GUIDE DE L'UTILISATEUR

```
//**********************EPICUSB HID enque16/nqw commands ***************
//Usage:
//enque16(<command>,<data16>)
//or
//nqw(<command>,<data16>)
// command range 0 to 0xFFF
//
// These commands are for the most part directly compatible with the Dowson VXD commands.
//
// Commands BtnOn(1),BtnOff(3),BtnP(3),BtnToggle(6) and variatons of these are sent to PC as USB HID reports to DirectInput.
// the format is nqw(<BtnOn,BtnOff, or BtnP>, 0xddbb);
// where dd is the device number and bb is the button number on the device.
// Devices and buttons start at 0.
//
//SetBcdDIPovx commands (0x400-0x40f) are converted to BCD, then sent to the Direct Input device(x) POV
//
//ResendAllHIDdevices (0x2FE) is translated internally and not sent
//
// VIRTUAL device.analog HID DirectInput equivalents:
//
// Setaxis     devicename.analogname=data;
// Incaxis     devicename.analogname++; or devicename.analogname += data;
// Decaxis     devicename.analogname--; or devicename.analogname -= data;
// SetMinaxis  devicename.analogname.min = data;
// SetMaxaxis  devicename.analogname.max = data;
// The min and max are set in the device descriptor:
// analog(VIRTUAL, name, WinAxis, type=MEM8, source=0, min=0, max=255);
// analog(VIRTUAL, v1, Rx, MEM16, ,0,0xFFFF); 16bit virtual analog named v1, 0-0xFFFF
// SetSAxesMin  no equivalent
// SetSAxesMax  no equivalent
// SetCenaxis   no equivalent
// setmapaxis   devicename.analogname.source=newAxis; (if .type=ANALOG8, if not change .type)
//
// Offsetaxis   ***before Rate curve**
// devicename.analogname.pre_offset = data;
// devicename.analogname.pre_offset++;
// devicename.analogname.pre_offset--
// ***after Rate curve ***
// devicename.analogname.post_offset = data;
//The event buffer is extracted by user programs with the GetEvent(int handle,EPIC_EVENT_STRUCT *event)
```

```
//function.
//This will return a 0 if there is an event extracted to the event structure
// The value will be negative if empty, an error, or busy
//
//event.cmd is the enque16 command high byte with 0xA0 added to it.
//event.data[0] is the sub command and is the low byte of the enque16 command
//event.data[1] is the low byte of data16 and event.data[2] is the high byte of data16.
//
// ****** common defines ********
  #ifndef MODTYPES
  #define MODTYPES

  #define FASTSCAN    0
  #define SLOWSCAN    1
  #define BITOUTPUT   2
  #define OUTPUT              2
  #define 7SEGDISP    3
  #define NODEBOUNCE          4
  #define NOSCAN              5

#endif

//******* Program and destination IDs ******************
// nqw(cmd,data,destID);  cmd can be 0 to 0xFFF, data is 16bit,
// destID is the program destination if registered
// cmd range 0-255 (0-0xFF) if sent to another EPIC device
// example nqw(0x10,myData,FLCC); will send command 0x10,myData to Flight Link Control Center
//              nqw(0x10,myData,EPICUSB2); will send to pigeon hole 16 (0x10) in EPICUSB2
//
#define EPICISA         0
#define EPICUSB0        1
#define EPICUSB1        2
#define EPICUSB2        3
#define EPICUSB3        4
#define EDEBUG              5
// destIDs 6-8 reserved for EPICenter
// destID 9 reserved for compatibility with older programs that used _OpenDevice(device#);
#define JSOUND              11
#define FLCC                12              //Flight Link Control Center
#define FS_COMM             13              //Robert Fischer's FS-Communicator
#define EPIC_INFO           14              //Peter Dowson's EPICinfo
```

```
#define REVERSE 0x08                    //for reverse axis flag

//for device axis definitions
#define VIRTUAL 0xFF
#define ANALOG8 0
#define ANALOG10 1
#define ANALOG16 2
#define MEM8 3
#define MEM10 4
#define MEM16 5
#define MODROW8 6
#define MODROW10 7
#define MODROW16 8
//============================
// Button controls
#define BtnOn                   0x001       // Send to Direct Input
#define btnon                   0x001       // Send to Direct Input
#define BtnOff                  0x002       // Send to Direct Input
#define btnoff                  0x002       // Send to Direct Input
#define BtnToggle               0x006       // Send to Direct Input EPROM V4.03 and later
#define BtnPulse                0x003       //Send to Direct Input
#define BtnP                    0x003       //Send to Direct Input
#define btnp                    0x003       //Send to Direct Input
#define FIND_ATTACHED_EPICS     0x7FF       //DLL command to register attached EPICs


//X #define BtnRepeat 0x004
