GENOVATION

Application Note:
MiniTerm USB Low-Level Programming

Revision B
April 2008

DISCLAIMER

The sample code is provided on an "as is" basis, and Genovation, Inc. expressly disclaims any or all warranties expressed or implied, including without limitation warranties of fitness for a particular purpose. In no event shall Genovation, Inc. be liable for any direct, indirect, incidental consequential damages of any kind whatsoever with respect to the use of the concepts or code.

Genovation, Inc. reserves the right, in its sole discretion and without any obligation, to make improvements to, or correct any error in any portion of the document or code.

TABLE OF CONTENTS

1: Introduction

The MiniTerm is a versatile LCD keypad with the following major attributes:

- A 1 x 16 or a 2 x 16 Liquid Crystal Display with backlight.
- 12 key sealed membrane or 20-key Cherry MX full-travel key switches.
- Internal buzzer.
- Optional card reader (barcode scanner, barcode slot reader, magnetic card slot read, RFID tag reader).
- RS-232 wedge (pass-thru) input.
- Programmable LEDs
- Fully programmable.
- Host connection via RS-232, USB or Ethernet.

It is the last item (host connection) that this application note addresses. More specifically, host connection via USB. Once it has been decided to connect the MiniTerm to the host PC using USB, there are still further options that must be discussed. The table below summarizes the USB host connection choices.

USB 2.0 Class	USB Speed	USB Product ID (Hex)	Comment
HID – Keyboard	Full (12Mbps)	0x0606*	The MiniTerm-K appears to the host as a typical PC keyboard (including Multimedia support as of v5.0). A secondary communication channel with the device is available via USB Feature Reports.
HID – Vendor Defined	Full (12Mbps)	0x0604*	The MiniTerm appears as a device with input and output capabilities that are general in nature. Genovation's MiniTermPro.exe application converts this generic device into a Virtual Com Port (VCP) for the end host application.

* When the MiniTerm is connected to the host in PC Keyboard mode, Genovation refers to this as a MiniTerm-K. Even though any given MiniTerm can operate in any USB or RS-232 host mode, the internal USB Product ID for the –K version is different from the vanilla MiniTerm USB implementation. This is handled automatically by the firmware in the MiniTerm.

In both cases the device connects as a type of HID (Human Interface Device). This means that under Windows, a software developer can access the device at application level without the need to develop privileged driver-level code.

Objective

The objective of this document and accompanying software is to describe techniques that can be used to communicate with the MiniTerm directly from a host application (without employing MiniTermPro). Sample software DLLs are supplied as examples and are intended only to demonstrate programming principles. The DLLs themselves are not intended for professional application.

USB Under The Hood

The single USB cable that connects the MiniTerm to the host disguises the flexibility of the USB interface. The host PC sees the MiniTerm not so much as a device with a single cable, but as a device with a number of data "pipes". Using the terminology of the trade, we say that these pipes are connected to "Endpoints". The endpoints are abbreviated "EP" and numbered, as in EP0 or EP1.

All devices have a "pipe 0" or "Endpoint 0". This pipe or endpoint is used to enumerate/configure/initialize the device (tell the host OS what the device is and what it can and cannot do). This procedure is largely automatic and occurs when the device is plugged in.

Optionally, there are other data pipes or endpoints that are used to move data back and forth with the host. In all cases the MiniTerm has the required endpoint 0 and two additional endpoints. IN endpoints move data from the MiniTerm to the host PC. OUT endpoints move data from the host PC to the MiniTerm.

Host Mode	EP0 Function	EP1 Function	EP2 Function
MiniTerm-K	Direction: IN/OUT - Configuration data. - Host LED state (E.g. NumLock). - Host command set via a special Feature Report.	Direction: IN - Standard PC keyboard key/macro data.	Direction: IN - Multimedia key/macro data.
MiniTerm	Direction: IN/OUT - Configuration data.	Direction: IN - ASCII key/macro data.	Direction: OUT - Host command set.

