

ZGUS Hidprobe Manual

1. Document

Document information

Info	Content
Keywords	HID, COM, USB, Serial
Abstract	Hidprobe is a single file executable Windows PC program for COM port and HID communication facility listing and communication as text and hexadecimal. Hidprobe includes scripting and a 7 bit single channel graph trace.

Software and Document Revisions

Rev	Date	Description
0.7e	27 Jul 2009	Manual updated to include graphics. Controls resized. New video documented
0.7b	31 May 2009	Some Hidprobe Script Additions (RunWait, ScriptName, ScriptFullName) Addition of ZGUS secure firmware files for use with ZG1L bootloader through Hidprobe as well as ZG1L and ZGUS documentation included. PIC Kit Serial Analyzers support removed due to inexpensive alternatives and due to no response to let ZGUS know if continued support desired.
	2006-2009	Various additions and improvements
	February 2006	First version released

Document Locations

The current version of this document can be accessed from <http://tech.zgus.com/hidprobe>

This document is distributed as zgus_hidprobe_manual.pdf with hidprobe software

A version of this document with reduced formatting, with no hyperlinking and with graphics removed is embedded within the hidprobe binary file and can be viewed with the help button

Copyright © Auscyber Pty Ltd 2006-2009

This information applies to software under development. Characteristics and specifications are subject to change without notice.

2. Contents

1. DOCUMENT	1
2. CONTENTS	2
3. HIDPROBE	5
3.1 Hidprobe Image and video	5
3.1.1 Hidprobe Video	6
3.1.2 Hidprobe Screen-shot Image	6
3.2 PC Side Devices	6
3.3 About Hidprobe	6
3.4 About HID and USB	7
3.5 About ZGUS Firmware Files	8
3.6 About COM	8
3.7 Contact Email	8
4. HIDPROBE USER INTERFACE	9
4.1 Hidprobe Basic Controls and Views Fig. 1	11
4.2 Tips checkbox	12
4.3 Script buttons and signal edit box	12
4.4 Baud Hint, Com Opt and Apply baud hint buttons	12
4.5 About, Help and Exit/Store Buttons	12
4.6 Find and Open Combos and Buttons Group	13
4.7 SetReport Recallable Group	15
4.8 SetReport By Keypress Only Group	16
4.9 Log Group	16
4.10 GetReport Group	17
4.11 Keys Sent Record	19
5. 'BAUD HINT' AND 'COM OPT' AND 'APPLY HINT' BUTTONS	20
6. CONNECTION/MULTIPLE DEVICES/MULTIPLE INSTANCES	22
7. HIDPROBE SCRIPT	22
7.1 Hidprobe Script User Interface	24
7.2 '? Script File' button	24
7.3 'Load/Run' or 'Stop Script' button	24
7.4 Signal Text Box	25
7.5 'Signal Script' button	25
7.5.1 Using Signal button to set DTR/RTS	25

7.6 Writing a Bootloader	25
7.7 Included Hidprobe Script Files	26
7.7.1 test.js	26
7.7.2 testfileregrow.js	26
7.8 Read/Write properties	27
7.8.1 BAUD	27
7.9 Read only properties:	27
7.9.1 Version	27
7.9.2 Connection	28
7.9.3 ScriptName	28
7.9.4 ScriptFullName	28
7.10 Write only properties:	28
7.10.1 DTR	28
7.10.2 RTS	28
7.10.3 XONXOFF	28
7.11 Functions:	29
7.11.1 alert	29
7.11.2 GetCountEvent	29
7.11.3 GetCountGetReport	30
7.11.4 GetNextEvent	30
7.11.5 GetReport	31
7.11.6 GetSignal	31
7.11.7 InjectGetReport	32
7.11.8 RunWait	32
7.11.9 SetReport	33
7.11.10 SetTimer	33
7.11.11 Sleep	33
7.11.12 UudecodeStrStr and UuencodeStrStr	33
7.11.12.1 UudecodeStrStr	33
7.11.12.2 UuencodeStrStr	34
8. HIDPROBE DEMONSTRATION FIRMWARE	34
9. TESTING	35
10. SYSTEM REQUIREMENTS	35
11. INSTALLATION	36
12. REGISTRY	37
13. ACKNOWLEDGMENTS	37

14. FAQ (FREQUENTLY ASKED QUESTIONS)	37
14.1.1 Why does text on buttons changes?	37
14.1.2 Are there default settings for connections?	37
14.1.3 Why can't I see any input reports?	38
14.1.4 I am getting garbage from COM connections	38
14.1.5 Why is there a COM baud option as high as 921600?	38
14.1.6 The trace window says 'Triggered Tracing' when it isn't.	38
14.1.7 My triggered trace is clipped or no points are plotted	39
14.1.8 My trace ends early and is jittery.	39
14.1.9 Why does Hidprobe Script use JScript?	39
14.1.10 What happens to the JScript alert?	39
14.1.11 Why use events instead of polling?	39
14.1.12 What are the script event handler functions?	39
14.1.13 Why is the log box telling me there is nothing to send in a 'repeat send' when I can see text in the SetReport Recallable edit box?	39
14.1.14 When I make changes to numerical values there is no action.	40
14.1.15 Why do I keep getting a disconnect when I send an OUT report?	40
14.1.16 Why does my device keep resetting when I do a find?	40
14.1.17 I can only get one USB VCOM device to work at a time.	40
14.1.18 Why the unusual user interface without a menu, toolbars or splitter windows?	40
14.1.19 Why is their no confirmed record of 'SetReport By Keypress Only' characters sent and the number of characters sent?	40
14.1.20 Can I report a suspected bug?	40
14.1.21 Should I be afraid of malicious code or spyware?	40
15. LICENSING AND DISCLAIMERS	41
16. CONTACT INFORMATION	42

3. Hidprobe

Hidprobe is a single file executable Windows PC program for COM port and HID communication facility listing and for communication through these facilities as text and hexadecimal. Hidprobe includes scripting and a 7 bit single channel graph trace. The Hidprobe distribution also includes files relevant to upgrading firmware of ZGUS products.

3.1 Hidprobe Image and video

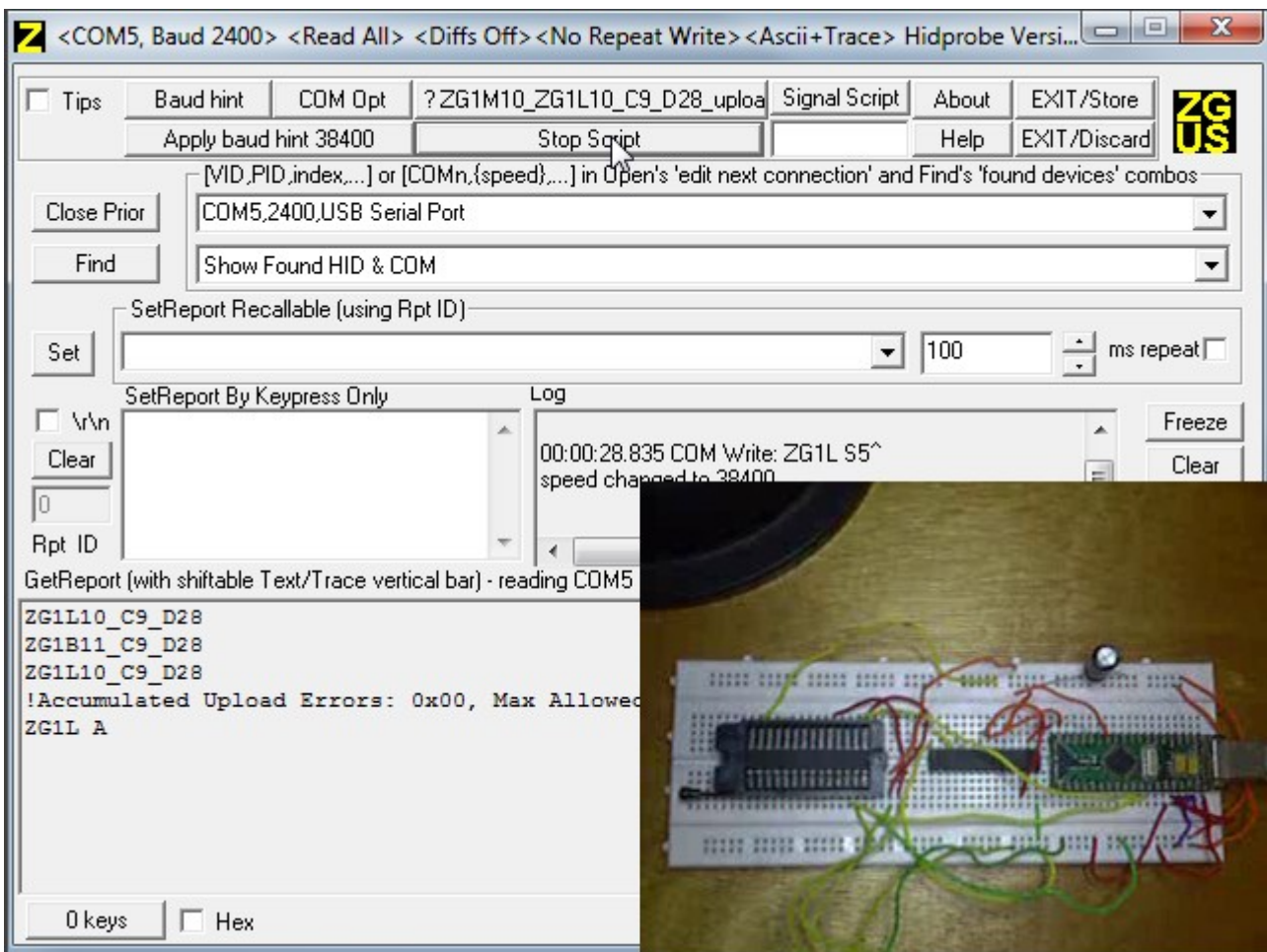


Image of Hidprobe at work using a script that has sent a command (ZG1L S5) at 2400 baud to change baud speed of an attached device to 38400 baud, has changed its own speed to 38400 baud and is waiting for an acknowledgment at this new speed. The window on the lower left is the response and acknowledgment from a previous command (ZG1L A).