//X #define BtnR 0x004
//X #define BtnClear 0x005

//X #define BtnAllClear 0x010
//X #define BtnTogether 0x011

//X #define BtnSetdelay1 0x020
//X #define BtnSetRepeat1 0x021
//X #define BtnSetdelay2 0x022
//X #define BtnSetRepeat2 0x023

//X #define BtnSetBuffer 0x024
```

```
//This does NOT go to DLL buffer
// This is to resend all DirectInput devices, but does not seem to do what it is supposed to
//
#define ResendAllHIDdevices 0x2FE

//This DOES go to DLL buffer
#define ResetAllAxes 0x2ff
#define ReverseAxis 0x3C0                    //obsolete.  use DeviceName.AxisName.flag |= REVERSE;

//SetBcdDIpovX (where X is the device number) will convert a 16 bit value to BCD and send to the device POV
#define SetBcdDIPov0 0x400
#define SetBcdDIPov1 0x401
#define SetBcdDIPov2 0x402
#define SetBcdDIPov3 0x403
#define SetBcdDIPov4 0x404
#define SetBcdDIPov5 0x405
#define SetBcdDIPov6 0x406
#define SetBcdDIPov7 0x407
#define SetBcdDIPov8 0x408
#define SetBcdDIPov9 0x409
#define SetBcdDIPov10 0x40a
#define SetBcdDIPov11 0x40b
#define SetBcdDIPov12 0x40c
#define SetBcdDIPov13 0x40d
#define SetBcdDIPov14 0x40e
#define SetBcdDIPov15 0x40f

//**********************The following are to maintain compatibility with the old Dowson Vxd codes ******************
// most of these can be replaced by EPL commands (see notes above on replacement usage)
// If these codes are used they will end up at the EPICIO.DLL as buffered events
//================================================================
// «Soft» joystick axes Set directly stores given value for axis:
#define SetU0 0x130
#define SetV0 0x131
#define SetU1 0x132
#define SetV1 0x133
#define SetU2 0x134
#define SetV2 0x135
#define SetU3 0x136
#define SetV3 0x137
#define SetX4 0x100
```

```
#define SetY4 0x101
#define SetZ4 0x103
#define SetR4 0x102
#define SetU4 0x138
#define SetV4 0x139
#define SetX5 0x104
#define SetY5 0x105
#define SetZ5 0x107
#define SetR5 0x106
#define SetU5 0x13a
#define SetV5 0x13b
#define SetX6 0x108
#define SetY6 0x109
#define SetZ6 0x10b
#define SetR6 0x10a
#define SetU6 0x13c
#define SetV6 0x13d
#define SetX7 0x10c
#define SetY7 0x10d
#define SetZ7 0x10f
#define SetR7 0x10e
#define SetU7 0x13e
#define SetV7 0x13f
#define SetX8 0x110
#define SetY8 0x111
#define SetZ8 0x113
#define SetR8 0x112
#define SetU8 0x140
#define SetV8 0x141
#define SetX9 0x114
#define SetY9 0x115
#define SetZ9 0x117
#define SetR9 0x116
#define SetU9 0x142
#define SetV9 0x143
#define SetX10 0x118
#define SetY10 0x119
#define SetZ10 0x11b
#define SetR10 0x11a
#define SetU10 0x144
#define SetV10 0x145
```

```
#define SetX11 0x11c
#define SetY11 0x11d
#define SetZ11 0x11f
#define SetR11 0x11e
#define SetU11 0x146
#define SetV11 0x147
#define SetX12 0x120
#define SetY12 0x121
#define SetZ12 0x123
#define SetR12 0x122
#define SetU12 0x148
#define SetV12 0x149
#define SetX13 0x124
#define SetY13 0x125
#define SetZ13 0x127
#define SetR13 0x126
#define SetU13 0x14a
#define SetV13 0x14b
#define SetX14 0x128
#define SetY14 0x129
#define SetZ14 0x12b
#define SetR14 0x12a
#define SetU14 0x14c
#define SetV14 0x14d
#define SetX15 0x12c
#define SetY15 0x12d
#define SetZ15 0x12f
#define SetR15 0x12e
#define SetU15 0x14e
#define SetV15 0x14f

// Calibration aids:
#define SetSAxesMin 0x1fd      // Sets all soft axes to their current minima
#define SetSAxesMax 0x1fe      // Sets all soft axes to their current maxima

// Restore all «soft» axes to default values i.e. all X, Y, R centred, all Z, U, V maximum
#define RestoreSAxes 0x1ff
//============================
// Point Of View (POV) controls
// Set provides 360 degree settings
```

```
#define setpov0 0x300
#define setpov1 0x301
#define setpov2 0x302
#define setpov3 0x303
#define setpov4 0x304
#define setpov5 0x305
#define setpov6 0x306
#define setpov7 0x307
#define setpov8 0x308
#define setpov9 0x309
#define setpov10 0x30a
#define setpov11 0x30b
#define setpov12 0x30c
#define setpov13 0x30d
#define setpov14 0x30e
#define setpov15 0x30f
//
// SetFine provides 36000 1/100th degree settings,
// or can be used to transfer any 16-bit value (0-65535)
#define SetFinePov0 0x330
#define SetFinePov1 0x331
#define SetFinePov2 0x332
#define SetFinePov3 0x333
#define SetFinePov4 0x334
#define SetFinePov5 0x335
#define SetFinePov6 0x336
#define SetFinePov7 0x337
#define SetFinePov8 0x338
#define SetFinePov9 0x339
#define SetFinePov10 0x33a
#define SetFinePov11 0x33b
#define SetFinePov12 0x33c
#define SetFinePov13 0x33d
#define SetFinePov14 0x33e
