pocketBERT Company

GUI and API Application Note

Status: Preliminary, version 0.6

Contents

1. Instructions for installing and running the pocketBERT GUI application. 2
2. How to manually install the pocketBERT GUI: 2
3. Instructions for installing and running the pocketBERT API and the API Test application. 4
4. API Development Information: 4
5. API Functions: 5
5.1. pocketBERT API functions 5
5.2. pocketBERT API functions 5
5.3. User BERT pocketBERT API functions (High Level Functions) 5
6. MICROSOFT FOUNDATION CLASS LIBRARY: pBERT_API_Test 11
7. API Test Application 13

1. Instructions for installing and running the pocketBERT GUI application.

The GUI, as with all the software and documentation, is supplied on a CD or a ZIP file. If supplied on a CD, the instructions that follow should match the structure of the CD. If supplied in a ZIP file, the ZIP file should be un-Zipped to a directory. This directory should then match the instructions.

1.1. GUI Installation:

The pocketBERT GUI must be installed on the PC before it can be run. It is not generally possible to run the GUI without installing it first, i.e. the GUI cannot be executed from the directory contained here.

To install the GUI, go to the directory "GUI Install" and run "SETUP.EXE". Follow the instructions to complete the installation.

It is best to select the defaults given in the instructions. Once the installation is complete, an icon will be placed on the Desktop. Double click on this icon to start the GUI.

Note: In order for the GUI to successfully run, the pocketBERT box must be connected to the PC and the power turned on. The first time the pocketBERT box is connected to a PC, there is a short installation that the system should automatically go through. The pocketBERT box is a Human Interface Device (HID) USB device, and as such, it uses the system HID USB drivers. There are no special or additional USB drivers required for operation.

Note: If the GUI does not install (error messages display) or run correctly (error messages on loading), it may be because the ActiveX components did not install on the system. See the file "How to Manually Install the pocketBERT GUI" to try to install the GUI manually. ("SETUP.EXE" is a 16-bit application and some 64-bit systems will not execute 16-bit applications, requiring a manual installation.)

1.2. GUI Execution:

Double click on the icon for the pocketBERT GUI placed on the desktop.

2. How to manually install the pocketBERT GUI:

There are certain PC systems on which the GUI will not install. Most of these systems seem to be 64-bit operating systems, although there are other systems where there is trouble as well. For the 64-bit systems that fail to install, it seems that 16-bit applications will not run, i.e., the 64-bit systems will not run 16-bit applications. Unfortunately, “SETUP.EXE” is a 16-bit application.

When the GUI will not install normally, it can be installed manually. Most of the GUI files can just be copied to a directory and executed from there. There are however, two ActiveX components that are

required to be installed and registered on the system. The following steps should aid in manually installing the GUI. Unfortunately, the steps vary a little depending on the operating system.

2.1. XP (32-bit):

- Create the directory "C:\Program Files\pocketBERT\pocketBERT".
- Copy the files "pocketBERT.exe", "pocketBERT_Dll.dll", "pocketBERT.stu" and "AtUsbHid.dll" to the directory created.
- Copy the files "Grid32.ocx" and "LED_12_17_01.ocx" to "C:\Windows\System32"
- Bring up the Command Prompt:
- Right click on "Start->Run".
- In the window, type "cmd" and click on "OK".
- The Command Prompt window should appear.
- Type "C:" in the Command Prompt and then the "Return" (or "Enter") key on the keyboard.
- Type "cd C:\Windows\system32" and then the "Return" key on the keyboard.
- Type "regsvr32 grid32.ocx" and then the "Return" key on the keyboard.
- There should be some sort of window that shows it was successful.
- Type "regsvr32 LED_12_17_01.ocx" and then the "Return" key on the keyboard.
- There should be some sort of window that shows it was successful.
- Type "exit" and then the "Return" key on the keyboard.