For the case of the MiniTerm-K, the MiniTerm key press and card reader data is sent to the PC application that has the system focus. The user of the PC determines what that application will be.

For the adventurous programmer, there are ways to attach software to incoming PC keyboard data. Although not the subject of this application note, the two methods are Keyboard Hooks and Raw Input Model. Consult the Microsoft Developer Network for more information.

For the case of the vanilla MiniTerm, MiniTermPro.exe connects to the IN and OUT pipes and converts that data into a Virtual Com Port (VCP). The application connected to the VCP determines what to do with the MiniTerm key press and card reader data. Using techniques discussed later in this application note, you can connect to the pipes yourself and eliminate the need for MiniTermPro.

2: MiniTerm-K (v5.0+)

If you have chosen to use the MiniTerm in its PC Keyboard (–K) configuration, you have made the decision to allow the user key data to be sent unmodified to the application running on the host PC. Therefore, you have also limited the amount of control you have over this data. You can still use the Host Command Set to write data to the LCD or for any other feature you wish to use. Consult our MiniTermPro.PDF for complete description of the command set.

Endpoint Assignments

As previously mentioned, the MiniTerm-K converts key macro data and card reader data (ASCII values 0x7F and below) to standard PC key codes and sends them to the host on EP1. If necessary the MiniTerm-K applies the necessary modifier (Shift or Ctrl).

The MiniTerm-K also converts 24 reserved values (0x80 to 0x97) into multimedia key codes and sends them to the host on EP2. The multimedia key codes supported are shown in the table below.

ASCII	USB	Comment
0x80	0x00B5	Scan Next Track*
0x81	0x00B6	Scan Previous Track*
0x82	0x00B7	Stop*
0x83	0x00CD	Play/Pause*
0x84	0x00E2	Mute*
0x85	0x00E5	Bass Boost*
0x86	0x00E7	Loudness*
0x87	0x00E9	Volume Up*
0x88	0x00EA	Volume Down*
0x89	0x0152	Bass Up*
0x8A	0x0153	Bass Down*
0x8B	0x0154	Treble Up*
0x8C	0x0155	Treble Down*
0x8D	0x0183	Media Select*
0x8E	0x018A	Mail*
0x8F	0x0192	Calculator*
0x90	0x0194	My Computer*
0x91	0x0221	Web Search*
0x92	0x0223	Web Browser/Home*
0x93	0x0224	Web Back*
0x94	0x0225	Web Forward*
0x95	0x0226	Web Stop*
0x96	0x0227	Web Refresh*
0x97	0x022A	Web Favourites*

* Please note that the Multimedia values are only available with firmware v5.00 or higher.

Endpoint 0 is used by the host OS to enumerate the device. As of firmware v5.00, Genovation has also created a "backdoor" method using Feature Reports to allow for a programmer to send host commands to the MiniTerm-K.

The Output Report

In order to send data to the MiniTerm-K, the Win32 API command `HidD_SetFeature()` is used. The report sent to the MiniTerm-K is in the form of a six-byte array.

[0]	[1]	[2]	[3]	[4]	[5]
Report ID	Data Length	Data Payload			
Always 0	0 to 4	Variable	Variable	Variable	Variable

The first byte, Report ID, is always zero. This is simply an identifier for the bus. The second element, Data Length, is set by the programmer to indicate how many data bytes are present in the payload. Finally the Data Payload section is where the host commands (LCD data, etc.) are placed. When more than 4 bytes need to be sent to the MiniTerm-K, multiple calls to `HidD_SetFeature()` are required.

The Input Report

Since the keypad and card reader data are sent as standard PC key codes, the function of the input report is minimal. To receive an input report from the MiniTerm-K, the Win32 API command `HidD_GetFeature()` is used. Again the report is a six-byte array, but the function is different.

