AN2328
Application note

HID device coding example

Introduction

This application note describes how to lookup and use HID devices under Windows. This note is related to the MEMS Evaluation boards which are usually connected via USB and accessed through the MEMS USB Reader software. The following information will assist you in developing your own applications. It provides a section of source code only, however, for a complete project it is necessary to add proper result testing and exception handling.

1 Coding example

1.1 Load HID library

A dynamic loading hid library is used.

```
//Pointers to a function are used, therefore:
typedef VOID    (__stdcall *PHidD_GetProductString)(HANDLE, PVOID, ULONG);
typedef VOID    (__stdcall *PHidD_GetHidGuid)(LPGUID);
typedef BOOLEAN (__stdcall *PHidD_GetAttributes)(HANDLE, PHIDD_ATTRIBUTES);
typedef BOOLEAN (__stdcall *PHidD_SetFeature)(HANDLE, PVOID, ULONG);
typedef BOOLEAN (__stdcall *PHidD_GetFeature)(HANDLE, PVOID, ULONG);

HINSTANCE                    hHID                = NULL;
PHidD_GetProductString       HidD_GetProductString   = NULL;
PHidD_GetHidGuid             HidD_GetHidGuid     = NULL;
PHidD_GetAttributes          HidD_GetAttributes  = NULL;
PHidD_SetFeature             HidD_SetFeature     = NULL;
PHidD_GetFeature             HidD_GetFeature     = NULL;

//Load the library:
hHID = LoadLibrary("HID.DLL");

//Update the pointers:
HidD_GetProductString = (PHidD_GetProductString)
     GetProcAddress(hHID, "HidD_GetProductString");
HidD_GetHidGuid     = (PHidD_GetHidGuid)
     GetProcAddress(hHID, "HidD_GetHidGuid");
HidD_GetAttributes    = (PHidD_GetAttributes)
     GetProcAddress(hHID, "HidD_GetAttributes");
HidD_SetFeature       = (PHidD_SetFeature)
     GetProcAddress(hHID, "HidD_SetFeature");
HidD_GetFeature       = (PHidD_GetFeature)
     GetProcAddress(hHID, "HidD_GetFeature");
```

1.2 Lookup device

```
typedef struct _HIDD_ATTRIBUTES
{
  ULONG   Size;          // = sizeof (struct _HIDD_ATTRIBUTES)
  USHORT  VendorID;
  USHORT  ProductID;
  USHORT  VersionNumber;
} HIDD_ATTRIBUTES, *PHIDD_ATTRIBUTES;

GUID HidGuid;
(HidD_GetHidGuid)(&HidGuid);
HDEVINFO hDevInfo;
HANDLE DeviceHandle;

//Get information about HIDs
try {
  hDevInfo = SetupDiGetClassDevs
    (&HidGuid,NULL,NULL,DIGCF_PRESENT|DIGCF_INTERFACEDEVICE);
  if (hDevInfo == INVALID_HANDLE_VALUE)
    return 0;
} catch (...) {
```

```
  return 0;
}


//Identify each HID interface
devInfoData.cbSize = sizeof(devInfoData);
DWORD MemberIndex = 0;
bool Result;

while (1)
{
  try {
    Result = SetupDiEnumDeviceInterfaces
              (hDevInfo,0,&HidGuid,MemberIndex,&devInfoData);

    if (!Result)
    {
      SetupDiDestroyDeviceInfoList(hDevInfo);
      return (0); //No more devices found
    }

    MemberIndex++;

  } catch (...) {
    return 0;
  }


  //Get the Pathname of the current device
  //detailData.cbSize = sizeof(SP_INTERFACE_DEVICE_DETAIL_DATA);
  detailData.cbSize = 5;
  DWORD Required = 0;

  try {
    Result = SetupDiGetDeviceInterfaceDetail
              (hDevInfo,&devInfoData,&detailData,256,&Required,NULL);

    if (!Result)
      continue;

  } catch (...) {
    continue;
  }

  //Get Handle for the current device
  try {
    DeviceHandle = CreateFile (detailData.DevicePath,
                               GENERIC_READ|GENERIC_WRITE,
                               FILE_SHARE_READ|FILE_SHARE_WRITE,
                               (LPSECURITY_ATTRIBUTES)NULL,
                               OPEN_EXISTING,
                               0,
                               NULL);

    if (DeviceHandle == INVALID_HANDLE_VALUE)
    {
      CloseHandle(DeviceHandle);
      continue;
    }

  } catch (...) {
    continue;
  }
```

```
    //Read Attributes from the current device
    HIDD_ATTRIBUTES Attributes;
    Attributes.Size = sizeof(Attributes);

    try {
      Result = HidD_GetAttributes (DeviceHandle,&Attributes);
      if (!Result)
      {
        CloseHandle(DeviceHandle);
        continue;
      }


    } catch (...) {
      continue;
  }


  // All information obtained
  // Attributes.VendorID
  // Attributes.ProductID
    // detailData.DevicePath <- Remember for future use

  // easy example can be:
  if ((Attributes.VendorID  == YOUR_VENDOR_ID) &&
      (Attributes.ProductID == YOUR_PRODUCT_ID))
    strcpy(your_devicePath, detailData.DevicePath);

} //end of while
```

1.3 Open device

```
ReadHandle = CreateFile (your_devicePath,
                         GENERIC_READ|GENERIC_WRITE,
                         FILE_SHARE_READ|FILE_SHARE_WRITE,
                         (LPSECURITY_ATTRIBUTES)NULL,
                         OPEN_EXISTING,
                         0,
                         NULL);
```

1.4 Read device

```
DWORD BytesRead = 0;
char Report[INPUT_REPORT_SIZE];
bool Result;

memset(&Report, 0, INPUT_REPORT_SIZE);
Result = ReadFile (ReadHandle,Report,sizeof(Report),&BytesRead,NULL);
```

2 Conclusion

This way of accessing HID devices is used in the MEMS USB Reader software and tested. Additional reference material on this subject can be found in reference [1].

3 References

[1] Jan Axelson, USB COMPLETE, Lakeview Research LLC, Madison, USA, 2001,

ISBN: 0-9650819-5-8

[2] USB specification 2.0, www.usb.org

[3] STMicroelectronics, www.st.com

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZE REPRESENTATIVE OF ST, ST PRODUCTS ARE NOT DESIGNED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS, WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.