GraphicUSB Devkit V1.01
User Manual

MQP Electronics Ltd
Unit 2, Park Road Centre
Malmesbury
Wiltshire SN16 0BX
United Kingdom
e-mail: sales@mqp.com Website: www.mqp.com

Contents

1 GETTING STARTED

1.1 Introduction

This manual assumes that you are already familiar with the Packet-Master analyser and normal use of the GraphicUSB application. Refer to the Packet-Master User Manual in case of any doubt.

Whilst GraphicUSB can display and export data in a wide variety of formats, there will always be cases when the developers would like access to the capture file from their own programs.

A possible reason is when it is necessary to perform an extremely large capture, in order to see a packet which occurs only infrequently. A user-written program could analyse the capture file using the developer's unique knowledge of the vendor class device. It could output the significant information to a small text file.

GraphicUSB is equipped with functionality which allows it to perform a capture from a command line, or to perform a capture and save the file without displaying it, thus eliminating the time needed to index the file.

The optional GraphicUSB Devkit contains a Dynamic Link Library file which user programs can link to. This DLL allows access to each event in turn, contained in the .mqu capture file saved by GraphicUSB.

An example "C" command line program illustrates how to make use of the DLL.

The Devkit is compatible with any of Packet-Master Analyser series.

1.2 Installing the Software and DLL File

1.2.1 Install the Software from CD

- Insert the GraphicUSB Installation disk into the CD drive.
- On the 'Choose Components' page, ensure that 'DevKit' is checked.
- Continue with install. The Devkit files will be found in their own folder called 'GraphicUsbDevKit'.
- The access DLL file will be placed in the same Program Files folder as the GraphicUSB application.

1.2.2 Files supplied

The key files provided are:

mquaccess.dll

This Dynamic Link Library (DLL) file contains the functions required to access the .mqu capture file. This is placed in the same Program Files folder as the GraphicUSB application. Consequently an absolute path to this location should be specified when linking to it.

mquaccess.h

This include file contains the declarations for the functions required to access the .mqu capture file. **mquaccess.h** should be included in your .c file to allow access to the dll at run time.

outtext.cpp

This is a sample C program which can be used as the basis of your own custom program. The sample is a command line program which performs the following:

- It starts by linking to the dll, checking that it is the right version.
- It opens a new text file for writing its results to.
- It instructs the dll to open a specified capture (.mqu) file.

- It requests a set of information about the file. A key item of information is whether the capture file was created using an analyser which was registered for use with the DevKit. If it was not, further analysis will not be possible.
- It requests each event in turn from the dll, outputting appropriate information to the text file, as required.
- It instructs the dll to close the .mqu file.
- It closes the text file.
- Finally it releases the dll.

capture.mqu

This is a sample capture file created by GraphicUSB, using an analyser registered for DevKit.

The folder also contains files created by Visual Studio 6, which were used to compile this project.

In the *Release* folder is:

outtext.exe

A compiled version of the program.

DevKit Test Release

A desktop link icon, for testing purposes.

1.3 Trying it Out

Documentation is provided mainly by comments in the **mquaccess.h** file, to ensure that the documentation used refers to the current version.

As delivered, the **outtext** project is set up as a Visual Studio 6 workspace and a 'C' project, although of course you may use **mquaccess.dll** in any way that suits your programming tools.

As delivered a desktop shortcut, **DevKit Test Release,** is provided to open a command line window and set the release folder as current. Double click this, then at the command line prompt, type:

```
outtext  capture.mqu  text.txt
```

The capture file will be processed and the file text.txt will be created.

Type:

```
Type  text.txt
```

for a brief flavour of the content.

1.4 Points to remember

- Unless you move the DLL, or installed it elsewhere, you should use the path name to the DLL:

```
\\Program Files\\MQP Electronics\\GraphicUSB\\mquaccess.dll
```

  Before you can use the DLL, you must link to it, using **LoadLibrary()**. At the end of your program you should release the DLL, using **FreeLibrary()**.