[0]	[1]	[2]	[3]	[4]	[5]
Report ID	Firmware Version	Platform Word		Status Word	
Always 0	0x50 or higher	0x09	0x04 or 0x05	Variable	Variable

The first byte, Report ID, is always zero. This is simply an identifier for the bus. The second byte is a short form of the version number of the format MSB**.**LSB. This will always indicate v5.0 or higher since this feature was not implemented prior to v5. As of this writing, the next two bytes simply indicate the LCD style. The value 0x0904 indicates a 1 x 16 LCD while 0x0905 indicates a 2 x 16 LCD. The final two bytes provide some run-time status information for the device. Consult the **Get and Clear Status Word** command from the MiniTermPro User Guide.

A side effect of reading the input report is that the MiniTerm-K will "chirp".

MiniTermK.dll

The programming concepts discussed next have all been captured in a sample DLL called MiniTermK.dll. MiniTermK.dll exports five functions.

```
extern "C" __declspec(dllexport) int MiniTermK_Enumerate(void);
extern "C" __declspec(dllexport) int MiniTermK_Open(void);
extern "C" __declspec(dllexport) int MiniTermK_Close(void);
extern "C" __declspec(dllexport) int MiniTermK_Write(char *szDataToSend);
extern "C" __declspec(dllexport) unsigned char *MiniTermK_Read(void);
```

To use the DLL in your own application you must include the header file MiniTermK.h. In addition you should link your application to MiniTermK.lib and finally MiniTermK.dll must be in the same directory as your application. A sample test application has also been included in the archive. This test application exercises all of the functions in the DLL.

MiniTermK_Enumerate( )

In order to communicate with a USB device it is necessary to get a handle that can be used to access the device via the host OS. In the case of the MiniTerm-K this tends to be problematic since the host claims exclusive use of devices such as keyboards, keypads and mice. To get a handle for this type of device it is necessary to request access without the usual `GENERIC_READ` or `GENERIC_WRITE` attributes. This will prevent us from using `ReadFile()` and `WriteFile()`, but we were not going to use them anyway.

The successful acquisition of a handle to a MiniTerm-K can be used to detect the presence or absence of an attached device.

There is no technique in place to communicate with one particular MiniTerm-K when several are attached to the same PC. Using the built-in EEPROM commands could be used to read and write serial numbers, but it is not a trivial exercise. It is generally recommended that a vanilla MiniTerm is better suited to a multi-MiniTerm environment if you need to communicate specifically with a given device.

In order for the DLL to do it's work, functions from HIDPI.h, HIDSDI.h, HIDUsage.h, and SetupAPI.h are called. The sample DLL project also links with SetupAPI.lib and HID.lib. All of the above are supplied courtesy of Microsoft.

The source code is included in the associated sample archive, MiniTermKDLL.rar. No attempt here will be made to describe the enumeration process line-by-line, but the source code is documented. In short, the function `MiniTermK_Enumerate()` parses all of the available devices looking for one that matches the Vendor ID and Product ID for the MiniTerm-K.

```
extern "C" __declspec(dllexport) int MiniTermK_Enumerate(void);
```

The function `MiniTermK_Enumerate()` returns 1 if a MiniTerm-K is attached and 0 if none are found. If a device is found, the global identifier `PathName[]` is filled in for internal purposes.

MiniTermK_Open( )

Using the `PathName[]` filled in by calling `MiniTermK_Enumerate()`, `MiniTermK_Open()` calls `CreateFile()` to obtain a handle to the MiniTerm-K. This handle is then stored in the global variable `Handle`. This handle is required to move data between the host and the MiniTerm-K. `MiniTermK_Open()` calls `MiniTermK_Enumerate()` so you don't need to call `MiniTermK_Enumerate()` yourself unless you simply want to know if a device is attached or not without opening it.

The return value is 0 on success. On failure, the return value is 1 if already open and 2 if there is no device detected.

```
extern "C" __declspec(dllexport) int MiniTermK_Open(void);
```

MiniTermK_Close( )

This function closes the file handle and then sets the `Handle` variable to zero. It returns 0 on success or 1 if you try and close a handle that is not open.

```
extern "C" __declspec(dllexport) int MiniTermK_Close(void);
```

MiniTermK_Write( )

This function sends a null-terminated string of data to the device. The maximum length of the string is 32 characters (arbitrary). If the length or null-termination requirement creates an issue for you, it is a simple matter to recode this function to accept a length parameter instead. `MiniTermK_Write()` calls `HidD_SetFeature()` in a loop until all of the data is sent.

