UPDD API – Getting Started

Revision 1.4 – 3rd Feb 2014
www.touch-base.com\documentation\API

Code example | C# | Unicode | CE | Mac OS X | Settings file
Using callbacks | PointerData structure | Calibration style structure | 4.1.10 | | Contact

Having downloaded the files you may wish to write a simple application to see the UPDD API interface in action.

Code Example

The following code segment shows the skeleton of a simple C++ fragment that demonstrates:-

1. initialising the API
2. registering a callback routine to receive raw X/Y co-ordinates and notification of driver unload
3. shutting down

```
#include "tbapi.h"                                          // main API header file
// The callback function
void TBAPI fnCallBack(unsigned long context, _PointerData* data)
{
        if(data->pd.type == _ReadDataTypeXY
        {
                printf("Device %d Generated x=%d y=%dn",    // display the returned co-ordinates
                (int)data->pd.device,                       // device id
                (int)data->pd.xy->rawx,                     // raw x co-ordinate
                (int)data->pd.xy->rawy);                    // raw y co-ordinate
        }
        else if(data->pd.type == _ReadDataTypeUnload)       // driver is unloading!
        {
         // begin exit process…
        }

// At startup
{
        TBApiInit();                                        // initialise the API
        TBApiOpen();                                        // establish driver connection
        TBApiRegisterDataCallback(TBApiGetRelativeDevice(0),    // first device on system
        0,                                                  // no context data
        _ReadDataTypeXY | _ReadDataTypeUnload,              // 'OR' type(s) of data to return
        fnCallBack);                                        // address of callback function above
}
…

// At exit…

{
       TBApiUnregisterDataCallback(fnCallBack);             // unregister functions before exit
         TBApiClose();                                      // close the driver connection
         TBApiTerminate();                                  // and conclude use of API
}
```

C#

A .net c# user reported the following was required to run the API in C#

Import the functions from TBAPI.DLL using the DllImport command

[DllImport("C:\\Program Files\\UPDD\\TBAPI.dll")]

Then for each function, such as this example, (on a Windows system)

void TBAPI TBApiInit(const TBCHAR* aSettingsPath);

where TBAPI = _stdcall

and TBCHAR = char

Use mangled reference names DLL_TBApiInit - use a utility like dumpbin to identify the mangled entry names to avoid entry point errors such as 'Unable to find an entry point named 'TBApiInit' in DLL 'C:\Program Files\UPDD\TBAPI.dll'.

Unicode

Pre 4.1.10

With the exception of Windows CE, which is Unicode only, the UPDD API is MBCS based only. Including the tbapi.h file in a Unicode program causes the Unicode definitions (intended for Windows CE) to be activated.

This can be avoided as follows.

```
#undef _UNICODE
#include "TBApi.h"
#define _UNICODE
```

4.1.10 and above

The API uses MBCS in the public interface all environments except Windows CE, where Unicode must be used. An error in the previous implementation meant that tbapi.h could not be used directly in a Unicode based program. This is no longer

the case and previous partial solutions (such as "undef'ing" _UNICODE) are no longer required.

CE

Under CE you need to pass UNICODE, so for example, the TBApiInit call is coded as:

```
TBApiInit((L"\\hdd\updd\\tbupdd.ini"));

or

TBApiInit(_T("\\hdd\updd\\tbupdd.ini"));
```

Mac OS X - Objective-C Cocoa application with Xcode 5

After advising a user on API usage within a Objective-C Cocoa application with Xcode 5 project we wrote up our finding as follows:

1. Install the UPDD driver.
2. Download the OS X UPDD API files and copy them into your project directory.
3. Add libTBApi.a to your Xcode project.
4. Locate libACE.dylib in /usr/local/lib. It can be found in the Finder by selecting the "Go to folder" menu item in the Go menu, and typing in:

   /usr/local/lib

   Add it to your Xcode project.
5. In Xcode, open your project's settings, and in the "Build settings" section, locate the following settings and change them to the indicated values:
   - Under the "Architectures" group, the "Architectures" settings should be set to "32-bit Intel (i386)"
   - Under the "Search paths" group, add the following path to "Library Search Paths": /usr/local/lib
   - Under the "Apple LLVM 5.0 - Language - C++" group, the "C++ Standard Library" setting should be set to "libstdc++ (GNU C++ standard library)"
6. Change the extension of all Objective-C files that will make calls to the UPDD API from ".m" to ".mm", so that they are Objective-C++ files.
7. In the file nonwindows.h, locate the line:

   typedef int BOOL;

   Delete it, or comment it out

   and then use nonwindows.h and tbapi.h in your application

Troubleshooting tips:

If there are any link errors regarding UPDD or libACE functions, make sure that both libTBApi.a and libACE.dylib are included in the "Link binary with libraries" build phase. (Located in the Project Settings / Build Phases section.)

Also check that all of the project settings listed in Step 3 are configured correctly, as all of them are necessary. Otherwise building your project will result in link errors.

Settings file

The UPDD settings file is usually located in the UPDD application folder. When writing an application using the UPDD API then at the time the TBApiInit function is called the current working directory must be set to the folder containing the settings file. One way to achieve this is might be to run the program from the updd application folder.

Using Callbacks

Via the API, an application can register callbacks to receive notification of a wide range of events internal to the driver. To receive notification of these events applications should register interest via TBApiRegisterDataCallback defining the relevant event settings (data-type constants).

The nature of the registered callback determines the type of data returned in a PointerData structure (although not all events are accompanied by additional data).

It is important that applications are aware if the driver is no longer available. This can be as a result of:-

1. The driver is being un-installed
2. All device instances are deleted from the device manager (explicity or implicitly by the removal of hot pluggable PnP hardware).
3. Windows shutdown (although this happens so late in the process as to be irrelevant for practical purposes).

To receive notification of these 'unload' events applications should register interest via TBApiRegisterDataCallback supplying the _ReadDataTypeUnload parameter.

PointerData Structure

This structure (struct _PointerData) can be found in the header file TBapi.h. It is used to hold pointer (and other) data returned from the driver to a registered callback routine.

Calibration Style Structure

This structure (struct _CalStyle) can be found in the header file TBapi.h. It is used to define values associated with a calibration style (see TBApiSetcalibrationStyle).

UPDD 4.1.10 integration notes

UPDD version 4.1.10 introduces a number of changes to resolve problems reported with previous API implementations.
API programs written for earlier versions must be rebuilt and minor changes are required.

TBApiInit

The requirement to set the working directory before calling TBApiInit no longer applies.
A change must be made to the call to TBApiInit to determine the desired method to locate the UPDD settings.

1) Using the "usual default" location.

```
char path[1024];
TBApiDefaultSettingsPath(path,sizeof(path));
TBApiInit(path);
```

This approach uses the default installation path used in most cases.
Unless your installation is manually installed to a nonstandard location or you are writing advanced code to work with pre-installation settings during an installation, this approach should suffice.

2) Using an explicit path

```
TBApiInit("c:\\program files\\updd\\unusual location");
```

3) Using a NULL argument to force the same behaviour as used in previous 4.x version

```
TBApiInit(NULL);
```

Contact

For further information or technical assistance please email the technical support team at technical@touch-base.com.