3.1.1 Hidprobe Video

The image is taken from the video
http://s.zgus.com/videos/hidprobe_bootload.mp4

This video can currently be seen streamed from
<http://tech.zgus.com/techzgus/videos/item/4>

3.1.2 Hidprobe Screen-shot Image

The image is also used as the screen-shot image at
http://s.zgus.com/hid/hidprobe_screenshot.png

3.2 PC Side Devices

For use with contemporary PCs and laptops running MS Windows, we recommend products that use FTDI USB-RS232 chips as suitable for use with Hidprobe. Drivers are automatically installed

There are very inexpensive USB convertor cables available, such as the TTL-232R-5V for US\$20, among others, indicated at
<http://www.mouser.com/Search/Refine.aspx?Keyword=TTL-232R>

Please note for use of Hidprobe with ZG1L bootloaders, five volt interfacing is required for stage 3 bootloading.

A more expensive and flexible device is the DLP-2232M-G available for US\$35 from
<http://www.mouser.com/Search/Refine.aspx?Keyword=DLP-2232M-G>
For example this device will allow the i/o voltage to be set and can be used for other purposes such as I2C, SPI and JTAG communication with many embedded devices.

3.3 About Hidprobe

Starting with version 0.7a, the C runtime library is no longer statically linked into hidprobe executables. It is unlikely but possible this will require additional downloads.

While Hidprobe has limited terminal functionality (both Hex and ASCII), Hidprobe is not a strict terminal program as Hidprobe will not send or interpret code sequences of terminal protocols such as ANSI terminal. By default keyboard input is interpreted as ASCII instead of Hex. One of the two keyboard input mechanisms allows an escape sequence to interpret keys as Hex instead of ASCII.

Hidprobe is designed to allow easy recalling and editing of previously sent information as well as saving small amounts of recently sent information and the connection history.

HID has one very big advantage over COM. Multiple HID connections can be made to the same device at a user level without installation of special drivers. Hence Hidprobe can passively listen and filter out data sent to a PC from a HID device independent of the main PC application and can independently send data to the HID device independent of the main PC application

Despite the software name, the software performs no real probing of HID CONFIGURATION data.

With regard to HID connections Hidprobe waits passively for IN reports as passively requested, EITHER FROM HIDPROBE OR EITHER FROM ANOTHER PC APPLICATION (this can incorrectly lead to impression data is being passed back non passively). Request of IN reports non-passively has not been included. Such a non passive request feature requires the HID device supports this feature and requires Windows XP or later.

Hidprobe or the main PC application can issue OUT reports to the same device

Additional HID features may be added in a future version according as requirements vary for what the software is required to do.

Hidprobe cannot communicate with keyboard HID devices or mouse HID devices.

Hidprobe is a software only tool. It is currently beta software and currently runs on Windows PCs. HID/COM firmware has been written to demonstrate the use of Hidprobe and is described below.

Hidprobe is entirely self-contained in a single executable well under 1MB, including the help information. Hidprobe currently does not create files (but Hidprobe script files can) and the executable file can be moved without affecting its ability to run. Information can be saved to a single path location in the registry. Hidprobe can fit four separate instances without overlap in a screen with resolution 1280x1024.

Further information available from <http://tech.zgus.com/hidprobe>.

Hidprobe will not currently run 'as is' on Linux PCs as WINE does not include the HID API. WINE may also not include support for applications hosting script functionality. Hidprobe will provide a list of every Human Interface Device (HID) it can find along with some basic information such as Vendor ID (VID), Product ID (PID), input and output report lengths (including byte for report ID) and product string. Hidprobe will also list all COM ports it can find along with their description.

3.4 About HID and USB

HID protocol use is not limited to interfacing to normal 'Human Interface Device' computer products. HID is a very popular way for generic interfacing with USB

without installing customised USB drivers on Windows PCs. However HID does not enable the most efficient use of available USB resources.

Hidprobe is intended to be most useful when the HID protocol is being used as a carrier (rather than an end) to develop and test USB connected devices that will run without a customized Windows USB driver.

If a HID device can be opened for Read and Write (such as non keyboard and non mouse HIDs) Hidprobe will allow you to read from and write to the device. The first byte of a read is the report ID and it will be indicated. For simple HID devices this will probably be the report ID for writing and should be set in the Report ID box. The input and output report lengths include the report ID.

HID does not have to use USB. In Windows a HID USB class minidriver sits between the HID API and USB.

Windows Vista supports a more complete use of USB resources without the need for providing custom drivers.

3.5 About ZGUS Firmware Files

The Hidprobe distribution also includes files relevant to upgrading firmware of ZGUS products in the zgbootload directory. This directory also includes relevant documentation.

3.6 About COM

Here COM refers to the part of the name Windows uses to describe a port using the RS232 protocol. A common alternative name is 'serial port'. A Virtual COM port or VCOM is still named as a COM port. Common VCOM devices use USB in such a way that applications that open COM ports do not know they are using USB instead of real COM UARTS. Hidprobe has been tested with FTDI chip VCOM devices and works well with them.

The context is clear enough not to confuse COM with the complex Microsoft ActiveX 'Component Object Model' interface and tools (also known as COM).

3.7 Contact Email

The contact email address for Hidprobe is hidprobebeta@zgus.com. Bugs can be reported to this email address.

Please see below for more complete contact information.

4. Hidprobe User Interface

Here COM refers to the device name used by PCs for RS232 devices and not to 'Component Object Model' of ActiveX used to facilitate language independent communication between complex controls.

By default terminology is HID terminology rather than COM (or RS232 port) terminology. The reason for this is that HID terminology is less ambiguous with regard to the direction of transfer and corresponds more closely to USB concepts, which is replacing COM.

IN, Get, InReport or GetReport corresponds to Read or Input (to a Window from attached device)

OUT, Set, OutReport or SetReport corresponds to Write or Output (from keyboard or resend to an attached device)

With COM terminology a 'read' or 'write' must always be in relation to a context since a read to one host or device means a write from another host or device.

With HID/USB terminology

- IN always means 'read from device/write to host' and
- OUT always means 'write to device/read from host'.

Some of the buttons function as toggle buttons. A current state can be determined by reading the top status line. Buttons that can signal more than one state will have the next state they can signal visible as text on the button.

Hidprobe has

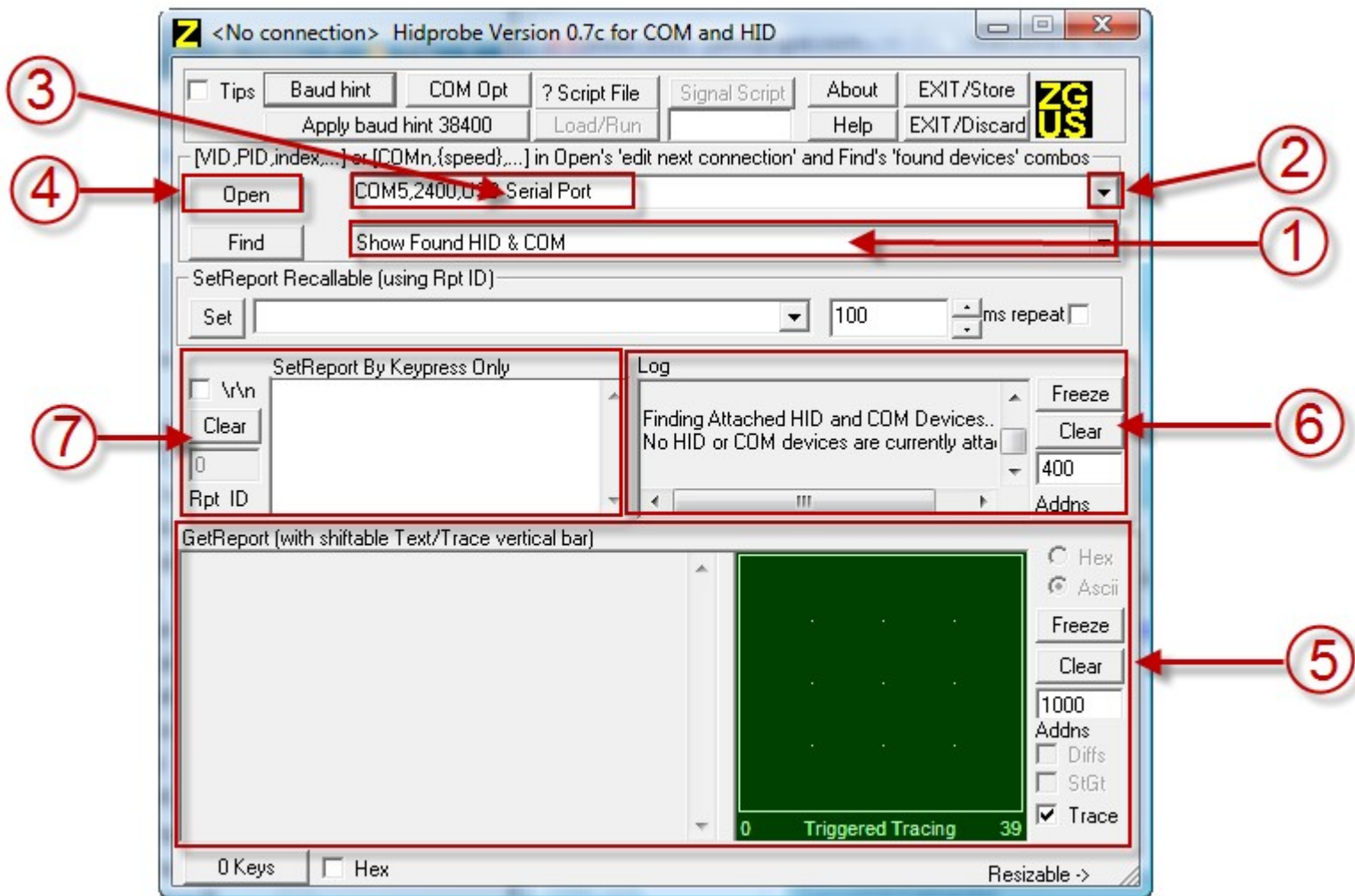
- A resizable dialog
- A connection status line at the top
- Tips check box. COM options. Script buttons and signal edit box. Help, About and Exit/Save or Exit/Discard buttons
- A find and open (HID/COM) section
- A SetReport Recallable section with controls with option for automated repeat. ASCII input with escape codes to enter hexadecimal can be used.
- A SetReport By Keypress only section with controls
- A Log Window with controls to keep you informed
- A GetReport Window with controls, including data trace graph window triggable with incoming data control codes.
- An ASCII/Hex inter-convertible and clearable record of recent 'SetReport By Keypress' keys sent and their number.

