AN2014-1
Gryphon®2 Application Note

Part 2: Using and Programming Gryphon for Engineers and Programmers

Document Revision: 1.0

Covers additional Gryphon programming tools, including gioctl, gchanctl, gschdtx.

CONTENTS

1 INTRODUCTION

This Application Note is for those engineers who want to begin using and writing application programs for the Gryphon® family of network protocol adapters. This Application Note is a continuation of the Application Note 2013-1 that describes the basics of Gryphon command-line tools. You will become familiar with more of the command-line tools that come with the Gryphon product.

2 ASSUMPTIONS AND AUDIENCE

This Application Note assumes that you are familiar with Application Note 2013-1.

3 SCOPE OF THIS APPLICATION NOTE

The Gryphon products are feature-rich, fully functional multi-protocol adapters running Linux. As such, the Gryphon has many features and utilities that allow engineers to develop networked ECUs and diagnose protocol networks and ECUs on those networks. This Application Note covers a few more functions of using the command-line utilities that are essential for becoming experienced using the Gryphon.

4 PRODUCT BACKGROUND

The Gryphon family of products are a set of automotive and vehicle network protocol adapters. The Gryphon has protocol card slots that accept protocol adapter cards for various standard automotive and truck protocols. Gryphons are used primary in engineering development applications, and for manufacturing programing and testing of ECUs.

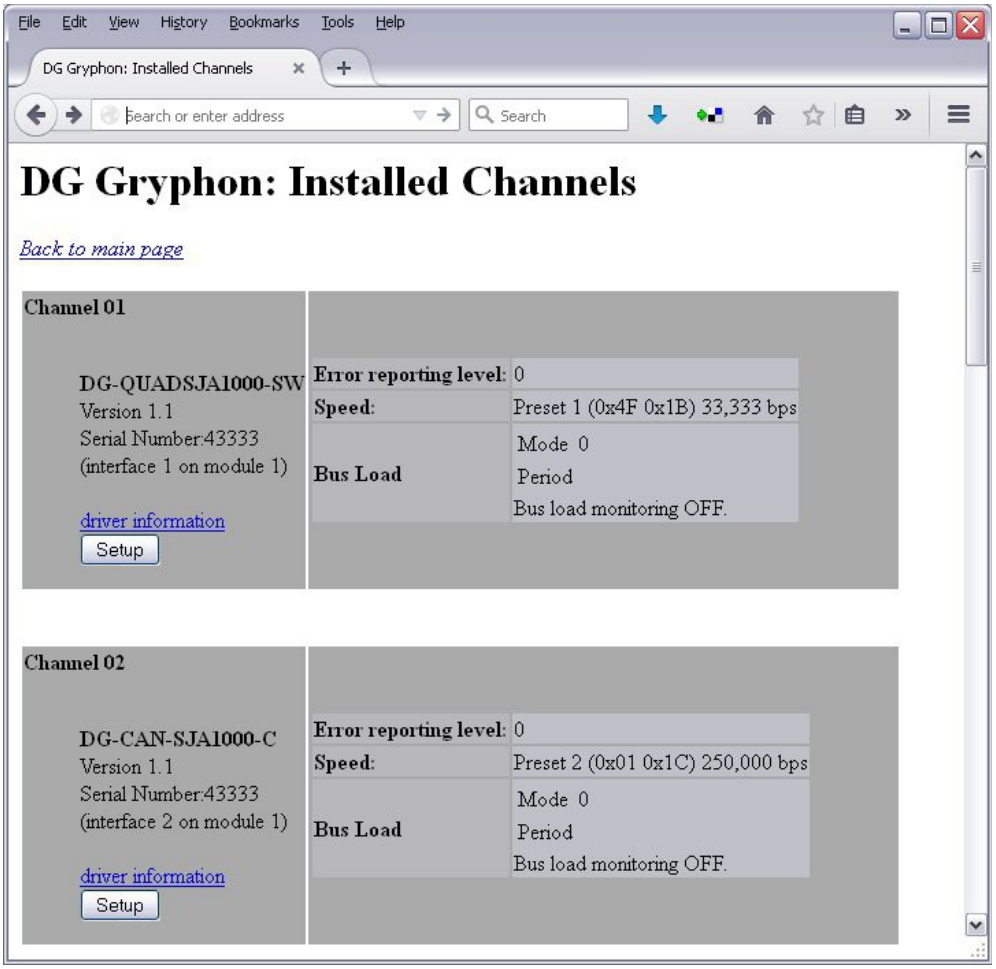
DG Technologies manufactures two versions of the Gryphon product, the Gryphon 2 and the S4. The Gryphon 2 has a faster processor, three (3) protocol card slots, three (3) USB host ports, wired Ethernet, and front panel display and control buttons. The Gryphon 2 can have up to twelve (12) channels of protocol communication by populating the slots with, for example, three quad CAN cards. The S4 has two (2) built-in channels of CAN protocol, and one (1) protocol card expansion slot. The products run the Linux operating system and communicate with a host computer through a standard Ethernet network. Both versions of Gryphon support optional wireless Ethernet by the use of USB Wi-Fi adapters.