The function takes one parameter, a pointer to the null-terminated data to write.

If the function proceeds the return value is 0. If the device is not open the function returns 1. If `HidD_SetFeature()` fails the function returns 2.

```
extern "C" __declspec(dllexport) int MiniTermK_Write(char *szDataToSend);
```

Example: `MiniTermK_Write("@CHello World!@B\x20");`

In the example above the following elements are sent to the MiniTerm-K:

- @C –Clears the LCD.
- Hello World! – Prompt string visible to the keypad user.
- @B\x20 – Beep the internal buzzer for $20_{16} * 2 = 40_{16} = 64_{10}$ms.

MiniTermK_Read( )

This function performs a call to `HidD_GetFeature()` in order to retrieve the input report.

If the function succeeds it returns a pointer to the first byte of the input report (the Report ID byte). If it fails the function returns NULL.

```
extern "C" __declspec(dllexport) unsigned char *MiniTermK_Read(void);
```

The data in the input report has been described in an earlier section of this document.

TestMiniTermK.exe

An additional sample application has been provided that exercises the functions in the MiniTermK DLL. TestMiniTermK.exe is a Windows command-line application and the source code for it has been included in the archive as well. It presents the following menu to the user and calls the DLL functions described above.

```
TESTMINITERMK.EXE -- Demo program for MiniTerm-K access via DLL
This program requires a MiniTerm-K attached via USB and uses MiniTermK.DLL

Command menu:
        1. Check for devices
        2. Open MiniTerm-K
        3. Send command string to MiniTerm-K
        4. Read MiniTerm-K status
        5. Close MiniTerm-K
        ?. Show this menu
        X. Exit.
Selection?_
```

3: MiniTerm (v3.0+)

When configured in vanilla form instead of -K, the MiniTerm is considerably more flexible. In this configuration there are separate input and output pipes that provide more functional access to the device since the host OS does not attempt to claim exclusive access to either of them. Since the device still takes advantage of HID class USB drivers, the programmer does not need to use privileged driver-level code. Also, it is much easier to facilitate more than one device attached to the host.

When the provided MiniTermPro.exe application is used, the data flow looks something like the following:

MiniTerm <=> USB Hub <=> OS <=> HID <=> MiniTermPro COM Port <=> User Application

It is the objective of this section to explain how the User Application can access the device(s) using HID level code, thus eliminating the MiniTermPro "middleware".

Endpoint Assignments

To the application programmer the vanilla MiniTerm exhibits these characteristics:

- Feature Report – A 5-byte control report that must be issued to "start" the MiniTerm.
- EP1 (IN) – An 8-byte IN pipe is used to transfer data from the MiniTerm to the PC.
- EP2 (OUT) – An 8-byte OUT pipe is used to transfer data from the PC to the MiniTerm.

Depending on MiniTerm version, the USB connection may be either a low-speed one (prior to v4.00) or a full-speed one (v4.00+). Low-speed endpoints move an IN and/or OUT packet every 10ms, full-speed endpoints move a packet every 2ms. This is a function of the internal architecture of the MiniTerm and associated USB descriptors.

If you are interested, examining the descriptors for the MiniTerm you are working with will tell you what configuration the MiniTerm is requesting from the host. There are many tools available that will report the descriptors of any connected USB device. For example: USBview.exe, UVCView.x86.exe, and many others.

MiniTerms of various versions will co-exist successfully on the same host.

Unlike the MiniTerm-K configuration, full read and write access to the vanilla MiniTerm is obtained as demonstrated in the sections that follow.