There are numerous large and small edit boxes. Read only edit boxes are grey. Most edit boxes and combo boxes have a right mouse button context menu that can be used for the following operations:

- Undo
- Cut, Copy, Paste and Delete
- Select All

File logging can be set up through Hidprobe script. Alternatively to obtain a lengthy file from the GetReport window set the max lines to a large number, freeze the window, select all in the window and copy (for example using right mouse button context menu in the window).

4.1 Hidprobe Basic Controls and Views Fig. 1



Hidprobe Basic Controls And Views

The image cannot be viewed in the version of the manual embedded in Hidprobe.

Region	Function
1	Click to show attached HID and COM devices
2	Click to show reusable record of previous connections
3	Automatic and manual editable box for next connection open
4	Close opened connection or open closed connection with region 3 information
5	GetReport region to show and control received information
6	Log region to show and control log information, including from scripts
7	SetReport by key-press region where a single key-press is sent immediately

4.2 Tips checkbox

Setting this checkbox on will provide balloon type tips when the mouse is left to hover over parts of Hidprobe. Tips is off by default. Currently only a selection of tips provided.

Buttons that are inactive will not show a tip.

Text boxes that are part of combo boxes need to have the mouse cursor hover over the down arrow on the right.

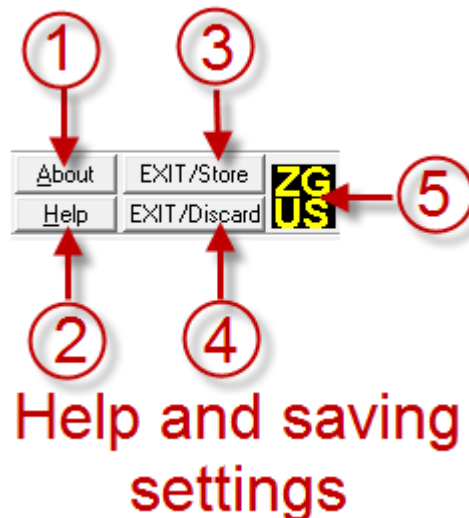
4.3 Script buttons and signal edit box

Please see separate section Hidprobe Script

4.4 Baud Hint, Com Opt and Apply baud hint buttons

Please see separate section below entitled: COM baud, 'Baud Hint' and 'COM Opt'

4.5 About, Help and Exit/Store Buttons



The About button (1) displays a dialog with information about Hidprobe and links.

The Help button (2) displays an embedded version of the Hidprobe manual without graphics,

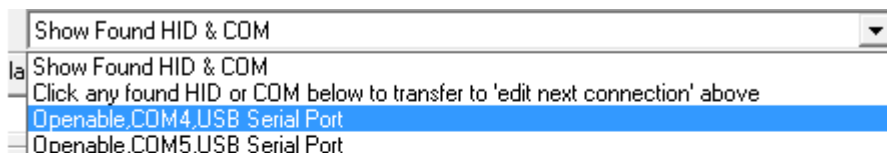
The EXIT/Store button (3) will quit and store the current 'Addns', 'SetReport Recalable' history and Connection history (up to last ten lines with most recent at the top) in the registry.

The EXIT/Discard button (4) closes the application without storing any changes in the registry.

The ZGUS image (5) is also a link to <http://tech.zgus.com/hidprobe>

4.6 Find and Open Combos and Buttons Group

The central concept of the combos beside the 'Open' and 'Find' buttons is as follows. The 'Show Found HID & COM' combo enables currently known devices to be listed and for a found device to have information transferred to the 'edit next connection' combo above for editing or immediate use SIMPLY BY CLICKING ON A LIST LINE. The 'edit next connection' combo above has a ten-line history and there is also a facility to save to the registry the connection history (by clicking on 'Exit/Save'). Editable comments will be stored but ignored when connecting.



**Click in region 2 of Fig. 1 to
choose editable port of region 3**

If the program is notified a device has been added or removed then the program will automatically start another enumeration, so it should not normally be necessary to click on the 'Find' button, unless you want to also list the devices in the Log window.



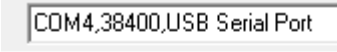
**If a port is open region 3 of Fig. 1 needs
to be cliked twice to effect change**

When a connection has been made the button with 'Open' has its text changed to 'Close Prior' on the basis you may have a new connection in preparation or ready to be made in 'edit next connection' box. The 'prior' connection to close is the connection listed in the top status line.

The 'Show Found HID & COM' combo box lists six pieces of information on a single line for HID

- 1) VID
- 2) PID
- 3) Index
- 4) InBytes length or GetReport length, including reportID byte
- 5) OutBytes length or SetReport length, including reportID byte
- 6) The devices Product String, if any

When transferred to the 'edit next connection' combo (by clicking on the line) the information can be edited. Connecting to a HID device only uses the first three numerals; everything else is ignored, is editable and can be stored.



COM4,38400,USB Serial Port

Choosing and editing port region 3 of Fig 1 has no effect until port opened or reopened with region 4

The 'Show Found HID & COM' combo box lists three pieces of information on a single line for a COM device

- 1) A status of 'Openable', 'In Use' or 'Registry'.
- 2) COM port name
- 3) Port description

A status of 'Openable' means the COM port is currently closed and has been confirmed can be opened.

A status of 'In Use' confirms the COM port is currently open.

A status of 'Registry' means no attempt has been made to confirm if a port found in the registry can be opened or is in use. The port descriptions that will not be tested is currently hard coded in. Some drivers with a virtual COM port may require multiple open attempts and may be slow to respond to an open request, so they are not tried.

When transferred to the 'edit next connection' combo (by clicking on the line) the line becomes edited to

- 1) COM port name
- 2) A baud value taken from the baud hint value
- 3) Port description

The line can be further edited Connecting to a COM device only uses the first two pieces of information along with the current COM (non baud) options. Everything else is ignored, is editable and can be stored.

4.7 SetReport Recallable Group

An output from PC (SetReport Recallable) combo box (with ten line combo box recall history) allows comments to be added that will be ignored. Pressing the 'Set' button or Enter key within the combo box will send or buffer the contents (as described below). '\ ' is the escape character. The escape sequences '\r', '\n', '\t', '\\', '\x' and '\X' have the same meaning as in the C language. The escape sequence '\c' starts a comment that is not sent. Unlike 'SetReport By Keypress Only' pressing the enter key does not add any characters. If you need to send a new line character then add in '\n' or '\r\n' at the end or immediately before '\c'.

For COM connections the entire line up to but not including '\c' (if present) is sent when the enter key or 'Set' button is pressed.

When the enter key or 'Set' button is pressed for HID connections as many OUT buffers as required are sent. If the final OUT buffer is not full when the enter key or 'Set' button is pressed then the previous un overwritten contents of the OUT buffer are sent (which are zero bytes if a new connection).

When the combo box has the focus pressing the up arrow and down arrow keys provides quick access to the history without using the mouse.

The checkbox 'ms repeat' will repeat the last SetReport actually sent (following opening a connection you will need to press 'Set' or press the enter key once to confirm: the Log window will inform you of this). The interval is in milliseconds. Updates to 'ms repeat' edit box are immediate if the 'ms repeat' check box is checked. The lower limit is 10ms and the upper limit is 10,000ms. If you enter outside these limits the pressing the enter key or losing focus will show what value is being used.

For HID the report ID byte in the 'Rpt ID' box in the group below is always prepended to a report and counted in the length. If there are 5 OutBytes is each SetReport then you need to provide four bytes to provide a complete report.

In 'SetReport Recallable' '\x01\x02\x03\x04\c send four bytes with hex value 0x01 0x02 0x03 0x04'

will send four bytes as described in the comment at '\c'. Adding a space between '\x04' and '\c' will send the space.

In 'SetReport By Keypress Only' '1234' with enter will send five or six bytes depending on whether the '\r\n' check box is set or not. The bytes sent are either '0x31 0x32 0x32 0x33 0x0A' or '0x31 0x32 0x32 0x33 0x0D 0x0A'.

To send the same from 'SetReport Recallable' send '1234\n' or '1234\n\c any comment 'or '\x31\x32\x33\x33\n' or '\x31\x32\x33\x33\0A' etc for the first case or

'1234\r\n' or '\x31\x32\x33\x33\r\n' etc for the second case.

However '1234\n \c any comment ' will send an extra space character as there is a space character between '\n' and '\c'.

4.8 SetReport By Keypress Only Group

Only key presses are sent. Pasting into the box or navigating around the box has no effect on what is sent. The enter key is sent as '\n' if the '\r\n' checkbox is not set and is sent as '\r\n' if the '\r\n' checkbox is set. A record of keypresses and their number is kept at the bottom of Hidprobe (please see below). ASCII values between 0x01(1) and 0x1A(26) can be sent by pressing keys between 'A' and 'Z' while the Ctrl key is pressed, they can be viewed in the record if the Hex check box at the bottom is set.

The 'Clear' button empties the window without any effect on what is sent.

For COM connections each key press is sent immediately.

For HID connections a new OUT report is sent when a buffer fills OR when the enter key is pressed (which adds '\n' or '\r\n'). If an OUT buffer is not full when the Enter key is pressed then the previous un-overwritten contents of the OUT buffer are sent (which are zero bytes if a new connection).

For HID the report ID byte in the 'Rpt ID' is always prepended to a report and counted in the length. If there are 5 OutBytes in each SetReport then you need to provide four bytes to provide a complete report. The update is immediate so check 'ms repeat' is off first. Report ID values must be between 0 and 255 (0xFF). If over the limit 255 will be used and will be shown when enter is pressed or focus is lost.

4.9 Log Group

The Log window can be frozen and cleared.

An 'Addns' edit boxes for controls the number of characters allowed to appear in the GetReport window. The lower limit is 5 and the upper limit is 5000. 'Addns' updates are immediate unless the update is over or under the limit. In this case pressing enter or losing focus will show the value used. When the addition limit is reached the buffer clears.



Automatic Detection

The log reports information such as errors, automatic notifications of device changes (which repopulates 'Show Found Hid & COM') and alert messages sent by scripts.

4.10 GetReport Group

The GetReports window (input to PC from device) can have incoming data viewed as Hex or ASCII. The Hex view time stamps incoming data. The GetReports window is subdivided into two windows that are separated by a movable splitter. The left hand side is a text window and the right hand side is a triggerable trace data graph window with seven bit resolution.