- The header file mqpaccess.h must match the version number of the mqpaccess.dll file. To ensure this your program must check the DLL version using **MQUACCESS_DllGetVersion()** before using any other function.

- The capture file must have been generated using an analyser registered for the DevKit option. You can check this by using **MQUACCESS_Open(const LPCTSTR filepath)** followed by **MQUACCESS_GetFileInfo(FileInfo* pFileInfo)** and then checking **fileInfo.bDevkitRegistered**.

2 FAST WAYS TO PRODUCE A CAPTURE FILE

2.1 From Command Line

It is possible to control GraphicUSB from another application using a command-line syntax. This allows another application to perform a capture without GraphicUSB becoming visible. For the sake of simplicity, the following examples show the commands being issued by use of the older WinExec() function. You may wish to use a more recent function such as CreateProcess() or the .NET function Process.Start(). The strings are all supposed to be on one line but will be shown split in this document.

The filename is shown in the examples without a full path. In this case the file will be saved in the Application Data folder for GraphicUSB. You can also specify a full pathname in order to save the file in a folder of your choice.

The available commands, which make use of the switches '-c0', '-c1' and '-c2', are as follows:

2.1.1 Start Capture

```
::WinExec("c:\\Program Files\\MQP
Electronics\\GraphicUSB\\GraphicUSB -c1 TestCapt.mqu",
SW_HIDE);
```

This will start GraphicUSB running, define TestCapt.mqu as the location to save the capture when complete, and start the capture. If GraphicUSB is already running visibly, it will become invisible and start the capture.

2.1.2 **Restart Capture**

```
::WinExec("c:\\Program Files\\MQP
Electronics\\GraphicUSB\\GraphicUSB -c2", SW_HIDE);
```

This assumes that a capture is in progress, (else an error message is displayed). It will abandon the capture in progress and start it again.

2.1.3 **Stop Capture**

```
::WinExec("c:\\Program Files\\MQP
Electronics\\GraphicUSB\\GraphicUSB -c0", SW_HIDE);
```

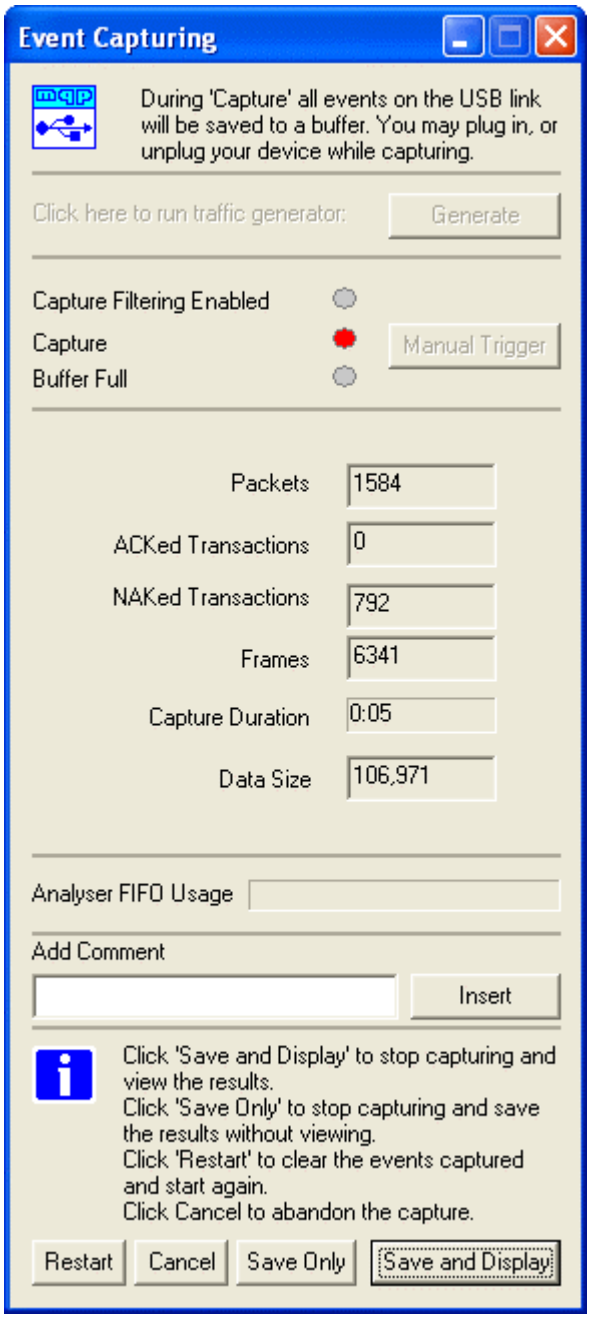
This assumes that a capture is in progress, (else an error message is displayed). It will stop the capture in progress and save it to the file named in the Start Capture command.