2.2. XP (64-bit):

Pretty much the same, except that "C:\Windows\System32" may be different, such as "C:\Windows\Syswow64", and "regsvr32" may be "regsvr64". Not fully evaluated yet – please contact support@pocketbert.com for further info.

2.3. Vista (32-bit):

Under evaluation - please contact support@pocketbert.com for further info.

2.4. Vista (64-bit):

Under evaluation - please contact support@pocketbert.com for further info.

2.5. Windows 7:

Under evaluation - please contact support@pocketbert.com for further info.

3. Instructions for installing and running the pocketBERT API and the API Test application.

3.1. API Installation:

There is no installation required for the API. The files for the API are accessed directly by the application using the API.

3.2. API Test Application Installation:

There is no installation required for the API Test Application. The API Test Application may be run directly from the directory "pocketBERTAPI\pBERT_API_Test_LoadLib\Release" by double clicking on "pBERT_API_Test_LoadLib.exe".

It is suggested, however, that the entire directory "pocketBERTAPI" be copied to another directory and the program "pBERT_API_Test_LoadLib.exe" be executed from there, so as to not change or corrupt any of the original files.

Note: The directory structure for the "pocketBERTAPI" directory must be maintained for the API to run correctly. The API application has been coded to access the API DLL that resides in specific, relative directories from the EXE.

Note: It does not matter if the GUI or the API is started first. They both must be started, however, for the API to function properly. The API requires that the GUI is also running to operate correctly.

3.3. API Test Application Execution:

The API Test application is run by double clicking on the "pBERT_API_Test_LoadLib.exe" file in the "pocketBERTAPI\pBERT_API_Test_LoadLib\Release" sub-directory.

4. API Development Information:

The API and the API Test application are supplied in two directories:

- "pocketBERTAPI\API"
- "pocketBERTAPI\pBERT_API_Test_LoadLib"

The first directory, "pocketBERTAPI\API", contains the API. It contains the .DLL file that is the API executable, the .H file that defines the entry points and other definitions for the API and can be used to compile a user application that wants to access the API, and the .LIB file that allows a user application to link to the API.

This API is written in Microsoft C/C++, version 6.0 compiler, and is a 32-bit, C application.

The second directory, "pocketBERTAPI\pBERT_API_Test_LoadLib", contains full source code and executables for a test application that uses the API. See the "pocketBERTAPI\pBERT_API_Test_LoadLib\Release" directory for the test application executable.

This application is written in Microsoft C/C++, version 6.0 compiler, and is a 32-bit, C++ application.

Note: The test API application accesses the API in the "pocketBERTAPI\API" directory, and as such, the relative directory structure to this directory must be maintained for the application to run successfully.

5. API Functions:

The API Test Application accesses the following functions. Therefore, the source code for the API Test Application should serve as a guide on when and how to execute each of the API function.

Listed here are the individual functions.

5.1.pocketBERT API functions

The following function can be executed at any time, whether the connection to the API is open or not. It returns the version information for the API.

MODULEAPI int pBERTAPI_Version(void);

The format for the 32-bit integer returned is (in hex) XXXXYYZZ, where “XXXX” are not used and should be return as 0’s, “YY” is the major version number and is typically “01”, and “ZZ” is the minor version number. An example would be 00000105 returned would indicate version 1.05, and 000002FF would indicate version 2.255.

5.2.pocketBERT API functions

The following functions open and close the connection between a user application and the API. The connection must be open in order for the user application to issue commands. The connection can be closed before the user application is terminated.

MODULEAPI int pBERTAPI_Open(void);

MODULEAPI void pBERTAPI_Close(void);

5.3.User BERT pocketBERT API functions (High Level Functions)

The following function returns a number of values related to BER mode.

MODULEAPI int pBERTAPI_Get_BER_Data(int *mode, unsigned int *seq, int *bermode, long double *berratio, long double *bercalc, long double *datarate, long double *databits, int *errcur, int *errhist, char *str);

The following function returns the number of data bits processed in the current period. Even though this value is an integer value, it is returned as a “double” to allow returning a larger value than the 32-bits in an integer.