Double clicking on the vertical splitter bar or on the trace window will evenly divide the two windows of the GetReport group evenly.

The window on the left will show all values in Hex mode and will show values below 0x80 in ASCII mode in some form. In ASCII mode 0x20 to 0x7E is shown as ASCII text. The range 0x09 to 0x0D {'\t' (0x09), '\n' (0x0A), '\v' (0x0B), '\f' (0x0C) and '\r' (0x0D)} are interpreted as normal (both '\r\n' and '\n' is interpreted as a single new line - '\n' without '\r' before it is transformed to '\r\n'). 0x01, 0x02 and 0x03 are interpreted as trace control codes. 0x00 is interpreted as 'do nothing' or NULL: it is ignored. Currently 0x04 to 0x08 inclusive is also currently ignored. 0x0x0E(14) to 0x19(31) and 0x7F(127) are passed on as events (with 0x0400 added) to a Hidprobe script that is running and is not in a sleep state (whether a COM or HID connection). If XON/XOFF flow control is in use then the normal XON(0x11 or 17) and XOFF(0x13 or 19) codes are within the range passed on as Hidscrip events.

The trace window only works when in ASCII mode and only plots incoming values between 0x80 and 0xFF with the last bit transformed to zero. Hence 128 different data values between 0x80 to 0xFF are transformed to 0x00 to 0x7F (0 to 127). The trace window is designed to emulate a primitive single channel digital storage oscilloscope under control of firmware, not a pen strip device.

A minimum of two values are required to show a plot. Only the lines between each point are plotted and the displayed width between each point is always four pixels, while the displayed height between each point is determined by the height of the data graph window and by the value of the point. The smallest resizable size has a minimum of 127 pixels height between 0 and 127 but may be more. The number of points that can be plotted is shown in the lower right window. There is no consideration given to the time gap between when each value is received: it is up to you to ensure the trace has a meaningful time interpretation. To show more plotted points widen the data graph window by resizing Hidprobe horizontally or by shifting the splitter bar. If the splitter bar is moved such that the trace window can only display less than 10 points then points will not be plotted. To show a bigger vertical gap between points, resize Hidprobe vertically. Any resize of Hidprobe or move of the splitter bar loses the current plot. If 'Triggered Tracing' is enabled and values above 0x80 are still being received then this will refill the trace, even though a new

trigger has not been received. This can be taken advantage of in early design phases.

There are three data values not in the range 0x80 to 0xFF that have special meaning to the trace window. They are the following trace control codes.

- 0x01 'Triggered Tracing': Enable triggered tracing and trigger now (clear current trace, start a new trace and keep plotting until all points plotted or a new trigger is received)
- 0x02 'Enabled': Enable tracing (when all points have been plotted the trace is cleared and a new one is started)
- 0x03 'Disabled': Disable tracing

If Tracing with triggering is enabled and more values have been plotted without retriggering than fit the current width then there will be no further plots (however a resize or move of the splitter bar will plot anew until full again or until a new trigger code is received).

There is a 'Trace' check box which is really a manual pause button. If the check box is set (in ASCII view) tracing is allowed as above and if it is not set then no further data will be plotted. However incoming trace control codes will still be read and updated internally in ASCII view, ready for when the pause is released while data would have been plotted is not plotted, hence there may be a gap in the trace data. During a 'trace pause' a resize or move of the splitter bar will lose the current plot. The trace window reflects the last code received if tracing has not been paused. A trace pause is automatic in Hex view and on disconnect. Hence on disconnect the last displayed plot will not be cleared. The trace plot will be cleared with any new connection.

The default value on connection is 'Enable tracing with triggering and trigger now'. In Hex mode the current Trace pause state is overridden so no tracing will occur.

The trace window will X axis comments reflects the last control code update set while not paused. Hex view sets pause, ASCII view pause is optional. A disconnection sets a pause, Hence trace window state should be considered stale before a new connection. A new connection always sets the trace window state to 'Enabled,Triggerable' but this will only take effect if in an ASCII view with the optional pause is removed (that is if 'Trace' check box is set). The first ASCII view after a new connection always has trace pause removed.

To plot any eight bit value with seven bit resolution, say ch of type unsigned char, send as: (ch>>1)|0x80.

To plot any 16 bit value with seven bit resolution, say x of type unsigned short, send as (x>>9)|0x80, assuming only last eight bits are sent.

To plot any 32 bit value with seven bit resolution, say x of type unsigned long, send as (x>>25)|0x80, assuming only last eight bits are sent.

To plot a 10 bit analogue to digital result with seven bit resolution, stored left justified in the last 16 bits of AD0DR0, send as (AD0DR0>>9)|0x80, assuming only last eight bits are sent.

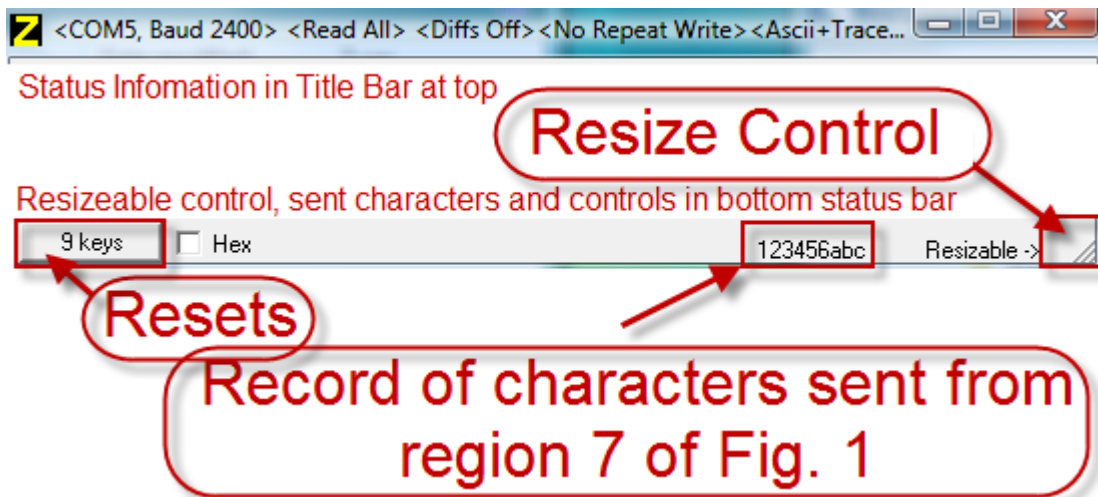
To plot a 10 bit analogue to digital result with seven bit resolution, stored right justified in the last 16 bits of AD0DR0, send as (AD0DR0>>3)|0x80, assuming only last eight bits are sent.

The GetReport left window can be frozen and cleared (use 'Trace' pause for the right hand window)

An 'Addns' edit boxes that controls the number of characters allowed in the GetReport window. The lower limit is 5 and the upper limit is 5000. Buffers can be viewed from the Hex view. Each new timestamp correspond to a buffer. 'Addns' updates are immediate unless the update is over or under the limit. In this case pressing enter or losing focus will show the value used. When the addition limit is reached, the buffer clears.

'StGt' check box which reads only the first IN report received after an OUT report is sent (useful for HID). StGt stands for Set then Get. When checked the status line includes the '<Read After Write>', otherwise '<Read All>'.

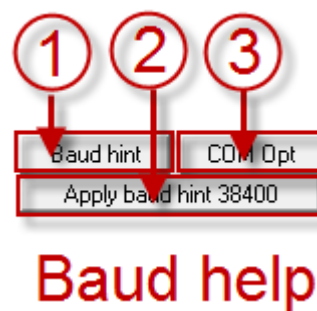
'Diff' check box which only shows new buffers received which are different to the prior buffer received (useful for HID). Currently for COM this is unlikely to work the way you might expect (the actual buffers received can be examined after each timestamp in the hex view). When checked the status line includes '<Diffs On>', otherwise '<Diffs Off>'.



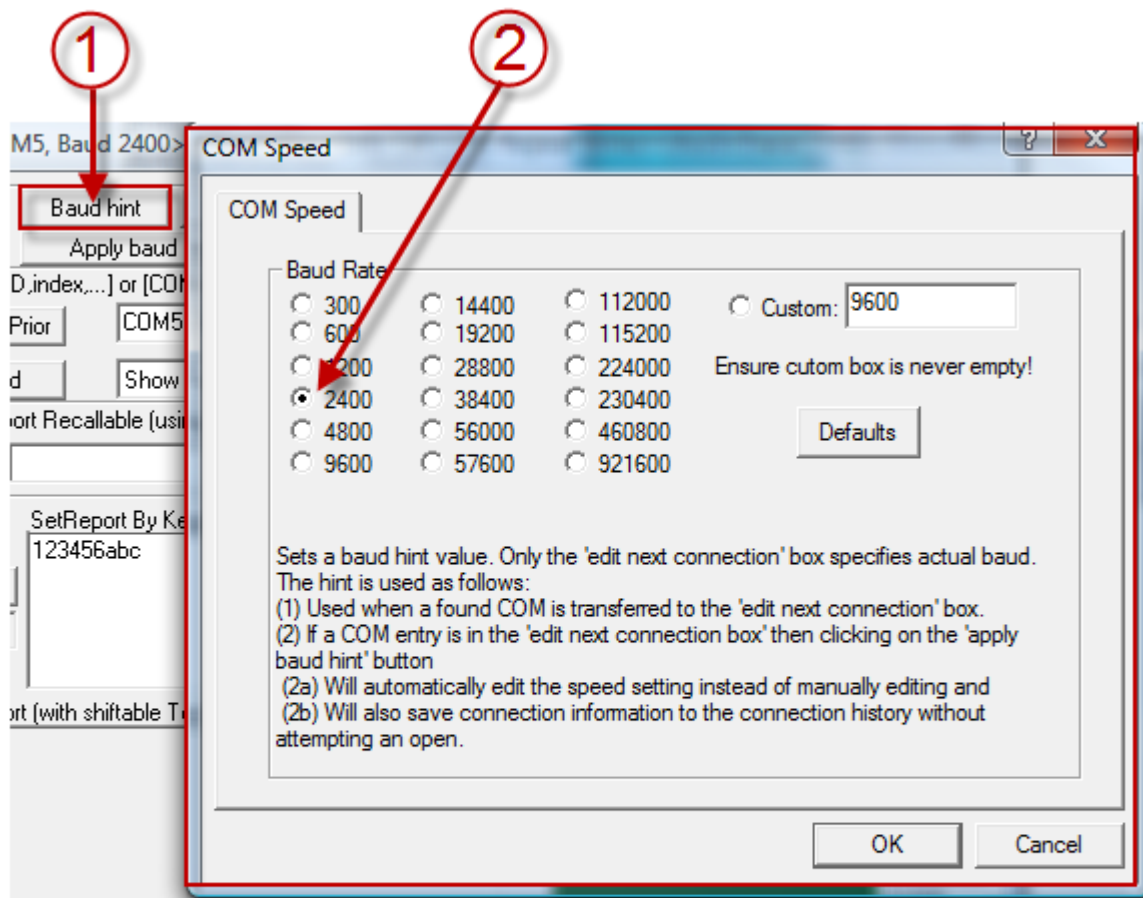
4.11 Keys Sent Record

The button at the bottom left records the number of keypresses in the 'SetReport By Keypress' window. The actual keys sent are recorded 'right justified' in space to the right. The Hex checkbox will interconvert the displayed record between Hex and ASCII. Clicking on the button will zero the record. Also using 'SetReport Recallable' will zero the record as will 'ms repeat'. Sending from script will not zero the record. ASCII values between 0x01(1) and 0x1A(26) can be sent by pressing keys between 'A' and 'Z' while the Ctrl key is pressed (provided a connection is active and the 'SetReport By Keypress' window has the focus).