2.1.4 **Display File**

```
::WinExec("c:\\Program Files\\MQP
Electronics\\GraphicUSB\\GraphicUSB  TestCapt.mqu ",
SW_SHOW);
```

This will display the captured file specified, in a visible instance of GraphicUSB.

2.2 From GraphicUSB Application without Displaying



When saving the capture file, you can click ‘Save Only’, and the file will be saved without the need to perform the normal display parsing. On very large files this can result in a considerable time saving.

Appendix 1 – The include file mquaccess.h

```
#ifndef _MQUACCESSDLL_H_
#define _MQUACCESSDLL_H_


// MQUACCESS_EXPORTS is defined when building the dll
// it should not be defined when calling the dll

#define MQUACCESS_DLL_VERSION 0x0101

// **** defines used in Event struct

// eventType possible values

#define EVENTNONE 0
#define EVENTPACKET 1
#define EVENTPLUGGED 2
#define EVENTUNPLUGGED 3
#define EVENTRESETSTART 4
#define EVENTRESETEND 5
#define EVENTSUSPEND 6
#define EVENTRESUMESTART 7
#define EVENTRESUMEEND 8
#define EVENTKEEPALIVE 9
#define EVENTBOTHLINESHIGH 10
#define EVENTSPURIOUSDATA 11
#define EVENTDATALINEHIGH 12
#define EVENTDATALINELOW 13
#define EVENTVBUSON 14
#define EVENTVBUSOFF 15
#define EVENTCHIRPSTART 16
#define EVENTCHIRPEND 17
#define EVENTHSRESETSTART 18
#define EVENTCOMMENT 19
#define EVENTTRIGGERSTART 20
#define EVENTTRIGGERSTOP 21
#define EVENTENDOFFILE 255

// PID values

#define     PIDTOKOUT         0xe1
#define     PIDTOKIN          0x69
#define     PIDTOKSOF         0xa5
#define     PIDTOKSETUP       0x2d
#define     PIDTOKDATA0       0xc3
#define     PIDTOKDATA1       0x4b
#define     PIDTOKDATA2       0x87
#define     PIDTOKMDATA       0x0f
#define     PIDTOKACK         0xd2
#define     PIDTOKNAK         0x5a
#define     PIDTOKSTALL       0x1e
```



```
#define     PIDTOKNYET        0x96
#define     PIDTOKPREORERR    0x3c
#define     PIDTOKSPLIT       0x78
#define     PIDTOKPING        0xb4
#define     PIDTOKRESERVED    0xf0

// usbSpeed possible values

#define USBSPEED_LS 0
#define USBSPEED_FS 1
#define USBSPEED_HS 2

#define DATABUFFERSIZE 1027

// structures used by mquaccess.dll

// basic information about the opened capture (.mqu) file
struct      FileInfo
{
      DWORD64     captureDuration;
      DWORD analyserSerNum;               // serial number of analyser
used, display as decimal
      WORD  fileVersion;
      WORD  swVersion;                    // version of GraphicUSB used
for capture
      WORD  modifyingSwVersion;
      WORD  analyserPid;
      WORD  analyserDriverVersion;
      WORD  analyserFwVersion;
      WORD  analyserPcbVersion;
      BOOL  bFileOpen;                    // 0=not open, 1 = open
      BOOL  bDevkitRegistered;            // 0=not registered,
1=registered
      DWORD fileSize;
      WORD  clocksPerUs;                  // 60 or 48 clocks per
microsecond (see timestamps)
};

// information describing a particular event in the file
struct      Event
{
      DWORD fileIndex;                    // could be used for progress