MODULEAPI double AccumulationPeriodBits(void);

MODULEAPI double EDAccumulatedBitCount(void);

The following functions return the number of data bits processed in the current period. Even though this value is an integer value, it is returned as a “double” to allow returning a larger value than the 32-bits in an integer.

MODULEAPI double AccumulationPeriodErrors(void);

MODULEAPI double EDAccumulatedErrorCount(void);

The following function returns the length of time in millisecs that the period has been running. Even though this value is an integer value, it is returned as a “double” to allow returning a larger value than the 32-bits in an integer.

MODULEAPI double AccumulationPeriodTime(void);

MODULEAPI double GatePeriodElapsed(void);

The following function returns the error ratio calculated from the number of errors and the number of data bits.

MODULEAPI double EDAccumulatedErrorRatio(void);

The following function returns the name of the log file currently being used by the GUI. The name of the file is always “BERT_Error_Log.txt”, as the GUI always writes to the same file. The file name returned includes the path and file name.

If the user application needs to change log file names, then the user application will need to locate the file and copy or rename the file.

MODULEAPI char far * EDLogFile(void);

The following function will reset the pocketBERT box, which may reset the mode and settings to states programmed in the box.

MODULEAPI double RecallSetup(void);

The following function reads whether the data and error logging is currently enabled or disabled, or enables or disables the data and error logging in the GUI.

MODULEAPI BOOL EDLoggingEnabled(int get_set, int on_off);

The "ret_set" parameter tells the function if this is a request for the current setting, or this is an attempt to set the parameter.

get_set	Description
1 (API_GET)	Gets the current setting and returns the value in the function return: 0 indicates logging is off, 1 indicates logging is on.
2 (API_SET)	Set the value to the setting in the "on_off" parameter: 0 sets logging off, 1 sets logging on. The function return should reflect the new setting.

The following command allows the user application to set certain parameters in the GUI.

MODULEAPI int pBERTAPI_Command (int func, int get_set, char *param, DWORD max_wait_time, char *result);

This function returns a "int" as its function return. The following table describes the return value:

Value	Command	Notes
1	Command succeeded	Data available in "result"
0	Command failed	Invalid command, command not supported
-1	Command failed	Previous command still in progress

-2	Command failed	Invalid command, command format bad
-100	Command failed	Time out waiting for API to read command. No response from API
-101	Command failed	Time out waiting for API to process command. No response from API
-110	Command Failed	API returned an invalid count while waiting for API to read command
-111	Command failed	API returned an invalid count while waiting for API to process command

The parameters passed to this function are listed in the following table:

func	get_set	param	Description
1 (API_DATA_BIT_COUNT)	1 (API_GET)	N/A	Same as the "**AccumulationPeriodBits" function.**
	2 (API_SET)	N/A	Not a valid command
2 (API_ERROR_BIT_COUNT)	1 (API_GET)	N/A	Same as the "**AccumulationPeriodErrors" function.**
	2 (API_SET)	N/A	Not a valid command
3 (API_TIME)	1 (API_GET)	N/A	Same as the "**AccumulationPeriodTime" function.**
	2 (API_SET)	N/A	Not a valid command
4 (API_ED_ERROR_RATIO)	1 (API_GET)	N/A	Same as the "**EDAccumulatedErrorRatio" function.**
	2 (API_SET)	N/A	Not a valid command
5 (API_LOG_FILE)	1 (API_GET)	N/A	Same as the "**EDLogFile" function.**
	2 (API_SET)	N/A	Not a valid command
6 (API_RECALL_SETUP)	1 (API_GET)	N/A	Not a valid command
	2 (API_SET)	N/A	Same as the "**RecallSetup" function.**
7 (API_LOG_FILE_ON_OFF)	1 (API_GET)	N/A	Returns the log file oOn (1) or Off (0) state.
	2 (API_SET)	"0" sets log file Off, "1" set s the log file On	Turns the Logging either On or Off
101 (API_BERT_MODE)	1 (API_GET)	N/A	Returns the current mode for the GUI in the "result" string:
	2 (API_SET)	N/A	Sets the GUI mode to BERT Mode. This is the same as clicking the "BERT Mode" button on the GUI.