The supported protocols include: CAN, GMLAN, GMUART, ISO9141-2, ISO11898 (CAN), ISO15765, J1850 GM (Class 2), J2284, J2411 (GM SWCAN), J2534, KWP2000 (ISO14230), and LIN.

5 CHANNEL CONTROL WITH GCHANCTL

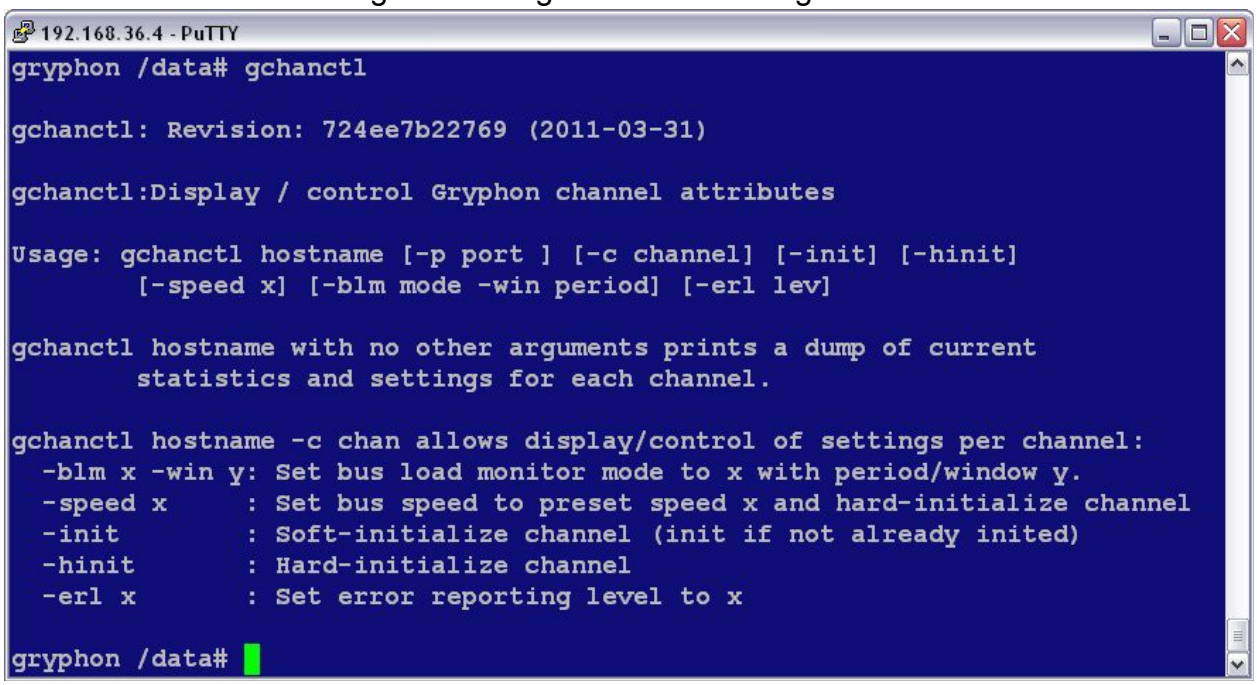
We often use the Gryphon web page interface to change the baud rate of the channels. The interface is shown in Figure 1.

Figure 1. Gryphon Channel Control Web Page



However, many times it is easier and faster to change the baud rate using the command-line command gchanctl (pronounced G-CHAN-CONTROL). You can use the gchanctl command to change preset baud rates, init the channel, set the error reporting level, and set bus-load monitoring right on the Gryphon command-line. When you run gchanctl on the command-line with no arguments, you get the following usage information shown in Figure 2.

Figure 2. Usage Information for gchanctl



For example, you can change the preset baud rate to Preset 2 with the following command:

```
gchanctl localhost -c 2 -speed 2
```

The above command produces the following output and reports that channel 2 speed was set and the channel was initialized:

```
Channel 2: Speed set.
Hard init.
```

You can also initialize the channel with either a soft-init using the -init option, or a hard-init using the -hinit option.

```
gchanctl localhost -c 2 -init
gchanctl localhost -c 2 -hinit
```

6 I/O CONTROL WITH GIOCTL

You can use the gioctl (pronounced G-I-O-CONTROL) command to send single I/O control commands to a Gryphon card. Each card has it's own list of I/O control functions that that specific card will recognize. Using I/O controls, you can get or set all of the card parameters and features.