The Output Report

In order to send data to the MiniTerm, the Win32 API command `WriteFile()` is used. The report sent to the MiniTerm is in the form of an eight-byte array.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
Report ID	Data Length	Data Payload						
Always 0	0 to 7	Variable	Variable	Variable	Variable	Variable	Variable	Variable

The first byte, Report ID, is always zero. This is simply an identifier for the bus. The second element, Data Length, is set by the programmer to indicate how many data bytes are present in the payload. Finally the Data Payload section is where the host commands (LCD data, etc.) are placed.

The Input Report

In order to receive data from the MiniTerm, the Win32 API command `ReadFile()` is used. Again the report is an eight-byte array.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
Report ID	Data Length	Data Payload						
Always 0	0 to 7	Variable	Variable	Variable	Variable	Variable	Variable	Variable

The first byte, Report ID, is always zero. This is simply an identifier for the bus. The Data Length element indicates the amount of returned key/card data (0 to 7 bytes). The remaining bytes contain data, if any. Data elements that are "unused" may contain any random value.

MiniTerm.dll

The programming concepts discussed next have all been captured in a sample DLL called MiniTerm.dll. MiniTerm.dll exports six functions.

```
extern "C" __declspec(dllexport) int MiniTerm_Enumerate(void);
extern "C" __declspec(dllexport) int MiniTerm_Open(void);
extern "C" __declspec(dllexport) int MiniTerm_Close(void);

extern "C" __declspec(dllexport) int MiniTerm_KbHit(void);
extern "C" __declspec(dllexport) unsigned char MiniTerm_Getch(void);

extern "C" __declspec(dllexport) int MiniTerm_Write(char *szDataToSend);
```

To use the DLL in your own application you must include the header file MiniTerm.h. In addition you should link your application to MiniTerm.lib and finally MiniTerm.dll must be in the same directory as your application. A sample test application has also been included in the archive. This test application exercises all of the functions in the DLL.

MiniTerm_Enumerate( )

In order to communicate with a USB device it is necessary to get a handle that can be used to access the device via the host OS. This function looks for the presence of a MiniTerm and saves it's identifier in the variable szPathName[ ]. It can easily extended to look for N MiniTerms by creating an array of pathnames and looping until SetupDiEnumDeviceInterfaces() returns 0.

The successful acquisition of a handle to a MiniTerm can be used to detect the presence or absence of an attached device.

In order for the DLL to do it's work, functions from HIDPI.h, HIDSDI.h, HIDUsage.h, and SetupAPI.h are called. The sample DLL project also links with SetupAPI.lib and HID.lib. At run time it uses SetupAPI.dll and HID.dll. All of the above are supplied courtesy of Microsoft. Depending on compiler, you may be able to link directly with these modules or you may have to load the DLLs at run-time.

The source code is included in the associated sample archive, MiniTermDLL.rar. No attempt here will be made to describe the enumeration process line-by-line, but the source code is documented. In short, the function `MiniTerm_Enumerate()` parses all of the available devices looking for one that matches the Vendor ID and Product ID for the MiniTerm.

```
extern "C" __declspec(dllexport) int MiniTerm_Enumerate(void);
```

The function `MiniTerm_Enumerate()` returns 1 if a MiniTerm is attached and 0 if none are found. If a device is found, the global identifier `szPathName[]` is filled in for internal purposes.

MiniTerm_Open( )

Once you have a file name path to the MiniTerm, you can use it in a classic CreateFile to get a handle to the MiniTerm. Using the `PathName[]` filled in by calling `MiniTerm_Enumerate()`, `MiniTerm_Open()` calls `CreateFile()` to obtain a handle to the MiniTerm. This handle is then stored in the global variable `hMiniTerm`. This handle is required to move data between the host and the

MiniTerm. `MiniTerm_Open()` calls `MiniTerm_Enumerate()` so you don't need to call `MiniTerm_Enumerate()` yourself unless you simply want to know if a device is attached or not without opening it.

