Callback Events

Call back events related to the main callback function TBApiRegisterEvent() and upddapi.h.

This document covers additional points related to the Callback Events.

EventTypeDigitiserEvent

For UPDD V6 a new callback type has been introduced:

#define _EventTypeDigitiserEvent 0x4000000

This provides a single event for touch information. This is intended to be a replacement for XY and PhysicalEvents and Flags events typically used in V5.

Flags events are no longer supported, but XY and PhysicalEvents will be retained. For new code _EventTypeDigitiserEvent offers a more complete / easier to implement solution.

The struct and related flags are shown below.

Some points to note:

digitizerType indicates if the device that generated this event is a pen or a touch device eg digitizerType == DIGITIZER_TYPE_PEN

The struct penEvent OR touchEvent is used dependant on digitizerType.

validBits indicates which bits are supported by the sending device (unsupported bits will be zero so this is only needed if a behavior change is needed based on supported bits)

screen / y provide co-ordinates scaled to the associated monitor

Z values are in the field z. HID also defines "tip pressure" but I've not seen a pen that uses this yet so for now I'm passing z or pressure values in z on the assumption that one or the other will be used; not both.

#defineTOUCH_BIT_FLAGS_LEFT 0x1

#defineTOUCH_BIT_FLAGS_RIGHT 0x2

```
#definePEN_BIT_FLAGS_BARREL 0x2
#definePEN_BIT_FLAGS_ERASER 0x4
#definePEN_BIT_FLAGS_IN_RANGE 0x8

#definePEN_BIT_FLAGS_INVERT 0x10

#defineDIGITIZER_TYPE_PEN 0x2
#defineDIGITIZER_TYPE_TOUCH 0x4
```

```
    uint8_t touchingLeft : 1; // bit flags relating to regular touch devices, relates to TOUCH_BIT_FLAGS_XXX above
    uint8_t touchingRight : 1;
   }touchEvent;
  }de;
  uint8_t deltaBits;        // a bit mask to indicate which bits are changed since last _digitiserEvent
  uint8_t validBits;        // a bit mask to indicate which bits are supported by the originating hardware
  long screenx;             // screen co-ordinate values, these values are in screen pixels and take account of the co-ordinate range of the associated monitors
  long screeny;             // so for example with 2 monitors, resolution 1024 x 768 side by side; with the left monitor bieng the primary,
  // touching the centre of the right gives about 1536,384
  long internalx;           // the corresponding windows co-ordinate value, the primary monitor has the range 0xffff, and other monitors are scaled from that
  long internaly;           // so in the example given above the result is 0x17fee,0x7fff
  long calx;                // the calibrated co-ordinates values; a value from 0 - 0xffff, giving the absolute position of touch in the range of the originating hardware
  long caly;                // so for example touching the centre of a screen will give  around 7ff regardless of the associated monitor
  TBBOOL zSupport;          // set to TRUE (1) if the originating hardware supports z values
  unsignedlong z;           // the raw z value reported by the controller, typically this is used to indicate pressure
   TBBOOL isTimed;          // set to TRUE (1) if the event is
```

```
triggered by a timeout (eg liftoff time)
    TBBOOL isToolbar;        // set to TRUE (1) if the event is for a
touch start started in a toolbar
    TBBOOL stylusSupport;    // set to TRUE (1) if the originating
hardware supports stylus values
    uint8_t digitizerType;     // see DIGITIZER_TYPE_xxx
   }digitiserEvent;
```