#define SetFinePov15 0x33f


// SetBcd is also used for 16-bit values, but the VxD converts the binary value to Decimal coding. The value
// needs to be in the range 0 - 9999 and is converted to a BCD (binary coded decimal) value 0x0000 - 0x9999
// before delivering to the target program
```

```
#define SetBcdPov0 0x350
#define SetBcdPov1 0x351
#define SetBcdPov2 0x352
#define SetBcdPov3 0x353
#define SetBcdPov4 0x354
#define SetBcdPov5 0x355
#define SetBcdPov6 0x356
#define SetBcdPov7 0x357
#define SetBcdPov8 0x358
#define SetBcdPov9 0x359
#define SetBcdPov10 0x35a
#define SetBcdPov11 0x35b
#define SetBcdPov12 0x35c
#define SetBcdPov13 0x35d
#define SetBcdPov14 0x35e
#define SetBcdPov15 0x35f

//============================
// Reset real axes to no scale no offset, no min/max, and no mapping, with all Win95 EPIC95 controlled
// joysticks enabled.  (Note this is also done by «ResetAllAxes», above) #define ResetRealAxes 0x3ff

//============================
// Mouse facilities

// NB, except for mouse button clicking, all the Mouse facilities
// require the utility EpicMous.EXE to be running!
#define MouseXScale 0x0f6      // Stores screen width (X) for X coordinates
#define MouseYScale 0x0f7      // Stores screen height (Y) for Y coordinates

// if these are initialised (say in :init) then all the X,Y mouse coordinates used in the MouseX and MouseY
//commands (below) are scaled appropriately if the screen resolution set at the time is different to the size
// to the size set here. The same applies to displayed/PH X,Y positions. (This latter feature can therefore be
// very useful in doing FS98 panel design -- set the X,Y scale values to the *designed* size of the panel,
// so that the mouse positions are (relatively) correct).

//********** USE EPIC command movemouse(x,y) (see SYNTAXUSB.DOC)
//where 0,0 is upper left
//0x8000,0x8000 is center of screen
//0xFFFF,0xFFFF is lower right (cursor will disapear here)

//**************these are in but do nothing at this time.******************
```

```
#define MouseX 0x0f8          // Stores X-coordinate for MouseBtn, below
#define MouseY 0x0f9          // Stores Y-coordinate for MouseBtn, below
#define MouseBtn 0x0fa        // Moves mouse to set (X,Y) and operates buttons

// To drive the Mouse  the (X,Y) screen coordinate, using the MouseX and MouseY calls then use
// the MouseBtn calls to operate buttons. A single button click on the left button, click on the left button,
// (MouseBtn,12), is the most useful.

// MouseBtn controls buttons as follows:
// 0 = no buttons pressed
// 1 = Centre button pressed or clicked
// 2 = Right button pressed or clicked
// 4 = Left button pressed or clicked
// 8 = Single click (auto press and release)
// 16= Double click (auto press and release twice in quick succession)

// Note that Left/Right may be swapped by Windows' mouse facilities.
#define MousePosPH 0x0fb      // Requests mouse positions to PH
#define MousePosdis 0x0fc     // Requests mouse positions to Displays

// For FS98, a more sophisticated method of using the Mouse is provided, which allows tailoring
// to each FS98 aircraft panel without needing different EPL  control programs for each.
// This is accomplished by making the EPICINFO Gauge set the Scaling and set up to 63 pairs of (x,y)
// coordinates, numbered 1-63 (01-3F hex). Then,  the EPL merely calls up the position number in the
// MouseBtn. command this goes into the top part of the button command value, so:
// enque16(MouseBtn,0x070c)
// does a left-button click on position number 7. This will do nothing at all
// (not even click the mouse) if that position is undefined at the time.

//============================
// Miscellaneous specials

//#define ReloadReset 0x0fd
// This isn't used by EPL programs, but is reserved for use
// by the EPIC firmware to inform the user program that a new EPL
// control program has just been loaded.
#define EnableJoys 0x0fe

// This specifies which of the 16 possible
// joysticks should be enabled or disabled. By default
// all 16 are enabled (assuming they are all entered into
```

```
// «Game Controllers»). Each bit in the parameter represents
// one joystick, with J0 as bit 0x0001, J1 bit 0x0002, etc.
// ReasetAllAxes and ResetRealAxes both reset this to ALL
// joysticks enabled.
//
// Thus enque16(EnableJoys,0xffff) enables all 16 joysticks
// whilst enque16(EnableJoys,0x0001) only enables J0.
#define SetOptions 0x0ff

//============================
```