5. 'Baud Hint' and 'COM Opt' and 'Apply Hint' buttons



The 'Baud hint' button (1) is intended to provide flexibility to the text box editable system of stored settings and histories. Using this button changes the speed baud hint speed listed in the 'Apply baud hint ...' button (2)



How to set the 'Baud hint' The 'Baud hint' edits region 3 and chooses Baud for region 2

Within the 'Baud hint' dialog clicking on a radio button (2) will choose a preset baud for the baud hint.

For a COM device the COM speed used to open a connection is written in the combo text beside the 'Open/Close Prior' button. The speed listed can be manually edited or can be automatically edited using the 'Apply baud hint ...' button above. The Other COM options can be viewed and changed by clicking on the 'COM Opt' button (3).

If a COM connection is currently open it can be changed while open by signaling a running script. Otherwise the connection will need to be closed first before the new applied speed value takes effect

The following help text is included when the 'baud hint' button is pressed:

Sets a baud hint value. Only the 'edit next connection' box specifies actual baud. The hint is used as follows:

- (1) Used when a found COM is transferred to the 'edit next connection' box.
- (2) If a COM entry is in the 'edit next connection box' then clicking on the 'apply baud hint' button
 - (2a) Will automatically edit the speed setting instead of manually editing and
 - (2b) Will also save connection information to the connection history without attempting an open.

6. Connection/Multiple Devices/Multiple Instances

Hidprobe will successfully connect to the same HID device from multiple instances of Hidprobe at the same time but can only connect to one COM device at a time.

Hidprobe uses a very simple system to identify HID devices for subsequent connection. Whenever a Vendor ID (VID) and Product ID (PID) is encountered during enumeration that is the same as before, an associated index number for the new HID device is incremented. Hence three numbers identify each HID device: its VID, PID and index. If there is only one HID device with a particular VID/PID then the index number is 0.

HID devices with the same VID and PID may be on the same physical device or different physical devices. If they are on the same physical device they may be as USB compound devices or as HID multiple top-level collections or both.

When connecting the same enumeration process is undergone until the required number of VID/PID encountered, matches the index number. A fresh enumeration may indicate an index number for a device that is different to what was previously enumerated in the same Hidprobe instance or in another current instance.

7. Hidprobe Script

Following are some issues to be addressed in the future. Currently the script SetReport will only send a maximum of 100 bytes from a string (a workaround exists using substr). Some examples will be provided which show the power and simplicity of hidprobe script with automated dialogue (such as updating firmware on a microcontroller or 'bootloading').

With Hidprobe script all data is string encoded and all Script external file access is through text based files. Hidprobe script provides access to powerful JScript engine built in string manipulation functions. This does not mean the strings are physically sent and received as strings. Hidprobe script uses '\x' to escape string encoded eight bit data for sending and will encode incoming data directly into strings (there are two string receive modes: ASCII and Hex). Included are two additional Hidprobe

provided functions to encode and decode string encoded eight bit data into string encoded six bit data (using UU-encode format algorithm) that retains capacity to remain in ASCII receive mode. The advantage of this is that plot data that bypasses processing by Hidprobe script and retain custom or XON/XOFF software flow control is retained.

If you want to manipulate data in real time then you can do this from Hidprobe script. An example is provided of interacting with the file system through the Scripting.FileSystemObject created with the intrinsic JScript ActiveXObject. You could set up another program to monitor a file written to through a pipe or redirection. Unlike the root Windows Script Host object, Hidprobe does not include a 'CreateObject' method to sink events from created ActiveX objects.

It should not be difficult to web enable your device with Hidprobe, just using the FileSystemObject with some CGI scripts and Regular Expression objects constructed from the global RegExp inbuilt object!

No function parameters are accepted by reference internally by Hidprobe. While this reduces efficiency it ensures there is less chance script programs will cause crashes.

If you just want to do a straight read that waits until information is available then a wrapper function can be written to be called from the main eventLoop (please see examples in test.js).

When GetReports return a zero length string the queue is empty. The maximum number of GetReports queued is 1000. The maximum number of events queried is 2000. There is only one script signal and it is not cleared by reading it. The script signal may be an empty string (it is by default). Only the following sets 'regular' events: Get Report, Timer, and the 'Script signal' button. A timeout event occurs when one of the previous events did not occur within the timeout time. A sleep does not allow any of the events to occur, however the script signal string is still set and the timer will still operate. GetReports are not added to the queue during a Sleep.

The signal is a string. When Signal Script is clicked the contents of the edit box is copied. It is up to the script to call GetSignal as soon as the event occurs to find what this string is. Unlike GetReport reports are not queued. However signal events are queued. The max queue size of GetReports is 1000. If the script delays and a second signal event is sent then the signal string from the script signal text box will be overwritten by the time the first signal event is actioned. If the script is asleep then although the signal string will be set the signal event will not be set. Hence after a sleep consider calling GetSignal to examine the current signal state. The signal string is also filled with the contents of the signal string text box when the script starts to run so on start up the contents of the edit box can be used to pass arguments. Simply call GetSignal() even though an event signal has not been sent. The maximum size of the edit box is 256. The 'Script Signal' does not need to be pressed to set up the initial signal. The initial signal is set up from whatever the contents of the edit box is on click of 'Load/Run'.

Responses to signal sent from the signal box can be written to Hidprobe Log window by Hidprobe Script with alert().

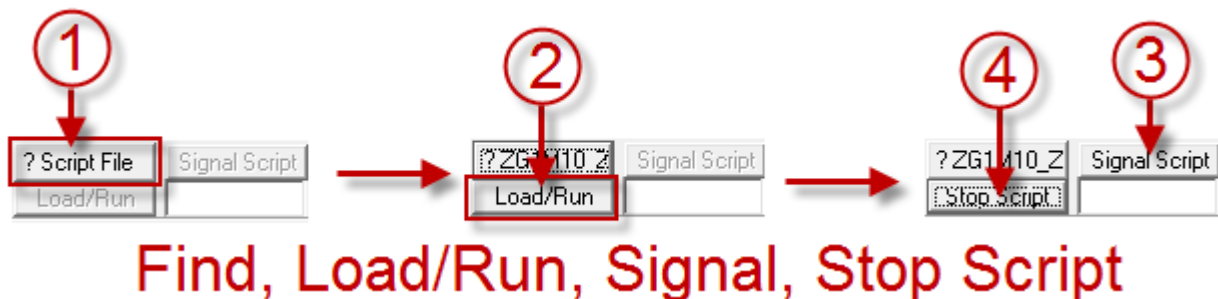
If script signaling with a line history is required then it can be quickly implemented from a console with an external script. Hidprobe script needs to read a file. Two Example Hidprobe Script files are included 'test.js' and 'testfilegrow.js'

'As is' many of the properties have no effect on the internal state of Hidprobe. Setting properties may silently fail without notification. The Hidprobe script API (or Application Program Interface) is immature, grossly incomplete, in a state of flux and is currently been written 'on the fly' to deal with day to day 'in house' requirements as they arise. The API is likely to be seriously overhauled if it appears to start becoming unmanageable. Hidprobe is, after all, still in early beta. Naturally it will be in our own interests to maintain backward compatibility with API changes and additions to stop what will be our own 'beta legacy' scripts from breaking. For example it is likely there will be function equivalents of properties prefixed by Get and/or Set (the interface does not allow the same level of function overloading as understood in C++) and/or a GetLastError function or property.

'short' means an unsigned two byte value and 'long' means an unsigned four byte value.

7.1 Hidprobe Script User Interface

The Hidprobe Script user interface has four controls. These are three buttons and one edit box.



7.2 '? Script File' button

The '? Script File' button (1) only serves to choose a file for the 'Load/Run' button (2) below. The file can be chosen at any time, even if is not complete and can be edited at any time even when a script is running. When a file has been chosen then the button text will change to reflect the file name.

7.3 'Load/Run' or 'Stop Script' button

The 'Load/Run' button immediately reads/rereads and loads/reloads the contents of the chosen file, as indicated in Script File button above.

When running the script program can terminate itself or be terminated by clicking on the 'Load/Run' button (which will now say 'Stop Script' (3) if running).

Running again will only use the current saved edited version at time that the 'Load/Run' button is pressed. This makes debugging very easy. The file chosen with '? Script File' need not even be finished and can still be edited.

To start running a newly edited version of a script that is already running, click twice on button that currently reads 'Stop Script'.

7.4 Signal Text Box

The Hidprobe Script signal text box is below the 'Signal Script' button (4). The contents of the edit box are used to set a signal string on script run startup that can be read at start up by GetSignal(). The contents of the box are sent automatically as a signal when a script is started with 'Load/Run'

7.5 'Signal Script' button

After setting a script to run, clicking on the 'Signal Script' button will read the contents of the signal text box and will also send a Signal Event to the script if the script is not sleeping at the time.