For example, here are the first few I/O controls for the SJA1000 CAN card:

Table 1: A Sample of a Few I/O Controls for CAN

I/O Control Name	Value	Description
GCANGETBTRS	0x11200001	Get SJA1000 BTR values
GCANSETBTRS	0x11200002	Set SJA1000 BTR values
GCANGETBC	0x11200003	Get SJA1000 Output Control Register
GCANSETBC	0x11200004	Set SJA1000 Output Control Register
GGETERRLEV	0x11200005	Get driver error level reporting
GSETERRLEV	0x11200006	Set driver error level reporting

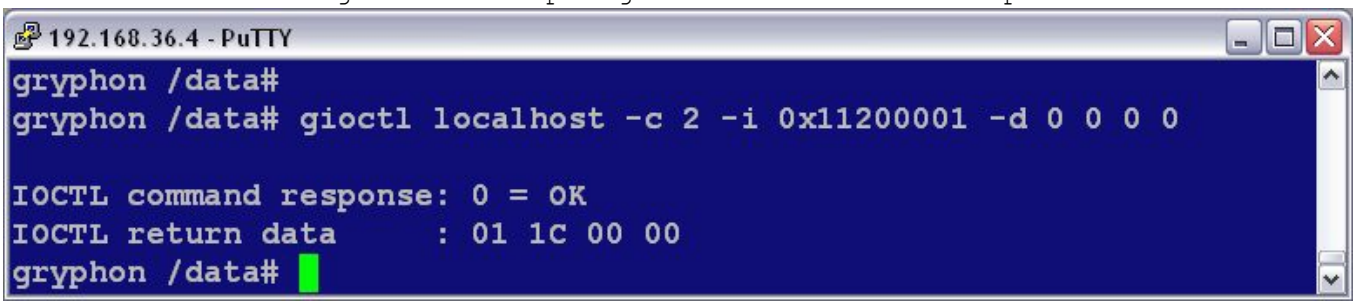
The I/O control names are defined in the source code header files, along with the values. You can find these SJA1000 CAN card I/O controls in the header file `dev_527.h`.

To run gioctl on the Gryphon command-line, you specify the channel, the I/O control value, and the data. For example, to get the CAN BTRs for channel 1, issue the following the command:

```
gioctl localhost -c 2 -i 0x11200001 -d 0 0 0 0
```

When you run the above command, it produces the output shown in Figure 3.

Figure 3. Example gioctl Command and Output



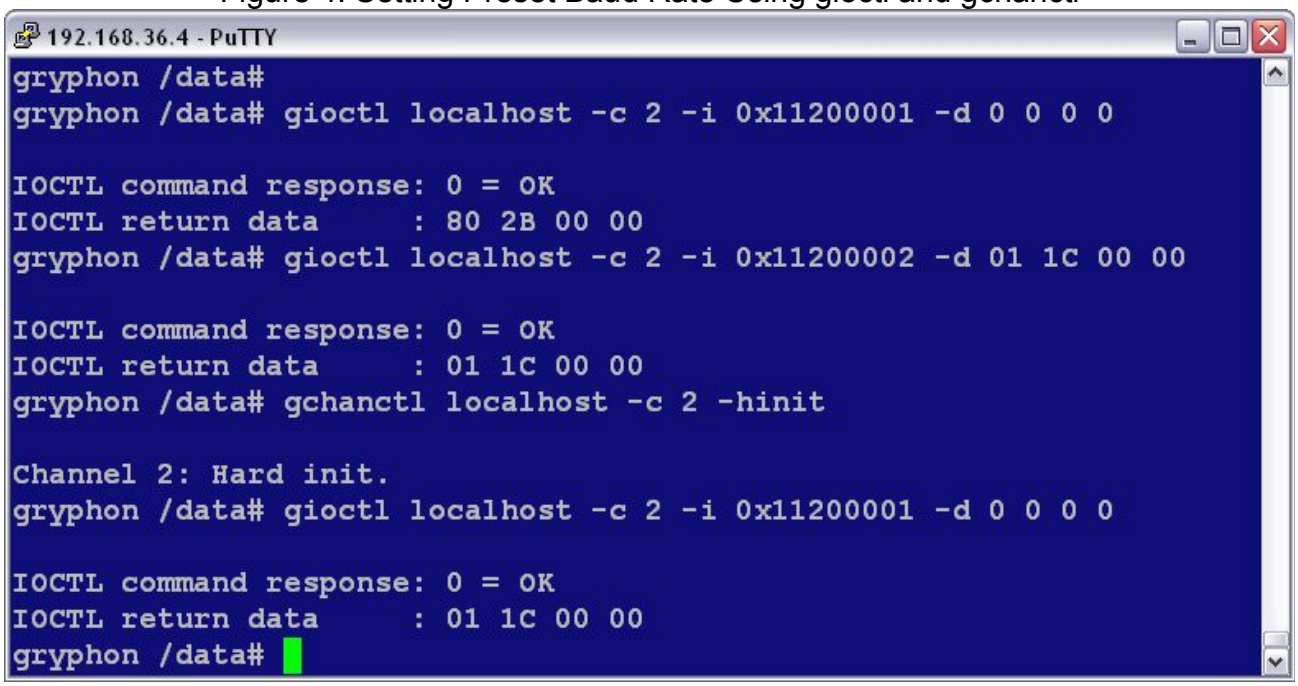
The return data are the BTR values, in this case 0x01 0x1C 0x00 0x00. This value corresponds to preset 2 baud rate on the SJA1000 card. Notice that we specified "`-d 0 0 0 0`" in the gioctol command. This is because, when we get I/O control values, we need to include a data value for each expected returned data value.