bar with FileInfo.fileSize
      DWORD64     timeStamp;                    // in clocks (0 means
file had syntax error)
      BYTE  eventType;                    // see list above
      int         numDataBytes;                 // if it's a packet
      BYTE  data[DATABUFFERSIZE];   // includes PID and CRC
      int         usbSpeed;                     // USBSPEED_LS,
USBSPEED_FS, USBSPEED_HS
      BOOL  bCrc16Error;                  // data0/1/2/mdata Packet Only
      BOOL  bCrc5Error;                   // in/out/setup/sof/split
Packet Only
      BOOL  bByteError;                   // Packet Only
```



```
      BOOL  bBitStuffError;               // Packet Only
      BOOL  bBothLinesHighError;    // Packet Only
      BOOL  bPacketError;                 // Packet Only. e.g. no EOP

      BYTE  nEopLengthInClks;       // Ls/Fs Packets, Resume End and
Keep Alive
      BOOL  bChirpJ;                      // Chirp Start only
      BOOL  bDn;                          // Data Line High/Low Only

      BOOL  bVbusCurrentState;            // Data Line High/Low or VBUS
High/Low Only
      BOOL  bDpCurrentState;        // Data Line High/Low or VBUS
High/Low Only
      BOOL  bDnCurrentState;        // Data Line High/Low or VBUS
High/Low Only

      BOOL  bSpuriousBytesFound;    // file contains non-parsed record
};



// exported functions

// ********** MQUACCESS_DllGetVersion();
// Returns a WORD containing the dll version number
// it is important to check this before using the dll further

// ********** MQUACCESS_Open(const LPCTSTR filepath);
// Opens the named mqu file in the dll
// must be done before using MQUACCESS_GetEvent or
MQUACCESS_GetFileInfo!
// returns TRUE for successful opening

// ********** MQUACCESS_GetFileInfo(FileInfo* pFileInfo);
// Fills in the FileInfo structure for the mqu file
// returns FALSE if file could not be accessed (e.g. if it hadn't been
opened)

// ********** MQUACCESS_GetEvent(Event* pEvent);
// Fills in the Event structure for each successive event in the mqu
file
// one event returned for each call
// returns FALSE for end of file
// or if file captured on analyser not registered for DevKit

// ********** MQUACCESS_Close();
// Closes the named mqu file in the dll
// returns TRUE for successful closing



#ifdef MQUACCESS_EXPORTS

extern "C" __declspec(dllexport) WORD     MQUACCESS_DllGetVersion();
```



```
extern "C" __declspec(dllexport) BOOL     MQUACCESS_Open(const LPCTSTR
filepath);
extern "C" __declspec(dllexport) BOOL
      MQUACCESS_GetFileInfo(FileInfo* pFileInfo);
extern "C" __declspec(dllexport) BOOL     MQUACCESS_GetEvent(Event*
pEvent);
extern "C" __declspec(dllexport) BOOL     MQUACCESS_Close();

#else

typedef WORD      (*MQUACCESS_DllGetVersion) ();
typedef BOOL      (*MQUACCESS_Open) (const LPCTSTR filepath);
typedef BOOL      (*MQUACCESS_Close) ();
typedef BOOL      (*MQUACCESS_GetFileInfo) (FileInfo* pFileInfo);
typedef BOOL      (*MQUACCESS_GetEvent) (Event* pEvent);
typedef BOOL      (*MQUACCESS_GetByte) ();

#endif

#endif
```