102 (API_CDR_MODE)	1 (API_GET)	N/A	Returns the current mode for the GUI in the "result" string:
	2 (API_SET)	N/A	Sets the GUI mode to CDR Mode. . This is the same as clicking the "CDR Mode" button on the GUI.
103 (API_SYNTH_MODE)	1 (API_GET)	N/A	Returns the current mode for the GUI in the "result" string:
	2 (API_SET)	N/A	Sets the GUI mode to Synthesis Mode, . This is the same as clicking the "Clock Synthesizer Mode" button on the GUI.
104 (API_CLEAR_BERT)	1 (API_GET)	N/A	Does nothing.
	2 (API_SET)	N/A	This clears and resets the BERT mode parameters for data bits, errors, time period, etc. This is the same as clicking the "Clear BERT" button on the GUI.

Note: For all commands where the "param" value is listed as "N/A", a value of 0 is recommended.

6. MICROSOFT FOUNDATION CLASS LIBRARY: pBERT_API_Test

AppWizard has created this pBERT_API_Test application for you. This application not only demonstrates the basics of using the Microsoft Foundation classes but is also a starting point for writing your application.

This file contains a summary of what you will find in each of the files that make up your pBERT_API_Test application.

6.1. pBERT_API_Test.dsp

This file (the project file) contains information at the project level and is used to build a single project or subproject. Other users can share the project (.dsp) file, but they should export the make files locally.

6.2. pBERT_API_Test.h

This is the main header file for the application. It includes other project specific headers (including Resource.h) and declares the CPBERT_API_TestApp application class.

Note: This above file contains the source include code for the test application

6.3. pBERT_API_Test.cpp

This is the main application source file that contains the application class CPBERT_API_TestApp.

Note: This above file contains the source executable code for the test application

6.4. pBERT_API_Test.rc

This is a listing of all of the Microsoft Windows resources that the program uses. It includes the icons, bitmaps, and cursors that are stored in the RES subdirectory. This file can be directly edited in Microsoft Visual C++.

6.5. pBERT_API_Test.clw

This file contains information used by ClassWizard to edit existing classes or add new classes. ClassWizard also uses this file to store information needed to create and edit message maps and dialog data maps and to create prototype member functions.

6.6. res\pBERT_API_Test.ico

This is an icon file, which is used as the application's icon. This icon is included by the main resource file pBERT_API_Test.rc.

6.7. res\pBERT_API_Test.rc2

This file contains resources that are not edited by Microsoft Visual C++. You should place all resources not editable by the resource editor in this file.

AppWizard creates one dialog class:

6.8. pBERT_API_TestDlg.h, pBERT_API_TestDlg.cpp - the dialog

These files contain your CPBERT_API_TestDlg class. This class defines the behavior of your application's main dialog. The dialog's template is in pBERT_API_Test.rc, which can be edited in Microsoft Visual C++.

Other standard files:

6.9. StdAfx.h, StdAfx.cpp

These files are used to build a precompiled header (PCH) file named pBERT_API_Test.pch and a precompiled types file named StdAfx.obj.

6.10. Resource.h

This is the standard header file, which defines new resource IDs. Microsoft Visual C++ reads and updates this file.

Other notes:

AppWizard uses "TODO:" to indicate parts of the source code you should add to or customize.