When setting I/O controls, some I/O controls take effect right away, and other only take effect after the next init. As an example to demonstrate this, let's set the BTRs to the preset value 2 values, and initialize the channel to have the value take effect. We can use the getBTR I/O control to monitor our progress. Here are the commands:

```
gioctl localhost -c 2 -i 0x11200001 -d 0 0 0 0
gioctl localhost -c 2 -i 0x11200002 -d 01 1C 0 0
gchanctl localhost -c 2 -hinit
gioctl localhost -c 2 -i 0x11200001 -d 0 0 0 0
```

The first command gets the BTRs, the next command sets the BTRs, the third command initializes the channel, and the fourth command gets the BTRs again. In Figure 4 you can see the result of running these commands.

Figure 4. Setting Preset Baud Rate Using gioctl and gchanctl



Set the BTRs is one of the I/O controls that does not take effect until after the next initialization.

All I/O controls are listed in the online Gryphon documentation. The documentation refers to the Gryphon card header files for you to find the specific I/O control values for you to use in your gioctl command. For your convenience, a summary of the header files for each card are listed below in Table 2.

Table 2: Gryphon Header Files Containing I/O Control Definitions

Gryphon Card	Header File Name(s)
Dual 82527 CAN card SJA1000 CAN card Dual SJA1000 Fault Tolerant CAN card Single-Wire CAN card Quad CAN card	`dev_527.h` `dev_sja.h`
Dual GM J1850 (DLC) card DCX card Dual J1708 card	`dev_dlc.h`
Ford J1850 (HBCC) card	`dev_scp.h`
KWP2000 / BDLC card	`dev_kwp.h`
LIN card Dual LIN card	`dev_lin.h` `dev_ubp.h`
Honda card	`dev_hon.h`
NBUS card	`dev_nbus.h`
PIC I/O card	`dev_iopic.h`
Power and I/O functions	`dev_pwr.h`
Generic IOCTL numbers	`gen_ioctls.h`
Delay Driver function	`dev_delay.h`
Ford UBP card	`dev_ubp.h`

7 LIN RESET USING GIOCTL

You can reset the LIN card using gioctl command GRESETHC08. The GRESETHC08 command has the I/O control value of 0x11800009.

```
gioctl localhost -c 1 -i 0x11800009
```

This gioctl command will reset the LIN card to it's power-on state.

8 SCHEDULING COMPLEX TRANSMISSION SEQUENCES WITH GSCHDTX

In the Application Note 2013-1, you learned how to transmit simple messages using the Gryphon command `gryphtx` (pronounced GRYPH-T-X). Here you will learn how to construct complex sequences of transmissions, and how you can run those sequences with the `gschdtx` (pronounced G-SCHED-T-X) tool. The gschdtx command can do complex transmissions that are not possible to accomplish using gryphtx.

The command gschdtx takes a user-defined input data file and starts a Gryphon schedule. First, let's look at the gschdtx command-line options, and then we will look describe the details of the input data file.

The command-line arguments for gschdtx are:

```
gschdtx hostname [[-c channel] [-i iterations] [-w] -f file] | [-k id]
```

Just like the other command-line commands, you specify an IP number host (or “localhost” for the local host). The gschdtx command has two modes: a run mode and a kill mode. The run mode options are -c -i -w and -f. The kill-mode option is -k.