The return value is 0 on success. On failure, the return value is 1 if already open and 2 if there is no device detected. The value 3 is returned if there is a problem obtaining a handle.

```
extern "C" __declspec(dllexport) int MiniTerm_Open(void);
```

In order for a vanilla MiniTerm to start moving data, the device needs to receive a feature report from the host. Therefore, `MiniTerm_Open()` calls `HidD_SetFeature()` to perform this task. As of this writing, the MiniTerm doesn't particularly care about the data content, just that the report is correctly received.

To prevent receive data loss, and to prevent program hanging while waiting for input data, a simple thread is used to collect data from the MiniTerm. This thread continually requests input reports using `ReadFile()`. A small circular buffer is used to store any data received from the MiniTerm. In the interests of introducing the concept of mutual exclusion, a simple critical section is used to orchestrate access to the circular receive buffer. A more professional approach would be to also open the MiniTerm in overlapped mode.

MiniTerm_KbHit( )

In support of the provided command-line test application, the DLL exports a function that indicates whether or not data has been received from the MiniTerm. Received data from the MiniTerm is automatically enqueued into a circular queue by the receive thread. The function `MiniTerm_KbHit()` checks to see if there is any data in the receive queue.

```
extern "C" __declspec(dllexport) int MiniTerm_KbHit(void);
```

If there is data waiting then the function returns non-zero. If the receive queue is empty, the function returns zero.

MiniTerm_Getch( )

The DLL exports a function to dequeue data from the receive buffer. Before calling this function it is absolutely imperative that the program first test for data availability using `MiniTerm_KbHit()`. The function `MiniTerm_Getch()` returns the oldest byte of data in the receive queue.

```
extern "C" __declspec(dllexport) unsigned char MiniTerm_Getch(void);
```

This function should be called repeatedly as long as `MiniTerm_KbHit()` returns non-zero. Exclusive access to the receive queue is controlled by the critical section.

MiniTerm_Write( )

In order to write data to the MiniTerm it is necessary to issue output reports using `WriteFile()`. Each output report can contain up to 7 data bytes.

This function `MiniTerm_Write()` sends a null-terminated string of data to the device (less the null). The maximum length of the string is 32 characters (arbitrary). If the length or null-termination requirement creates an issue for you, it is a simple matter to recode this function to accept a length parameter instead. `MiniTerm_Write()` calls `WriteFile()` in a loop until all of the data is sent.

The function takes one parameter, a pointer to the null-terminated data to write.

```
extern "C" __declspec(dllexport) int MiniTerm_Write(char *szDataToSend);
```

If the function proceeds the return value is 0. If the device is not open the function returns 1. If `WriteFile()` fails the function returns 2.

Example: `MiniTerm_Write("@CHello World!@B0");`

In the example above the following elements are sent to the MiniTerm:

- @C –Clears the LCD.
- Hello World! – Prompt string visible to the keypad user.
- @B0 – Beep the internal buzzer for $30_{16} * 2 = 60_{16} = 96_{10}$ms.

Test9xxDLL.exe

An additional sample application has been provided that exercises the functions in the MiniTerm DLL. Test9xxDLL.exe is a Windows command-line application and the source code for it has been included in the archive as well. It presents the following menu to the user and calls the DLL functions described above.

```
Test9xxDLL.exe -- Demo program for Genovation Model 9xx MiniTerm.
This program requires a MiniTerm attached via USB and uses MiniTerm.DLL

Command menu:
        1. Check for devices
        2. Open MiniTerm
        3. Close MiniTerm
        4. Check for input data
        5. Receive input data
        6. Send command string to MiniTerm
        ?. Show this menu
        X. Exit.
Selection?_
```

Further Reading

Genovation has some related information in another application note entitled, “Micropad 623 Serial Keypad Developer Manual”. It can be found at:

http://www.genovation.com/files/623DevManual.pdf

The MiniTerm also has an advanced “Screen Edit Mode” that provides for a series of predetermined prompts in order to gather multiple fields of input from a user, but without the need to program the interaction on the host. Please see:

http://www.genovation.com/files/MiniTermProAdvanced.pdf