
Public Type DeviceStatus
    InputActive(MAX_INPUTS) As Byte
    OutputActive(MAX_OUTPUTS) As Byte
    PortActive(MAX_PORTS) As Byte
    PortIOMode(MAX_PORTS) As Byte
    PortMode(MAX_PORTS) As Byte
    OutputMode(MAX_PORTS) As Byte
    AnalogActive(MAX_ANALOG) As Byte
    HATActive(MAX_HATS) As Byte
    RotaryActive(MAX_ROTARY) As Byte
    PWMActive(MAX_PWM) As Byte
    LCDActive(MAX_LCDS) As Byte
    GLCDActive(MAX_GLCDS) As Byte
    MuxDisplayActive(MAX_MUXDISPLAYS) As Byte
    AlphanumericActive(MAX_ALPHANUMERICS) As Byte
    DACActive(MAX_DACS) As Byte
    SPIActive(MAX_SPI) As Byte
    LatchedActive(MAX_LATCHED) As Byte
    DotMatrixActive(MAX_DOTMATRIX) As Byte
End Type
```

Device specific data structure.
Defined in: ApiDeclarations.bas

Parameters

InputActive
Inputs is active if 1, disabled if 0.
OutputActive
Output is active if 1, disabled if 0.
PortActive
Port is active if 1, disabled if 0.
PortIOMode
Port is set to input mode if 1, output mode if 0.
PortMode
Currently only valid for ElectronFlux class devices. Return values are:

```
Public Const MODE_DIRECT = 0
Public Const MODE_MUX = 1
Public Const MODE_RWR = 2
Public Const MODE_GLCD = 3
Public Const MODE_DAC = 4
Public Const MODE_LATCHED = 5
Public Const MODE_SPI = 6
Public Const MODE_DOTMATRIX = 7
```

OutputMode: Not currently supported.

AnalogActive: Analog channels active if 1, disabled if 0.

HATActive: Not currently supported.

CRTRWRActive: CRT RWR channels active if 1, disabled if 0.

PWMActive: Not currently supported.

LCDActive: Not currently supported.

rrently supported.

AnalogActive: Analog channels active if 1, disabled if 0.

HATActive: Not currently supported.

CRTRWRActive: CRT RWR channels active if 1, disabled if 0.

PWMActive: Not currently supported.

LCDActive: Not currently supported.

GLCDActive: Graphic LCD channels active if 1, disabled if 0.

MuxDisplayActive: Multiplexed display channels active if 1, disabled if 0.

AlphanumericActive: Not currently supported.

DACActive: DAC channels active if 1, disabled if 0.

SPIActive: SPI channels active if 1, disabled if 0.

LatchedActive: Latched channels active if 1, disabled if 0.

DotMatrixActive: DotMatrix channels active if 1, disabled if 0.