The -c option specifies the default channel. This default channel is usually overridden on each line of the data file.

The -i option specifies the number iterations to run the entire file.

The -w option specifies if you want gschdtx to wait until the schedule is complete. This wait option runs gschdtx in the “foreground”, and omitting -w runs gschdtx as a background process. When you run gschdtx without the -w option, it will return a process id. You can then use this process id to prematurely kill the running schedule using the -k option.

The -f argument specifies the data file.

The -k option specifies that you want to kill the process with process id “id”.

Here is an example of gschdtx invocation:

```
gschdtx localhost -i 10 -f myDataFile.txt
```

This command will run the myDataFile.txt file schedule ten times. If this command returns ID 123, you can use the -k option to kill the schedule as follows:

```
gschdtx localhost -k 123
```

Each line of the data file has the following format:

```
[-s sleep] [-q quantity] [-p period] [-c channel] -h header bytes -d data
bytes ... [-e extra bytes ...]
```

The -s option specifies the time in milliseconds to wait before starting this message. The default is zero.

The -q option specifies the number of times to transmit this message. The default is one.

The -p option specifies the time in milliseconds to wait between each transmission of this message. The default is zero.

The -c option specifies the channel for this message, and overrides the default channel set on the command-line, if any.

The -h argument specifies the message header bytes to transmit. You can transmit either one, two, or four header bytes.

The -d argument specifies the message data bytes to transmit.

An example myDataFile.txt data file might look like the following:

```
-c 2 -p  50 -h 01 02 -d 01 02 03 04 05 06 07 08
-c 2 -p 100 -h 03 04 -d 11 22 33 44 55 66 77 88
-c 2 -p  50 -h 05 F6 -d A1 B2 C3 D4 E5 F6 87 98
```

gschdtx will transmit the first message once, and wait for 50ms. Then, gschdtx will transmit the second message and wait for 100ms, and transmit the third message and wait for 50ms. If you run gschdtx with a -i iterations option, it will run this sequence of these three messages for that many iterations. For example, the following gschdtx command line will transmit thirty messages of the above data file:

```
gschdtx localhost -i 10 -f myDataFile.txt
```

The above example is CAN data, but gschdtx works with any protocol data, such as J1708, DLC, J1850, and LIN. You can also mix channels and protocols in the data file.

While your schedule is running, you can use gryphrx (pronounced GRYPH-R-X) to monitor your data transmissions.

9 SUMMARY

In this Application Note, you have seen how to perform channel control with the command gchanctl. You have also see how to issue any Gryphon I/O control command to a card using the gioctl command. You have see how to use these commands together to easily change the baud rate to one of the preset values. You have also seen how to construct a schedule data file, and run that schedule using gschdtx.

FURTHER READING

"Gryphon C++ Class Library User's Manual", 2006 Dearborn Group, Inc.

"Gryphon C Library User's Manual", 2006 Dearborn Group, Inc.

REFERENCES

Gryphon Application Note 2013-1, Using Gryphon Utilities.

Gryphon web page documentation.

"Gryphon C++ Class Library User's Manual", 2006 Dearborn Group, Inc.

Wireshark User's Guide for Wireshark 1.4, Copyright 2004-2010 Ulf Lamping, Richard Sharpe, Ed Warnicke.

ATTRIBUTIONS

Linux is licensed under the GNU General Public License (GPL) and the full name of the software system is GNU/Linux.

Ethernet is the name of the IEEE 802.3 standard LAN technology.

Wireshark is a free and open-source network packet analyzer released under the terms of the GNU General Public License (GPL).

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

VirtualBox is a registered trademark of Oracle Corporation.

The stylized name Gryphon®2 is a registered trademark of DG Technologies, Inc. The name Gryphon, Gryphon 2, and Gryphon S4 are trademarks of DG Technologies, Inc.

Unified Modeling Language (UML) is a standardized general-purpose modeling language adopted by the Object Management Group (OMG) and accepted by the International Organization for Standardization (ISO) as industry standard for modeling software-intensive system.

COPYRIGHT NOTICE

CONTACT INFORMATION

U.S. Sales and Corporate Office
DG Technologies, Inc.
33604 West Eight Mile Road
Farmington Hills, Michigan 48335-5202
Phone: 248.888.2000
sales@dgtech.com
http://store.dgtech.com

Heavy-Duty Development Center
2415 Directors Row, Suite G
Indianapolis, IN 46241
Phone: 317.248.9332

World-Wide Technical Support
Phone: 248.888.2000
http://www.dgtech.com/support/support_guide.html
techsupp@dgtech.com