7.5.1 Using Signal button to set DTR/RTS

While the Hidprobe user interface has no controls to set DTR and RTS, it is possible to write a script that can be signaled to set DTR and RTS from the user interface.

7.6 Writing a Bootloader

Using Hidprobe Script as it now, with a connection set up by a user, there is no reason why a bootloader could not be developed and tested.

The script can set DTR and RTS (typical for bootloaders) either by itself or by signaling the script to do this from the user interface. All data can be sent encoded within \x escape sequences. A few short functions can convert data formats.

String functions UudecodeStrStr and UuencodeStrStr are designed to help transfer binary data efficiently using an Ascii setting. Using the Hex setting the script can read all responses.

In fact Hidprobe could enable a HID based USB bootloader to be written (with a custom boot loader on the firmware).

7.7 Included Hidprobe Script Files

7.7.1 test.js

A log file named testlog.txt maybe gnerated.

Just comment in one of the following lines to perform basic tests

```
eventLoop();
//baudLoop();
//sleepLoop();
//test_DTR_RTS_Loop();
//testInjectLoop();
//testUU();
//testDCEvents();
//testBAUD();
```

7.7.2 testfileregrow.js

This script is NOT a Hidprobe script. Testfileregrow.js illustrates advanced but useful scripting concepts.

Testfilefregow.js can be adapted to interact with Hidprobe when Hidprobe is running another script.

The truth is testfileregrow.js had a useful but forgotten purpose with Hidprobe scripting back in the mists of time (long before Hidprobe with scripting was made publicly available and long before Hidprobe scripting documentation was prepared). Hence testfileregrow.js needs to be updated and illustrated for a current practical purpose. From examination of testfileregrow.js it appears to cater for two main scenarios involving processing information that can arrive at an arbitrary time.

One scenario is where the arbitrary information can be

- from standard input in a console session
- from an external file fed or piped in as standard input (such as `cat file.txt | cscript testfileregrow.js`)
- from an external file that can be added to but not deleted from

For this scenario `echo(fnpointer)` can be used where `fnpointer` is a pointer to a function (or callback) that processes the new information. However it should be noted piping in an external file as standard input that it is intended can be added to is not practical. Instead the file should be processed as an external file instead of piped in as standard input.

The other scenario is where arbitrary information from an external process (such as Hidprobe running a separate script) can be

- added to an external file
- deleted from the same external file
- added again to the same external file (regrown)

For this scenario echoRegorowableFile(fnpointer) can be used where fnpointer is a pointer to a function (or callback) that processes the new information. The external file is not piped in as if standard input.

In summary. If an external file will not be deleted from or if an interactive console session is desired then echo(fnpointer) can be used . If the external file can be delted from echoRegorowableFile(fnpointer) should be used

7.8 Read/Write properties

The Read/Write properties are:
BAUD

7.8.1 BAUD

long=BAUD or BAUD=long

long=BAUD reads BAUD rate. Returns -2 if there is no connection open. Returns -1 if a HID connection is open. Returns 0 if a COM connection should be open but was unable to read the baud rate using an operating system call. Returns a positive value as the baud rate if was able to read the baud rate using an operating system call.

BAUD=long(four bytes) attempts to set BAUD rate only if a COM connection is open. Fails silently if a COM connection is not open. Can use BAUD as a read propery to determine result. For example

```
BAUD=38400; myBaud=BAUD; if(myBaud!=38400) errorBaud(myBaud,38400);
```

Important: If a change of BUAD is successful there will be no update of Connection status information (the speed at opening will be shown and returned by the Connection property).

Currently the write only BAUD property is the only way to change BAUD rate with breaking a current COM connection (you can signal script to do this).

7.9 Read only properties:

The read only properties are:
Version
Connection
ScriptName
ScriptFullName

7.9.1 Version

string=Version

Returns Hidprobe Name and version in a string

7.9.2 Connection

string=Connection

Returns the current status information in Hidprobe title bar as a string. To read a numeric value of connection information please see the BAUD property (-2 if no connection is open, -1 if a HID connection is open, 0 or positive if a COM connection is open).

7.9.3 ScriptName

string=ScriptName

Returns the file name of the current loaded script.

Useful to analyse a name to determine arguments for a script instead of passing argument to a script.

7.9.4 ScriptFullName

string=ScripFulltName

Returns the full path of the current loaded script.

Useful to determine the current path of a script.

7.10 Write only properties:

The write only properties are:

DTR

RTS

XONXOFF

7.10.1 DTR

DTR=bool(true or false)

Please see below

7.10.2 RTS

RTS=bool(true or false)

Both DTR and RTS write only properties have no effect on internal state and records of Hidprobe and only takes affect if a COM connection is currently open. With a new COM connection the real DTR and RTS is automatically false no matter what the prior script settings of DTR and RTS were.

On an Embedded Artists board DTR=true will pull down the reset pin and RTS=true will pull down P0.14 (the boot loader pin)

7.10.3 XONXOFF

XONXOFF=bool(true or false)

This turns on or off XON/XOFF flow control at the OS level if a COM connection is active. There appears to a manual and automatic aspect to this setting from the perspective of the internal workings of Hidprobe. The manual part is that if Hidprobe receives 0x13 as sent by the attached device then Hidprobe should manually stop

sending to the device until 0x11 is received as sent by the device. The automatic part is that 0x13 is automatically sent by the PC OS (not by Hidprobe) when the receive OS buffer is approaching full to ask the device to stop sending. 0x11 is automatically sent by the PC OS (not by Hidprobe) when the OS receive buffer approaches empty to request more. Presumably the automatic part takes effect by writing requests by Hidprobe failing to complete immediately. While this will not stop Hidprobe the practical effect of this is that packets queued internally by Hidprobe for transmission may start to get silently dropped by Hidprobe.

Hidprobe has its own internal buffering system and Hidprobe will pass on the normal XON and XOFF control characters as script events if Hidprobe is in ASCII mode whether or not XONXOFF has been set on and whether or not a COM or HID connection is active. XON and XOFF can also be generated independently of the automatic OS response. For example from the keyboard 'Ctrl S' generates XOFF or 0x13 and 'Ctrl Q' generates XON or 0x11 (provided the 'SetReport By Keypress Only' edit box has the focus).

7.11 Functions:

The functions, in addition to JScript engine built in functions are:

```
void alert(string)
short GetCountEvent()
short GetCountGetReport()
short GetNextEvent(long)
string GetReport()
string GetSignal()
void InjectGetReport(string)
bool RunWait(string)
bool SetReport(string)
bool SetTimer(long)
short Sleep(long)
string UudecodeStrStr(string)
string UuencodeStrStr(string)
```

7.11.1 alert

void alert(string)

Implemented as an internal Hidprobe function and named alert to retain consistency with traditional expectations. The string is sent to the Log window. Unicode characters will not be stripped (unlike with SetReport and InjectGetReport). 'alert' is not part of JScript. Instead Windows Script Host adds a host WScript.Echo function that generates a message box when script is run with wscript.exe and writes to standard output when script is run with cscript.exe.

7.11.2 GetCountEvent

short GetCountEvent()

Returns the number of queued events waiting to be processed by a running script event loop.

7.11.3 GetCountGetReport

short GetCountGetReport()

Returns the number of queued reports waiting to be processed by a running script event loop. There will be one or more characters per report up to a maximum buffer size per report.

7.11.4 GetNextEvent

short GetNextEvent(long)

The main events are script shutdown, event wait timeout, event timer, getreport and signal.

In addition in Ascii mode a certain range of ASCII codes are passed through as events instead of added to reports.

It is important to keep in mind that GetNextEvent returns a short instead of a string. This means integers can be used directly in case tests for switch statements. Please see the caution in GetSignal section.

GetNextEvent requires a parameter is passed specifying a timeout in milliseconds to return if no event is received within this time. A timeout of 0 specifies return immediately. A timeout of -1 specifies don't return until there is an event to report. If a timeout occurs then the event signal is HIDSCRIPT_EVENT_TIMEOUT. If the timeout is 0 and there is no event to report then HIDSCRIPT_EVENT_TIMEOUT will be reported, even though there was no real timeout.

If an event number of 0 (HIDSCRIPT_EVENT_SHUTDOWN) is returned then 100ms is available to cleanly exit the script before the script is forcibly terminates, if a script is running.

Please see included Hidprobe script files for sample use

The Hidprobe event signals are

```
var HIDSCRIPT_EVENT_SHUTDOWN      =0x00;
var HIDSCRIPT_EVENT_TIMEOUT       =0x01;
var HIDSCRIPT_EVENT_TIMER         =0x02;
var HIDSCRIPT_EVENT_GETREPORT     =0x03;
var HIDSCRIPT_EVENT_SIGNAL        =0x04;

var HIDSCRIPT_EVENT_TIMEOUT_OUTOFRANGE =0x0800;
var HIDSCRIPT_EVENT_WAITING       =0x0801;
//above only applicable if a script makes a multi threaded call
var HIDSCRIPT_EVENT_ERROR         =0x0802;
```

If Ascii mode is active then the following ASCII codes are passed through as events instead of added to reports: 14 to 31 inclusive and 127. The event codes are the Ascii codes with 0x0400 added to them

An example is provided in test.js that uses InjectGetReport to insert string encoded data into the incoming data processing stream without a connection. Comment in testDCEvents() to demonstrate.

In test.js the following apply to the Ascii mode generated events

```
var HIDSCRIPT_EVENT_DC1      =0x0411;
var HIDSCRIPT_EVENT_DC1_STR  ="\x11";
var HIDSCRIPT_EVENT_DC2      =0x0412;
var HIDSCRIPT_EVENT_DC2_STR  ="\x12";
var HIDSCRIPT_EVENT_DC3      =0x0413;
var HIDSCRIPT_EVENT_DC3_STR  ="\x13";
var HIDSCRIPT_EVENT_DC4      =0x0414;
var HIDSCRIPT_EVENT_DC4_STR  ="\x14";
```

HIDSCRIPT_EVENT_DC1_STR is string encoded to help inject data in. Since GetNextEvent returns a short HIDSCRIPT_EVENT_DC1 does not need to use a string in a switch case test.

7.11.5 GetReport

string GetReport()

Returns the next waiting report **as a string**. Returns an empty string if there are no reports.

For a non empty report there will be one or more characters per report up to a maximum buffer size per report.

In Ascii mode some characters are passed through as events instead of added to reports.