If your application uses MFC in a shared DLL, and your application is in a language other than the operating system's current language, you will need to copy the corresponding localized resources MFC42XXX.DLL from the Microsoft Visual C++ CD-ROM onto the system or system32 directory, and rename it to be MFCLOC.DLL. ("XXX" stands for the language abbreviation.

For example, MFC42DEU.DLL contains resources translated to German.) If you don't do this, some of the UI elements of your application will remain in the language of the operating system.

7. API Test Application

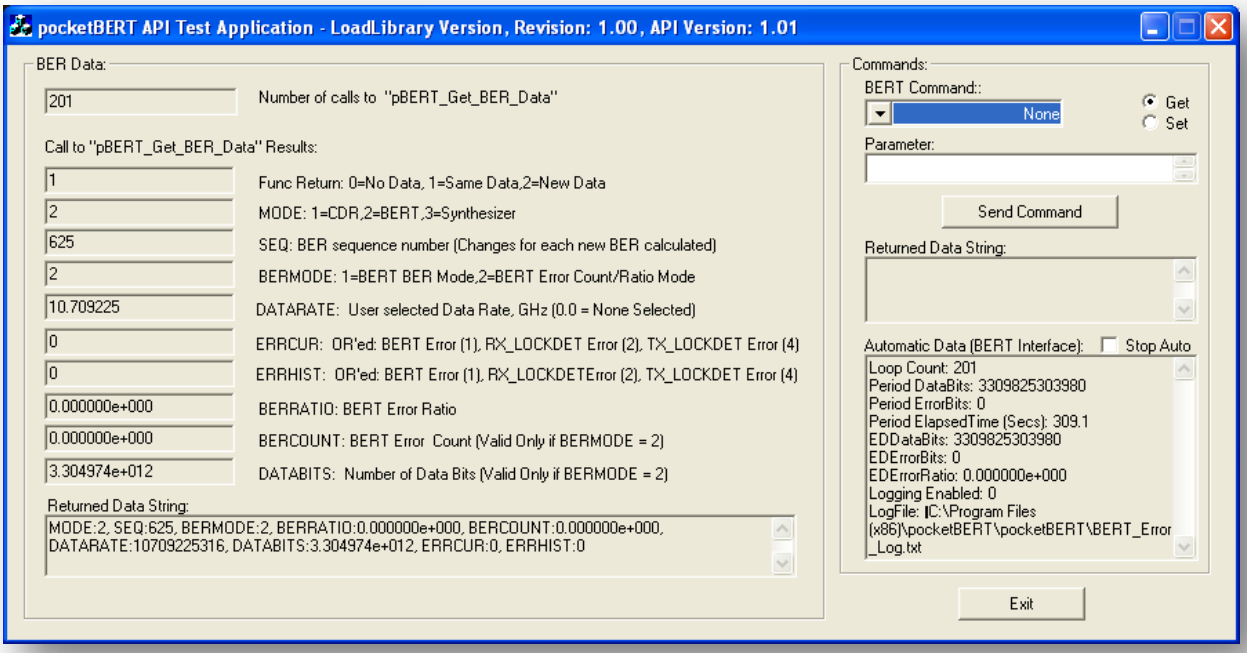

Figure 1: API test application screen

The left side of the screen, enclosed in the "BER Data" box, is the data returned from the "pBERTAPI_Get_BER_Data" function. This function is being executed automatically by the test application, so the data in this box is updated every time the command is executed.

The right side of the screen, enclosed in the "Command" box, contains the data for rest of the functions. The bottom list, labeled "Automatic Data BERT Interface", contains the information that is returned from the many commands that are being executed automatically by the test program. The section above this box contains controls to allow the user to issue commands manually.

To manually send a command, first select the "BERT Command" that is to be sent. Then select either "Get" or "Put". "Get" will read the current setting for the value/function and no "Parameter" is required to be selected. "Put" will set the value and does require a "Parameter" to be entered to be used to set the value. Finally, click the "Send Command" button. The data returned from the function will be displayed in the "Returned Data String" box.