7.11.6 GetSignal

string GetSignal()

This retrieves the current content of the Signal text box **as a string**. This function should be called if the HIDSCRIPT_EVENT_SIGNAL is received. This event signal is generated when the 'Signal Script' button is pressed.

Since JScript does not require type declarations, if the signal is to be tested as a numerical value it can be tested directly against a numerical value. For example if -2 is entered in the signal edit box then var signal=GetSignal() can be tested with a numerical value:
if (signal==-2) then ...

The reason this is successful is JScript attempts to convert both objects to the same type for the purpose of the comparison

Caution: case tests in a JScript switch statement are against the unconverted underlying object without converting both to the same type. Hence test a signal passed as 2 as case "2": or for a single character (since 'x' defines a string x, not character x) as case '2': but not as case 2:

7.11.7 InjectGetReport

void InjectGetReport(string)

Inserts a pseudo response from a device. It can be used when a device is closed and can be used to help test your scripts. An injected GetReport will also generate a GetReport event so you can both display and read back your response. Also data can be plotted in ASCII mode. This could be useful if 'playing back' massaged data during a disconnection. With InjectGetReport there is no automatic injection of a 0x01 TriggeredTrace control code that would simulate setting the initial state with a real connection (if you need the state to be set then inject 0x01). For InjectGetReport you can use \x escape coding. If you send characters that are not part of regular ASCII then the most significant byte of the multibyte Unicode character will be ignored.

7.11.8 RunWait

bool RunWait(string CmdLine)

Hidprobe as the script host attempts to execute the following Windows function: `CreateProcess(NULL, CmdLine, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi);`

If `CreateProcess` returns `FALSE` then `FALSE` is returned. If `CreateProcess` returns `TRUE` then Hidprobe script host running in its own separate thread waits for the created process to terminate and then returns `TRUE`. This does not mean the created process successfully completed its task, only that another process was successfully created and that Hidprobe script waited for the new process to terminate.

Currently Hidprobe Script does not provide a property to return the value of the Windows function `GetLastError()`, which is useful if the process failed to start. The source of such errors is better determined outside Hidprobe script.

Following is an example of use

```
CmdLine="C:/Python26/python.exe test\\test.py"
if(RunWait(CmdLine))
    log("executed " + CmdLine)
else
    log("execution failed of " + CmdLine)
```

Note use of '\\\ ' instead of '/' for the directory separator for the argument to python.

7.11.9 SetReport

bool SetReport(string)

SetReport strips the most significant byte out of any double byte Unicode characters sent through JScript, thus forcing use of the extended ASCII subset. ASCII hexadecimal can be sent as '\x' (lower case x only).

7.11.10 SetTimer

bool SetTimer(long)

A time in milliseconds is passed that determines how often a timer event is generated. If a time of less than 10 is passed then the timer is turned off and false is returned.

7.11.11 Sleep

short Sleep(long)

Sleep will prevent events being recorded internally by Hidprobe. Sleep returns 1 if sleep was successful, 0 if script is being shut down and 0x0F00 if a sleep could not be set. Other 0x0Fnn values may be returned. No events will accumulate during a sleep, such as TimerFire events and GetReports events. When a sleep is over events will start to accumulate again. The TimerFire event is not reset and the next timer event will occur at the same time as if there was no sleep.

7.11.12 UudecodeStrStr and UuencodeStrStr

The following two functions convert between string encoded arbitrary eight bit data and string encoded arbitrary six bit data using the UU-encode algorithm for binary data. Essentially the advantage of using UU-encode format is that arbitrary data can be sent and received that will always only take on one of 64 values between 0x21 and 0x60 (33 and 96) inclusive with only a 33% loss of bandwidth by encoding eight bit data as six bit data. All the control codes are unused, the space character is unused, the lower case letters are unused and nothing in the range 0x80 to 0xFF is used. With Hidprobe scripting with a connection receiving in ASCII mode: data (above 0x80) can still be graphed directly without being processed by the script. Also XON/XOFF or custom software flow control can be retained (currently not fully taken advantage of). A description of UU-encode is available at <http://www.wotsit.org>.

7.11.12.1 UudecodeStrStr

string UudecodeStrStr(string)

Series of the eight characters (with space characters pre stripped) are decoded and returned as three space separated hex digits. If there are up to seven characters remaining they are not included in the decoding. Use Ascii mode to ensure the script can interpret incoming ISP data with minimal filtering. For example string "25605e48" decodes as string "140fa8 "

7.11.12.2 UuencodeStrStr

string UuencodeStrStr(string)

Space striped hex digit character pairs are encoded and returned as a \x encoded string. There must be six characters to start an encoding and further multiples of six to continue to the next encodings. Left over characters are not included. For example string "140fa8" encodes as string "\x25\x60\x5e\x48". Hence three bytes that might be stored in text format as six bytes actually gets sent as ascii printable four bytes and not as three bytes.

- So far no accumulating CheckSum functions have been included.

- So far no functions to retrieve a left portion of an encoded or decoded string to string conversion have been included.

8. Hidprobe Demonstration Firmware

The firmware uses a USB VID of 0x1781 and PID of 0x0930 as assigned to Auscyber Pty Ltd through another party. The firmware can be downloaded from http://tech.zgus.com/hidprobe/hidprobe_demo.hex

This firmware runs on Philips LPC2148 ARM7TDMI-S microcontroller. This firmware is suitable for use on an Embedded Artists LPC2148 USB Quickstart Board plugged into an Embedded Artists Prototype Board. The COM baud rate is 38400.

The firmware side (not the PC side) sends IN reports (five bytes long) through HID and COM and receives OUT reports (four bytes long) through HID and COM (leaving aside the additional HID report ID byte which is not included when interrupt endpoints are used). The first four bytes of the IN reports and the first four bytes of the OUT reports represent the following information that control the alternating state of eight LEDs (on pins 0.15 to 0.22 inclusive): byte 1:led state1, byte 2:led state2, bytes 3 and 4: low and high bytes of time in milliseconds to alternate between the two states. An OUT report will be echoed by an IN report. The first two bytes will not be changed but the third and fourth bytes will be changed to limit the millisecond repeat time to between 10 and 10,000 milliseconds if necessary. The fifth byte of an IN report represents the state of input pin 0.14: 0x00 if low, 0x01 if high. The reason for choosing pin 0.14 is indicated below.

As well as OUT reports echoing IN reports, OUT reports are also generated once a second and whenever the state of input pin 0.14 changes. Also at the same time the once per second OUT report is sent the internal COM receive buffer is cleared so OUT reports sent through COM need to be started and completed between each one second scheduled IN report. An IN report will also be sent through HID (and not COM) if an IN report is requested through the USB control endpoint with a HID Get InReport Request (however currently Hidprobe only waits passively for IN reports and does not include a way to actively request an IN report). The firmware uses idle mode power control.

Following are some specific comments about setting up the two boards Embedded Artists boards.

The Embedded Artists LPC2148 USB Quickstart Board has three jumpers. They are the reset and boot loader jumpers (connected to DTR and RTS COM signals) and the power from USB jumper. After uploading firmware remove the reset and boot jumpers. The USB link led is not connected as an UP_LED (as indicated in a schematic in a Philips datasheet). The led in fact indicates that a CONNECT signal to pull up USB D+ through a 1.5K resistor is being asserted. If VBUS detection is enabled then the signal will disappear giving the impression the led is an UP_LED. This is important for the following reason. To make USB work from your own code you will need to code PINSEL1 as `PINSEL1 |= 0x0x80004000; //CONNECT(out) and VBUS(in)` and not what you might expect as `PINSEL1 |= 0x0x40004000; //UP_LED(out) and VBUS(in)` (assuming relevant bits are already 0 such as after a reset).

The Embedded Artists Prototype board has numerous jumpers. To view LEDs, jumpers for LEDs for pins 0.15 to 0.22 inclusive should be connected. The jumper for led for pin 0.23 should preferably be left disconnected and the led digit SPI interface pin 0.23 jumper should preferably also be left disconnected. Pin 0.14 is not an ideal choice for an input button for the simple reason that if pin 0.14 is pulled down when a power reset occurs the LPC2148 will enter ISP boot loader mode. However it is pulled up on the LPC2148 USB Quickstart Board (so avoiding a floating input) and is also an External Interrupt pin (EINT1 allowing both level and edge sensitive polarised interrupt including in power down mode).

9. Testing

The software has been tested with Multiple Hid and serial ports devices attached to the PC at the same time through the same USB hub (including Virtual COM devices that are really COM to USB converters).

With three instances of Hidprobe open three connections to the same LPC2148 Quickstart Board were made: one by COM through a Virtual COM device and two through HID. Updates from one instance were read by the other instances.

10. System Requirements

Hidprobe is written in C++ with template libraries. The run time requirements for Hidprobe are modest for software that requires real time access and perform real time UI updates. The OS requirements are the C run time libraries, the Windows

API, Windows common controls, the Registry (if you choose 'EXIT/Store') and the RichEdit control (to display Help).

Various versions of Hidprobe were developed and tested on MS Windows 2000/XP/Vista PCs.

Hidprobe will not currently as is under Linux with WINE, as WINE does not include HID API compatibility. It is currently unknown if WINE includes support for applications that act as script hosts.

Starting with version 0.7a, the C runtime libraries are no longer statically linked in. It is unlikely but possible this will require additional downloads from <http://tech.zgus.com/hidprobe> or if preferred from Microsoft.

11. Installation

Hidprobe is distributed as a zip file (typically hidprobe.zip) with the following files and directories:

- zgus_hidprobe_manual.pdf,
- hidprobe.exe,
- hidprobe_demo.hex,
- test.js,
- testfileregrow.js.
- directory zgbootload

The directory zgbootload contains three other directories

- base
- lib
- log

- ZGUS.pdf ZGUS documentation
- ZG1L.pdf bootloader firmware documentation

and various secure bootload script files written to use through Hidprobe such as
ZG1B11_ZG1L10_C9_D28_upload_secure.js
ZG1M10_ZG1L10_C9_D28_upload_secure.js

There are also data only secure bootload files that can be used to upload secure firmware into the base directory without using Hidprobe

lib includes a required hidprobe script library

To install just unzip hidprobe.exe to a directory of your choice. To run hidprobe just double-click hidprobe.exe or make a shortcut to hidprobe.exe. To uninstall, just delete hidprobe.exe. To install hidprobe_demo.hex firmware, just use a normal flash ISP bootloader (such as supplied by Philips with Device LPC2148, Xtal Freq 12000, Baud Rate 38400) or JTAG debug device which can load and control a IAP flash download program placed in RAM.

To use Hidprobe to upgrade/downgrade/swap firmware please see the included ZG1L manual.

If hidprobe.exe does not run then C runtime redistributables can be obtained from <http://tech.zgus.com/hidprobe>. The three dll files can be installed in the same directory as Hidprobe or in any 'Path' directory such as C:\WINDOWS\system32

12. Registry

Hidprobe writes to and reads from a single directory path in the registry. Registry entries are only made if you click on 'EXIT/Store'. Cleaning the registry manually is not advised due to the consequence of errors. If you wish to clean the registry of hidprobe entries then remove the registry path HKEY_CURRENT_USER\Software\Zgus\Hidprobe. Removing registry entries will not affect the ability of hidprobe to run. Hidprobe.exe can be moved on the PC without affecting its ability to run or to use the registry.

13. Acknowledgments

John Heenan wrote code and documentation. Windows UI code was used and adapted by John Heenan from various open sources for which a licence is granted. The largest source of windows presentation code for use and adaptation was from the WTL (Nenad Stefanovic of Microsoft?) and from Bjarke Viksoe of www.viksoe.dk, The UI code feature of passing SetReport line into a history combo box with enter key was coded independently by John Heenan.

14. FAQ (Frequently Asked Questions)

14.1.1 Why does text on buttons changes?

These are toggle buttons. Look at the status line at the top. Status of is reported between angle brackets '<' and '>'. The text on toggle buttons indicates the next state that can be signalled.

14.1.2 Are there default settings for connections?

Yes. The default settings are 'ms repeat' off, Diffs off, StGt off. For COM ASCII is default. For HID Hex is default. For HEX trace cannot be enabled. For ASCII trace

pause is off and tracing is enabled with triggering as if firmware had sent trace control code 0x01.

14.1.3 Why can't I see any input reports?

Check you are connected with Diff off, StGt off and Hex view active. Check the GetReport and Log views are unfrozen.

14.1.4 I am getting garbage from COM connections

Check your COM options, including speed. Try the default COM options

14.1.5 Why is there a COM baud option as high as 921600?

Virtual COM though USB outstrips real COM and applications do not have to continuously pump data at their chosen and accepted rate. Some microcontroller UART peripherals and level converters can handle such speeds. USB can handle this speed. The Texas Instruments TUSB3410 RS232/IrDA Serial-to-USB Converter includes a software selectable baud rate up to this speed. The NXP/Philips ARM9 LPC3180 includes three high speed UARTs that can handle this speed and has USB 2.0 Full-speed interface along with DMA. FTDI Chip have USB converter chips with RS232 transfer rates up to 1M baud.

A single USB 2.0 Full-speed logical interrupt endpoint can reserve guaranteed bandwidth with data integrity up to 512000 bits per second in both directions at the same time and multiple logical interrupt endpoints can be used by the same device to reserve higher guaranteed bandwidth in both directions at the same time. Bulk endpoints provide data integrity with unguaranteed bandwidth and Isochronous endpoints provide very high levels of bandwidth without data integrity. FTDI chips can operate in Bulk or Isochronous modes.

14.1.6 The trace window says 'Triggered Tracing' when it isn't.

The trace feature is designed to be a firmware controlled feature, not a user interface controlled feature (unless commands are sent for interpretation to your firmware with SetReport). The 'Trace' check box is really a trace pause check box. The X axis comments always reflects the trace status before tracing is paused. A trace pause override occurs when switching to Hex view or when asked to pause in ASCII view or when disconnecting. An 'Enabled,Triggering' state always occurs with a new connection, regardless of what the firmware requests. This is to cater for interrupted data streams with an unknown prior request state. If a HID connection a pause is automatic because Hex is the default for HID. On changing to ASCII for the first time for HID the pause will be automatically removed. For a new COM connection tracing not be paused because ASCII is the default mode. Hence the only trace features not under firmware control when in ASCII view is 'Trace pause' and the first time ASCII is viewed with a new connection. Hence the firmware can be considered to be the controlling partner for tracing, not you, unless the firmware allows you to be the controlling partner.

14.1.7 My triggered trace is clipped or no points are plotted

Points are always four pixels width from each other, no matter what the width of the trace window is. The wider the trace window is the larger the number of points that can be accommodated for display. Adjust the trace window width with the splitter bar or with a resize of Hidprobe. If the trace window is made so narrow that less than 10 points can be fitted then lines will not be plotted. The number of points used to form lines is shown on the bottom right of the trace window. You can tell your firmware to adjust the number of points for plotting for each trace or you can adjust the trace width to accommodate the number of points for plotting in a trace, which your firmware can tell you through the ASCII view. Currently Hidprobe will not allow the firmware to automatically adjust the trace width and so the number of points for plotting.

14.1.8 My trace ends early and is jittery.

This happens at high speeds when there may be small delays in order to allow several points to be plotted at once. If there is a new trigger before all the points at the end are plotted then the end points don't get plotted.

14.1.9 Why does Hidprobe Script use JScript?

The Hidprobe audience is expected to be embedded developers. The syntax of JScript is closer to C. C is to firmware development what VB is to PC software development. Either or both or even other script engines such as Perl or Python can be included. At the moment it is more convenient to stick to a familiar syntax for examples. It should also be mentioned that has a scripting host Hidprobe provides its own set of support functions and that some functions that are a part of some scripting hosts and are not a part of the language will not be present.

14.1.10 What happens to the JScript alert?

Hidprobe implements the familiar Jscript host function 'alert' as a request to write to the log window instead of popping up a message box whose contents disappear when the box is dismissed.

14.1.11 Why use events instead of polling?

Programs that run on PCs (including Hidprobe script) should not 'spin' between testing for events. The event loop provides a more flexible and efficient system than polling after a sleep or a spin. A Hidprobe Sleep is really a way of turning off accumulation and reporting of events for a specified time.

14.1.12 What are the script event handler functions?

You create your own script event handler functions by hooking them up yourself from the event loop centered around the GetNextEvent function.

14.1.13 Why is the log box telling me there is nothing to send in a 'repeat send' when I can see text in the SetReport Recallable edit box?

After a connection has been established click on Set or press enter when focus is in the edit box to confirm you want to use the saved value.

14.1.14 When I make changes to numerical values there is no action.

If the current value is outside limits then it will be ignored. To see the current value used either press the enter key, lose focus from the editing box or look at the log.

14.1.15 Why do I keep getting a disconnect when I send an OUT report?

Check the first digit has the correct report number (frequently 0). Check report number from first digit of an input report from Hex view. The last input ID read is indicated in the text for GetReport group box.

14.1.16 Why does my device keep resetting when I do a find?

This may happen if you have a COM port that can also assert reset and boot loader pins through DTR and RTS signals and if the COM port is closed at the time of the find. During a find COM ports are opened to see if they are in use or not. The simple fact of opening can cause DTR/RTS assertions even if the port was previously closed with these options turned off. On an Embedded Artists quickstart board simply remove the reset and boot loader pin jumpers to avoid this happening.

14.1.17 I can only get one USB VCOM device to work at a time.

If you purchased devices at the same time from the same source the devices may have the same serial number encoded in the firmware. A utility should exist to check and change the serial number of your devices if necessary.

14.1.18 Why the unusual user interface without a menu, toolbars or splitter windows?

First the GetReport window contains two windows with a vertical splitter bar. Controls relevant to a particular view window are deliberately kept close to the view window. The easiest way to get started with this was to group controls and views into dialog based group boxes. At least the dialog user interface is resizable.

14.1.19 Why is there no confirmed record of 'SetReport By Keypress Only' characters sent and the number of characters sent?

There is! Please see the 'Keys Sent Record' in the Hidprobe User Interface section. The record is below the 'GetReport' group, which is where such a record is traditionally expected.

14.1.20 Can I report a suspected bug?

If you find a bug specific to existing stated functionality you are welcome to report it.

14.1.21 Should I be afraid of malicious code or spyware?

Hidprobe is not malicious code or spyware. You need to be aware that scripts provided to you that can be run within Hidprobe may be malicious.

15. Licensing and Disclaimers

Short Name: Hidprobe 0.7b

Full Name Hidprobe 0.7b for HID and COM

License: License to use as an application granted only on acceptance of the no liability clause below. No license granted to alter binary. No part of 'About' information may be removed, altered or added to.

Copyright (C) Copyright 2006-2009 Auscyber Pty Ltd

Distribution: compiled binary code may be freely distributed and used as-is as an application, without warranties of any kind.

No Warranty: No warranty of any kind, such as no warranty of fitness for any purpose, including no warranty of fitness for merchantability.

No Liability: Auscyber Pty Ltd shall not be liable for any damage or harm the software may cause by its use, whether from faults or not, - including direct, indirect, incidental, consequential, loss of business profits or special damages

Download <http://tech.zgus.com/hidprobe>

Documentation <http://tech.zgus.com/hidprobe>

Contact hidprobebeta@zgus.com

Online Information: <http://tech.zgus.com/hidprobe>

16. Contact Information

ZGUS is a trading name of Auscyber Pty Ltd (ABN 65 089 449 632)

ZGUS
22 Ragnar St
Edmonton QLD 4869
Australia

Telephone +61 7 4045 3118
Facsimile +61 7 4045 2408

Email support@zgus.com
Internet www.zgus.com

Postal Address:
ZGUS
PO Box 793
Edmonton QLD 4869
Australia

The information in this document is believed to be accurate in all respects at the time of publication but is subject to change without notice. Auscyber Pty Ltd assumes no responsibility for errors and omissions, and disclaims responsibility for any consequences resulting from the use of information included herein. Additionally, Auscyber Pty Ltd assumes no responsibility for the functioning of unsubscribed features or parameters. Auscyber Pty Ltd reserves the right to make changes without further notice. Auscyber Pty Ltd makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Auscyber Pty Ltd assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Auscyber Pty Ltd products are not designed, intended, or authorised for use in applications intended to support or sustain life, or for any other application in which the failure of the Auscyber Pty Ltd product could create a situation where personal injury or death may occur. Should Buyer purchase or use Auscyber Pty Ltd products for any such unintended or unauthorised application, Buyer shall indemnify and hold Auscyber Pty Ltd harmless against all claims and damages.

SOFTWARE PREVIEW